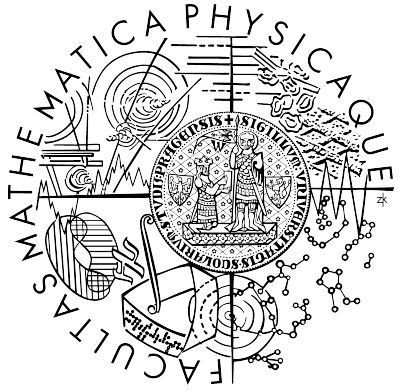
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Petr Šťastný

Monitoring DNS serverů domén druhé úrovně

Katedra softwarového inženýrství

Vedoucí diplomové práce: RNDr. Ing. Jiří Peterka

Studijní program: Informatika
Studijní obor: softwarové systémy

Poděkování

Děkuji RNDr. Ing. Jiřímu Peterkovi za vedení této práce a za připomínky a rady.

Prohlášení

Prohlašuji, že jsem svou diplomovou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne 5.12.2010. Petr Šťastný

Obsah

Název práce: Monitoring DNS serverů domén druhé úrovně
Autor: Petr Šťastný
Katedra (ústav): Katedra softwarového inženýrství
Vedoucí diplomové práce: RNDr. Ing. Jiří Peterka
E-mail vedoucího: Jiri.Peterka@mff.cuni.cz
Abstrakt: Tato práce přímo navazuje na bakalářskou práci. Nejprve uvádí veškerou potřebnou teorii o protokolech HTTP, SMTP a dalších, která je následně využita k sestavení metodiky rozšiřujících testů, které ověřují dostupnost a funkčnost základních internetových služeb domén. Metodika je následně implementována jako aplikace, která s využitím distribuovaného zpracování umožňuje automatizovaně analyzovat velké množství domén. Ze získaných výsledků jsou následně sestaveny statistické výstupy. Jedna kapitola je též věnována přehledu útoků na DNS a možnostmi zabezpečení DNS serverů a záznamů domén.
Klíčová slova: DNS, domény, DNSSEC, distribuované systémy

Title: Monitoring of SLD DNS servers
Author: Petr Šťastný
Department: Department of Software Engineering
Supervisor: RNDr. Ing. Jiří Peterka
Supervisor's e-mail address: Jiri.Peterka@mff.cuni.cz
Abstract: This publication directly follows the bachelor thesis. It contains necessary theory of HTTP, SMTP and some other protocols and services. This knowledge is then used to draw a methodology to build additional tests to verify availability and functionality of basic Internet services of a domain name. This methodology is then implemented as an application that uses distributed processing to analyse a large number of domains. Obtained results are then compiled into statistical outputs. One chapter is also devoted to overview of the attacks on DNS and security options of DNS servers and domain records.
Keywords: DNS, domains, DNSSEC, distributed systems

Kapitola 1 – Úvod

1.1 Dosavadní zkušenosti s aplikací

On-line aplikace na analýzu jednotlivých domén, která byla vyvinuta v rámci mé bakalářské práce, se setkala s úspěchem u internetové veřejnosti. Denně je využívána stovkami uživatelů, kteří tento nástroj používají či hledají informace o DNS. Ozvalo se několik počítačových odborníků, kteří tento nástroj používají denně při své práci při údržbě svých domén. S nástrojem nenastaly žádné problémy, funguje spolehlivě bez potřeby jakékoliv údržby. Překlad rozhraní do anglického jazyka způsobil, že je využíváno množstvím zahraničních uživatelů (a více než česká verze). Např. za období 1.7.2009 – 15.9.2009 bylo anglické rozhraní použito k 16300 testům, české rozhraní ve stejném období k 5900 testům.

Nástroj je však zajímavý jen pro testování jednotlivých domén. Sice byl při dokončování bakalářské práce proveden hromadný test všech domén druhé úrovně pod TLD .cz, avšak bylo tak učiněno pouze jednorázově, tedy bez možnosti sledování vývoje stavu v čase a bez možnosti porovnání s jinými TLD. Nástroj také neměl žádnou podporu pro automatizaci hromadných testů, vše šlo dělat pouze ručně s nutností vysokého pracovního nasazení obsluhy (ruční příprava seznamu domén, jejich vložení do systému, ruční spuštění hromadné analýzy, ruční sestavení výsledků). Navíc toto hromadné zpracování bylo možné dělat pouze na jednom počítači a úkoly nebylo možné distribuovat na více uzlů.

1.2 Cíl práce

Tato práce navazuje na mou bakalářskou práci a má několik cílů. Jedná se o rozšíření aplikace na analýzu stavu DNS serverů domén o několik dalších testů, které zkoumají funkčnost některých dalších služeb, které souvisí s doménovými názvy na Internetu – jedná se zejména o zkoumání WWW a e-mailových služeb. Hlavním cílem práce je vytvořit nadstavbu pro plně automatizovanou analýzu velmi velkého množství domén s využitím distribuovaného zpracování a následné zpracování analýzy se statistickými výstupy. To vše bude zaměřeno zejména na testování všech domén druhé úrovně (SLD)

pod několika sledovanými TLD. Dalším, především teoretickým tématem této práce, bude otázka bezpečnosti protokolu DNS. Cílem je vytvořit ucelený přehled slabin protokolu DNS, možných útoků na tento protokol a způsoby obrany proti nim a zajištění důvěryhodnosti a integrity dat.

1.3 Jak číst tuto práci

Pro správné pochopení této práce se předpokládají teoretické znalosti DNS, tedy zejména principy DNS systému a příslušného protokolu. V této práci budou uváděny pouze další rozšiřující informace. V části 1.5 jsou shrnuty podstatné informace z bakalářské práce, které jsou důležité pro tuto diplomovou práci.

Kapitola 2 poskytuje teoretické základy pro implementaci dalších testů, které se budou u domén provádět. Zmiňuje se o protokolech HTTP, SMTP a SIP v rozsahu, který je potřeba pro naše testy a jejich implementaci, a dále nabízí krátké seznámení se záznamy SPF a technologií DNSBL.

Kapitola 3 se zaměřuje na otázky bezpečnosti protokolu DNS a dopady na používání služeb DNS v síti Internet, zmiňuje nejznámější metody útoků proti protokolu DNS či s využitím tohoto protokolu. Další část uvádí informace o rozšíření DNSSEC protokolu DNS, která přináší bezpečností prvky pro zajištění autentizace, ověření původu a integrity dat. Závěrem uvádí také zamyšlení nad přínosy, výhodami a nevýhodami DNSSEC.

Kapitola 4 na základě teoretických znalostí zavádí další rozšiřující testy WWW, e-mailových a telefonních služeb a bezpečnostních prvků domén.

Kapitola 5 obsahuje informace o implementaci nové verze aplikace, zahrnující změny již existujících částí, rozšíření o nové testy a o plnou automatizaci hromadného provádění testů, nadstavbu pro distribuované zpracování a zpracování výsledků.

Kapitola 6 popisuje způsob provedení analýz, které byly s využitím vyvinuté distribuované aplikace provedeny na všech doménách CZ a SK, a obsahuje získané

výsledky a porovnání s výsledky z analýzy, provedené v rámci bakalářské práce v roce 2007.

1.4 Motivace

Stejně jako v případě bakalářské práce, i tato diplomová vychází z dlouholetého zájmu autora o DNS a ze zkušeností při implementaci a údržbě DNS serverů a souvisejících služeb. Navíc v dnešní době je stále důležitější myslet také na zabezpečení služeb a počítačových sítí, bez toho to dnes prostě nejde, a tak i tomuto tématu, i když pouze v teoretické rovině, je věnováno mnoho místa. Zajímavým tématem je také distribuované zpracování, díky kterému si systém může poradit s opravdu velkým množstvím domén k analýze.

1.5 Shrnutí bakalářské práce

V úvodu bakalářské práce, na kterou se v této diplomové práci přímo navazuje, byly uvedeny podrobné teoretické znalosti o doménový názvech a protokolu DNS, byly vysvětleny primární a sekundární DNS servery, cachovací DNS servery, základní DNS záznamy a jejich použití (SOA, A, AAAA, NS, MX, CNAME, PTR) a glue záznamy. V kapitole o protokolu DNS byl vysvětlen způsob komunikace, rekurzivní a nerekurzivní dotazy, zónové transfery (AXFR), autoritativní a neautoritativní odpovědi a formát DNS zpráv.

Hlavní částí bakalářské práce byl návrh metodiky analýzy záznamů domén na DNS serverech. Jednalo se o ty nejzákladnější testy:

- odpověď serverů
- sériová čísla zóny
- autoritativita serverů pro doménu
- nastavení potřebných glue záznamů
- shoda glue záznamů a A záznamů v zóně domény
- přítomnost NS záznamů v zóně domény
- shoda NS záznamů se seznamem autoritativních serverů
- rekurzivní dotazy

- veřejné AXFR
- DNS servery na veřejných IP adresách
- doporučený počet DNS serverů
- TTL hodnoty u NS záznamů na nadřazeném DNS serveru
- TTL hodnoty u NS záznamů v zóně
- reverzní záznamy DNS serverů
- různé autonomní systémy (AS) DNS serverů
- různé podsítě DNS serverů
- různé IP adresy DNS serverů
- server ze SOA MNAME jako NS záznam
- kontrola položky SOA MNAME
- stejné MNAME v SOA od všech serverů
- kontrola položky SOA RNAME
- doporučený tvar sériového čísla
- kontrola hodnoty SOA REFRESH
- kontrola hodnoty SOA RETRY
- kontrola hodnoty SOA EXPIRE
- kontrola hodnoty SOA MINIMUM

Aplikace, která je implementována v této diplomové práci, rozšiřuje aplikaci z bakalářské práce, výše uvedené testy ve výsledné aplikaci zůstávají a jsou přidány testy další.

V implementaci je popsán způsob komunikace s DNS servery, způsob zjišťování čísel autonomních systému a podsítí, cachování získávaných dat pro eliminaci některých opakovaných dotazů, webové rozhraní a požadavky na provoz aplikace.

Závěrem byla provedena analýza DNS serverů pro všechny domény 2. úrovně v doméně CZ. Taktéž byla zjišťována dostupnost DNS serverů těchto domén z více lokalit a software použitý na DNS serverech.

Kapitola 2 – Protokoly a služby

2.1 HTTP protokol

Hypertext Transfer Protocol (HTTP) je aplikační komunikační protokol pro přenos informací na Internetu. Používá se k přístupu na WWW stránky a související data (např. obrázky). Je to tedy zejména protokol pro komunikaci WWW prohlížeče, který stránky stahuje a zobrazuje uživateli, s WWW serverem, který stránky poskytuje. HTTP verze 1.1, kterou budeme používat, je definována v RFC 2616[13]. V této kapitole jsou uvedeny základní informace o tomto protokolu v rozsahu, který je potřeba k implementaci dalších testů.

2.1.1 Požadavek klienta

Obvyklé použití protokolu HTTP spočívá v tom, že uživatel zadá ve svém prohlížeči do adresního řádku URL adresu stránky, která ho zajímá. WWW prohlížeč si z této URL adresy vezme název serveru, který přeloží na IP adresu. Prohlížeč pak otevře spojení s daným serverem, který na svém vstupu očekává požadavek. Klient zašle serveru svůj požadavek a server na něj odpoví. To vše je samozřejmě běžnému uživateli skryto a ten ani nemusí tušit, že nějaké DNS a HTTP existuje. Komunikace probíhá nad protokolem TCP většinou na standardním portu 80. Požadavek klienta vypadá obecně následujícím způsobem:

```
<metoda> <cesta> HTTP/1.1
<hlavičky>

<data POST>
```

Obsah protokolu je tedy „textový“, tj. nikoliv množství binárních kódů, ale pro člověka snadno čitelná komunikace. Tím se vyznačuje mnoho internetových protokolů (výjimkou je DNS, které musí být maximálně komunikačně úsporné).

Na prvním řádku požadavku se uvádí metoda (požadovaná operace), cesta (URL adresa) a označení použitého protokolu.

Metoda říká, jaká operace se má na serveru provést. Metod, které definuje HTTP/1.1 je mnoho, ale nejvíce používanými metody jsou:

- GET – získání objektu (souboru) s danou URL
- POST – odeslání dat z formuláře a získání objektu s danou URL
- HEAD – získání objektu s danou URL bez těla (obdržíme pouze hlavičky odpovědi)

Následuje cesta, resp. část URL adresy, udávající cestu k objekt na WWW serveru, který nás zajímá. Za názvem objektu (souboru) mohou být uvedeny URL parametry.

Na dalších řádcích, kterých může být libovolné množství, se uvádějí další hlavičky, tedy dodatečné informace, které ovlivňují zpracování požadavku nebo formát odpovědi. Hlavičky jsou ve formátu „klíč: hodnota“. Některé nejpoužívanější a zajímavé hlavičky požadavku:

- **Accept** – MIME typy dat, které WWW prohlížeč akceptuje a je schopen je interpretovat uživateli
- **Referer** – URL adresa, ze které uživatel přišel na tuto stránku (na které byl na požadovanou stránku odkaz)
- **Accept-Language** – seznam jazyků, které prohlížeč akceptuje nebo spíše preferuje (závisí na nastavení jazyka uživatele)
- **Accept-Encoding** – podporované metody komprese dat odpovědi (např. gzip, deflate), umožňuje to výrazně snížit objem přenášených dat
- **User-Agent** – řetězec, identifikující prohlížeč (software, verze, operační systém a další informace). Podle toho může server přizpůsobit stránku prohlížeči podle jeho schopností a podporovaných rozšíření.
- **Host** – jediný povinný parametr, obsahující doménové jméno z URL adresy a identifikující web, který nás na WWW serveru zajímá. Je povinný z toho důvodu, že WWW servery většinou provozují na jedné IP adrese a jednom portu WWW stránky pro více různých domén (virtualhostů) a je nutné mezi nimi rozlišit.
- **Connection** – způsob ukončení spojení či provedení více HTTP požadavků v jednom TCP spojení

    - **Close** - server po odeslání celé odpovědi ukončí TCP spojení, pro získání dalšího objektu ze stejného serveru je nutno navázat spojení nové
    - **Keep-Alive** – spojení není po odeslání odpovědi uzavřeno, server očekává (v nějakém časovém limitu) další požadavek. To šetří režii komunikace, protože v typické situaci mimo samotné stránky je potřeba ihned stáhnout související obrázky, kaskádové styly (CSS) a další objekty, které se na stránce vyskytují.
- **Cookie** – klient posílá serveru data cookie, uložená na počítači uživatele
- **If-Modified-Since** – klient zasílá datum a čas posledního stažení stejného objektu, server mu tento objekt pošle pouze pokud se od té doby změnil (jinak ho má klient stále u sebe v cache a není potřeba jej znovu přenášet po síti).

Pokud se požadavek provádí metodou POST, následuje jeden prázdný řádek, POST data (data pocházející z odeslaného formuláře) a opět prázdný řádek, označující konec požadavku.

Zadáme-li v prohlížeči např. URL adresu http://www.seznam.cz/firma/info.php, náš WWW prohlížeč přeloží název „www.seznam.cz“ na IP adresu, na server s danou IP adresou naváže TCP spojení na port 80 a odešle přibližně následující požadavek a čeká na odpověď:

```
GET /firma/info.php HTTP/1.1
Host: www.seznam.cz
```

2.1.2 Odpověď serveru

Odpověď má velmi podobný formát. Na prvním řádku je nejprve uvedena verze HTTP protokolu a uveden trojciferný kód, označující výsledek operace, popř. číslo chyby, a dále textový popis významu kódu. Obvyklé kódy jsou:

- **1xx** – informační hlášky – téměř nepoužívané
- **2xx** – úspěch
    - 200 OK – požadavek proběhl v pořádku
- **3xx** – přesměrování (liší se důvody)

    - 301 Moved Permanently – server sděluje klientovi v hlavičce „Location“ nové stálé umístění požadovaného objektu (klient si může novou adresu zapamatovat a příště ji rovnou použít, původní již nebude platit)
    - 302 Found (Moved Temporarily) – podobné jako předchozí, jedná se však o dočasné přesměrování (klient by si ho měl následovat, ale neměl by si ho zapamatovat)
    - 304 Not modified – odpověď na požadavek s hlavičkou If-Modified-Since, pokud se požadovaný objekt od poslední návštěvy nezměnil
- **4xx** – chyba na straně klienta – v požadavku je chyba nebo nemůže být vyřízen
    - 400 Bad Request – syntaktická či jiná chyba v požadavku
    - 401 Unauthorized – pro přístup k danému objektu je vyžadována autentizace nebo nebyla úspěšná
    - 403 Forbidden – požadavek je v pořádku, ale server jej odmítl provést, např. přístup k nepovolenému zdroji
    - 404 Not Found – požadovaný objekt nenalezen
- **5xx** – chyba na straně serveru – požadavek je v pořádku, ale z důvodu chyby na serveru nelze provést
    - 500 Internal Server Error – vnitřní chyba serveru, také v případě chyby ve skriptu

Na dalších řádcích se mohou vyskytovat hlavičky, které doplňují informace k odpovědi. Mezi nejpoužívanější patří:

- **Content-Encoding** – metoda komprese těla odpovědi (gzip, deflate, …) - server použije kompresi pouze pokud to klient v hlavičce požadavku nabídl
- **Content-Language** – jazyk textu v těle odpovědi
- **Content-Length** – velikost těla v bytech
- **Content-Type** – MIME typ objektu v těle odpovědi, volitelně také použitá kódová stránka
- **Date** – datum a čas zpracování požadavku (podle hodin serveru)
- **Location** – URL adresa cíle přesměrování, pokud se přesměrovává
- **Server** – informace o softwaru serveru – často zde bývá uveden software a verze WWW serveru, skriptovacího jazyka a dalších rozšíření
- **Set-Cookie** – cookie, která si má klient uložit
- **Transfer-Encoding** – možný příznak rozdělení těla odpovědi na více částí

Za poslední hlavičkou následuje prázdný řádek, podle kterého klient pozná začátek těla. Je důležité uvést, jakými způsoby sděluje server klientovi, jak velké je tělo odpovědi. Klient potřebuje tuto informaci znát, aby poznal konec odpovědi, tedy aby rozpoznal, zda nepřicházející data znamenají, že nastal konec zpracování požadavku nebo zda se ještě další data na serveru připravují a budou přijata za chvíli. Původním řešením bylo uvedení hlavičky Content-Length, která obsahovala celkovou velikost těla v bytech. To je však omezující, protože server musí v takovém případě ještě před zahájením odesílání dat těla znát celkovou velikost odpovědi (jelikož hlavičky musí být odeslány před tělem), a tudíž nemá možnost průběžně odesílat klientovi data, ale musí počkat, až je celé tělo na serveru sestaveno a pak jej může poslat v celku. Protokol HTTP 1.0 umožňoval pouze tento způsob.

Lepší variantou, jak to umožňuje protokol HTTP verze 1.1, je rozdělení odpovědi do několika bloků dat (chunked response). Server uvede v odpovědi hlavičku **Transfer-Encoding: chunked**. Následně zasílá data po blocích. Před každým blokem je na vlastním řádku uvedena jeho délka (hexadecimálně). Celé tělo odpovědi končí zasláním bloku velikosti 0. Díky tomu může server odesílat klientovi výsledky průběžně, aniž by věděl, kdy výstup skončí.

Příklad odpovědi s přesměrováním:

```
HTTP/1.1 302 Found
Date: Thu, 28 Aug 2008 18:23:52 GMT
Server: Apache
Location: /navigation.html
Content-Length: 0
Content-Type: text/html
```

Zde můžeme vidět, že nás server přesměrovává na jinou stránku a prozrazuje o sobě, že je použit WWW server Apache.

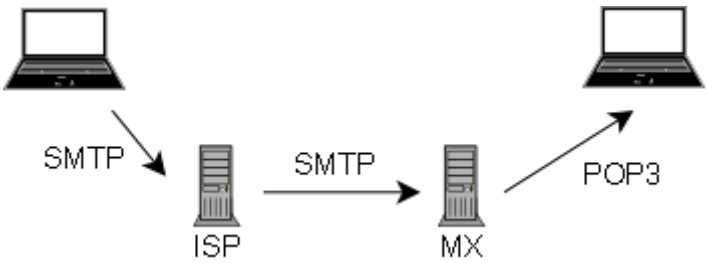
Příklad odpovědi s tělem a s chunked přenosem:

```
HTTP/1.1 200 OK
Content-Type: text/plain
Transfer-Encoding: chunked
```

```
13
Prvni blok odpovedi
1e
Druhy a posledni blok odpovedi
0
```

2.2 SMTP protokol

Simple Mail Transfer Protocol (SMTP) slouží k přenosu e-mailových zpráv. Používá se jak k přenosu zprávy od klienta (z jeho e-mailové klienta) na SMTP server ISP, tak také pro další doručení mezi mailservery. Už se ale nepoužívá ke stažení zprávy koncovým příjemcem z jeho schránky (k tomu jsou určeny např. protokoly POP a IMAP). Je to opět klasický TCP/IP protokol, klient zasílá textové příkazy a informace a server na ně reaguje zasláním odpovědi. Komunikace je zde výrazně intenzivnější oproti HTTP protokolu. SMTP byl původně definován v RFC 821[14], následně aktualizován v RFC 2821, nedávno však byl vydán nový dokument RFC 5321[4], který dříve zmíněné dokumenty nahrazuje.



Ilustrace 1: schéma doručování e-mailů přes SMTP servery

2.2.1 Základní komunikace

Klient naváže TCP spojení se serverem na portu 25. Klient zasílá textové příkazy, většinou jedno klíčové slovo, za ním následují parametry příkazu. Server na příkazy od klienta reaguje zasíláním odpovědí ve formě tříciferného kódu, za kterým následuje slovní popis výsledku (kladné odpovědi či chyby). Tento slovní popis nemá pro servery většinou žádný význam, ty se řídí pouze kódy (jsou výjimky, např. odpověď na příkaz EHLO, viz. dále). Smysl má pro lidi, kterým význam kódu osvětlí anebo lze v textu

najít podrobnosti o chybě apod. Oddělovačem příkazů je nový řádek <CR><LF>. Odpovědí může server na jeden příkaz zaslat více. Za tříciferným kódem odpovědi uvede buď znak pomlčky (není poslední) anebo mezeru (je poslední).

Dělení návratových kódů a jejich příklady:

- **1xx** – předběžné kladné odpovědi – server akceptoval požadavek a čeká na jeho potvrzení nebo zamítnutí
- **2xx** – konečné kladné odpovědi
  - 220 – uvítací zpráva
  - 250 – obecná kladná odpověď
- **3xx** – průběžné kladné odpovědi – server akceptoval požadavek a čeká na další související data
  - 354 – počátek čtení obsahu obálky
- **4xx** – negativní odpověď, problém lze napravit – např. dočasný problém na serveru, pokusíme se s ním spojit později
- **5xx** – negativní odpověď, závažný problém

Po otevření TCP spojení zasílá server uvítací hlášku (často uvede svůj software a jeho verzi), čímž dává najevo, že je připraven přijímat příkazy. Klient se následně představí pomocí příkazu HELO (pokud nepodporuje rozšíření ESMTP) nebo EHLO. Jestliže server nepodporuje ESMTP, reaguje na příkaz EHLO chybou (neznámý příkaz), klient odpoví příkazem HELO a komunikace bude probíhat bez rozšíření.

Pro úspěšné předání zprávy serveru je potřeba zaslat serveru tyto údaje:

- e-mailová adresa odesílatele „obálky“ (příkaz MAIL FROM)
- e-mailová adresa příjemce „obálky“ (příkaz RCPT TO) – těchto příkazů lze odeslat více, pokud má server doručit zprávu více příjemcům
- obsah obálky (příkaz DATA)

Po příkazu DATA čte server vstup až do znaku tečka na samostatné řádce (tedy až do sekvence `<CR><LF>.<CR><LF>`). SMTP protokol neřeší formát a kódování těla zprávy. K tomu se používá standard MIME, definovaný v několika RFC. MIME umožňuje použít různé znakové sady, těla zpráv s více částmi, netextové přílohy aj.

Po odeslání těla zprávy se klient rozloučí příkazem „QUIT“, server pošle také své rozloučení a ukončí spojení. na rozdíl od protokolu HTTP, SMTP neumožňuje během jednoho TCP spojení odeslat více zpráv. Samozřejmě je možné udělat takové rozšíření, které by to umožnilo.

Následuje příklad celé SMTP komunikace bez použití rozšíření. Všechny řádky, začínající trojciferným kódem, jsou odpovědi serveru.

```
> 220 ZMailer SMTP-server running at Centrum.cz
< HELO klient
> 250 mail8.centrum.cz
< MAIL FROM: <petr@stastny.eu>
> 250 2.1.0 Sender syntax Ok;
< RCPT TO: <petr.stastny@centrum.cz>
> 250 2.1.5 Recipient address syntax Ok
< DATA
> 354 Start mail input; end with <CRLF>.<CRLF>
< obsah zpravy
< .
> 250 2.6.0 message accepted
< QUIT
> 221 2.0.0 mail8.centrum.cz Out
```

Tím server zprávu převzal a doručí ji do schránky příjemce (v tomto případě, kdy je zpráva v cíli), případně provede předání zprávy na další SMTP server.

2.2.2 Serverová rozšíření

Samotný protokol SMTP tak, jak byl původně definován v RFC 821[14], podporoval jen základní množinu příkazů. Proto byl také zaveden mechanismus rozšíření (extensions), který umožňuje, aby SMTP server klienta informoval o dostupných rozšíření a SMTP klient je podle svého rozhodnutí mohl využít.

Aby bylo možné odlišit klienta a server SMTP protokolu, kteří rozšíření podporují a kteří nikoliv, přibyl příkaz EHLO, který klient serveru zasílá namísto příkazu HELO při uvítání. Jestliže server rozšíření podporuje, reaguje na tento příkaz kladně a vypíše seznam dostupných rozšíření. Pokud server rozšíření nepodporuje, odpoví, že příkaz

EHLO nezná, klient reaguje příkazem HELO a komunikace dále pokračuje bez rozšíření starým způsobem. Mechanismus rozšíření byl definován v dokumentu RFC 1869 [17].

Každé rozšíření může přidat nové SMTP příkazy, změnit chování existujících příkazů či přidat další parametry k příkazům MAIL a RCPT. Seznam standardních rozšíření eviduje organizace IANA na stránce http://www.iana.org/assignments/mail-parameters. Nestandardní či experimentální rozšíření se pojmenovávají s prefixem „X-“.

Následuje příklad začátku komunikace s mailserverem domény centrum.cz, kde můžeme vidět množství podporovaných rozšíření:

```
> 220 ZMailer SMTP-server running at Centrum.cz
< EHLO u-pl2.ms.mff.cuni.cz
> 250-mx1.centrum.cz Hello u-pl2.ms.mff.cuni.cz
> 250-SIZE 22000000
> 250-8BITMIME
> 250-PIPELINING
> 250-CHUNKING
> 250-ENHANCEDSTATUSCODES
> 250-DSN
> 250-X-RCPTLIMIT 10000
> 250-ETRN
> 250 HELP
```

Některá zajímavá rozšíření:

- **HELP** – server nabízí nápovědu ve formě čitelné pro lidi, nápovědu získáme zasláním příkazu „HELP“
- **8BITMIME** – přenos těla zpráv v 8bitovém kódování (SMTP standardně umožňuje přenos jen v 7bitovém ASCII) (RFC 1652[15])
- **SIZE** – server může oznámit maximální povolenou velikost zprávy, kterou dovolí přijmout, a klient může oznámit velikost zprávy, kterou se chystá předat k doručení (RFC 1870[16]), podle toho server může zprávu odmítnout ještě před posláním dat
- **CHUNKING** – předání těla zpráv po blocích dat – podobné chunked kódování těla HTTP odpovědi (RFC 3030[18])

- **PIPELINING** – předávání SMTP příkazů ve větších dávkách tak, aby klient nemusel často čekat na odpověď serveru (RFC 2920)
- **DSN** – konfigurace oznamování úspěchu a neúspěchu při doručování zpráv
- **ETRN** – vyprázdnění fronty čekajících zpráv – pro případy, kdy se cílový mailserver připojuje k Internetu jen občas, a nemůže tak čekat na pravidelné opakované doručování podle timeoutů (RFC 1985)
- **ENHANCEDSTATUSCODES** – rozšíření pro upřesnění návratových kódů
- **STARTTLS** – možnost přeměny spojení mezi klientem a serverem z nešifrovaného na šifrované (server a klient se dohodnou na přidání vrstvy komunikace)
-

2.2.3 Greylisting

Greylisting je metoda ochrany proti spamu, využívající opakovaného doručování zprávy v případě, že je cílová e-mailová schránka dočasně nedostupná. Jedná se o poměrně účinný způsob, avšak má zásadní nevýhody.

Spamovací roboti jsou většinou nastaveni tak, že jejich cílem je co nejrychleji rozeslat velké množství e-mailových zpráv a přitom ignorovat všechny případné chyby. Toho se využívá v greylistingu, jehož implementace je velice jednoduchá. Při prvním pokusu o doručení obdrží klient po příkazech MAIL a RCPT chybovou hlášku s kódem 450 (či podobným), což značí dočasnou nedostupnost cílové e-mailové schránky. Server si zapamatuje trojici:

- IP adresa klienta
- e-mailová adresa odesílatele
- e-mailová adresa příjemce

Podle standardu SMTP komunikace musí klient v případě této chyby zprávu uložit k opakovanému pokusu o doručení. Při druhém doručování již e-mailový server zprávu se stejnou trojicí akceptuje. Většina spamovacích serverů se o další doručení nepokouší, a tak se zpráva od něj nedoručí.

Účinnost této metody je velmi vysoká, avšak přichází s ní také některé problémy. Výhodou tohoto řešení je, že má velice nízké nároky na systémové prostředky serveru a

sítě – tělo zprávy se vůbec na server nepřenese (zpráva je odmítnuta ihned po příkazu RCPT), server nemusí provádět složité antispamové a antivirové kontroly, při kterých provádí analýzu obsahu zprávy. Greylisting se nemusí nijak složitě konfigurovat, není potřeba pravidelně aktualizovat nějaká pravidla apod. Nesmí být však používán jen jako jediný způsob ochrany proti spamu, jelikož se dá velice snadno obejít odesíláním jednoho spamu dvakrát s určitým časovým odstupem.

Druhé doručování pro jednu trojici je vždy zpožděno v řádu desítek minut, hodin či dokonce dní, kdy zpráva čeká ve frontě klienta k opakovanému pokusu o doručení (záleží na jeho konfiguraci). Toto zpoždění ale může být zásadní např. při důležité obchodní komunikaci. Další zprávy se stejnou trojicí se již doručují hned. Tento problém lze trochu zmenšit nastavením whitelistu, tedy např. uvedením důvěryhodných IP adres odesílatelů. Větší problémy přináší situace, kdy se „hodný“ e-mailový klient neřídí přesně podle RFC a o opakované doručení se nepokusí, žádaná e-mailová zpráva se tím ztratí. Další problém nastane v případě, že se klient pokusí o další doručení z jiné IP adresy (např. pokud jsou zprávy odesílány ze serverové farmy). Také se může stát, že klient při nějaké poruše či havárii přijde o obsah fronty zpráv k opakovanému doručení.

2.2.4 Odesílatelé spamu a záznam typu SPF

Stejně jako na světě přibývá spamu a množí se způsoby, jak obcházet současná opatření proti němu, vznikají i nové technologie a postupy, jak se spamům bránit. Obvyklým problémem je, že odesílatel spamu (resp. jakéhokoliv e-mailu) může do hlavičky zprávy uvést libovolnou e-mailovou adresu odesílatele. Paralelu můžeme najít u klasické papírové pošty, u které můžeme také na obálku napsat libovolné jméno a adresu odesílatele, nic nám v tom nezabrání a nikdo nemá možnost si zkontrolovat správnost těchto informací. To, že jsou nepravdivé, pochopí po přečtení až příjemce zprávy. Spammeři tedy pro adresy odesílatele samozřejmě nepoužívají svoji adresu, ale libovolné a často náhodné adresy s náhodnými doménami.

To má navíc nepříjemný efekt – pokud se nedaří spam doručit příjemci, doručující SMTP server odešle chybovou hlášku na onu adresu odesílatele. Zahlceni tedy nejsou pouze příjemci, ale také lidé, jejichž adresy bylo zneužito pro rozesílání spamu. Majitelé a správci domén o tom jistě vědí své. Tomuto doručování chybových hlášek se můžeme

bránit tak, že si dáme záležet na konfiguraci mailserveru naší domény. I z mnoha dalších důvodů je vhodné nepoužívat tzv. doménový koš (tj. zprávy pro neznámého příjemce jsou doručovány do určité speciální schránky), ale mít jasně stanovený seznam adres příjemců, které na naší doméně akceptujeme, a ostatní adresy striktně odmítat. Tím velmi snížíme pravděpodobnost, že se spammer trefí do adresy odesílatele, které skutečně existuje.

Výše uvedené opatření však pouze sníží množství přijímaného spamu pro náhodně generované příjemce na naší doméně a také sníží množství zpětně přijatých chybových hlášek o nemožnosti doručení spamu. Spam jako takový však bude stále s uvedením naší domény rozesílán. A často to dopadá tak, že správci serverů, které jsou postiženy návalem nevyžádaných zpráv, začnou křičet na nás, proč spam rozesíláme my, i když my za to vůbec nemůžeme.

Pokusem o řešení tohoto problému je tzv. systém SPF (Sender Policy Framework), který je definován v RFC 4408[9]. Jedná se o systém, který umožňuje majitelům domén definovat pravidla, říkající které uzly v Internetu jsou oprávněny posílat e-maily s adresou odesílatele z dané domény. Prakticky to vypadá tak, že majitel domény zanese do zóny speciální záznam TXT, ve kterém je uveden seznam IP adres počítačů, ze kterým mohou pocházet e-maily s danou doménou. Typicky to bude seznam IP adres firemních počítačů, serverů, firemní SMTP brány apod. Tento systém musí být samozřejmě podporován ostatními servery. Pokud SMTP server obdrží e-mail a k doméně odesílatele získá dotazem do DNS záznam typu TXT, který obsahuje SPF pravidla, provede kontrolu. E-mail pak propustí jen pokud pravidlům vyhovuje. Jinak jej odmítne. Kontrola se provádí ještě před zasláním těla zprávy (po odeslání příkazu MAIL FROM). SMTP server, který SPF nepodporuje, jej nezkontroluje a e-mail propustí dál.

Zápis pravidel do záznamu typu TXT byl původně vymyšlen pro možnost rychlého nasazení této technologie. Brzy poté byl však definován speciální typ DNS záznamu – SPF a od používání TXT záznamů pro tyto účely by se mělo upustit. Syntakticky jsou oba typy záznamů stejné.

Systém SPF je poměrně mladý a příslušné RFC je označeno jako experimentální, tato technologie tedy není zatím příliš rozšířena. Aby bylo možné pomocí ní vymýtit spam, musely by mít všechny domény na světě příslušné (a správně nadefinované) SPF záznamy a všechny SMTP servery na světě by musely provádět příslušné kontroly. Na správné nastavení pravidel v SPF záznamu je potřeba dávat si velký pozor. Pokud omylem na nějaký počítač či nějakou podsíť zapomeneme, může nastat problém s doručováním regulérních e-mailů. Jsou však domény, u kterých SPF pravidlo nastavit nelze – např. u freemailových služeb typu seznam.cz, z nich se e-maily odesílají odkudkoliv.

Takto například vypadá příslušný TXT záznam domény ietf.org:

```
v=spf1 ip4:64.170.98.0/26 ip4:64.170.98.64/28 ip4:64.170.98.80/28
ip4:64.170.98.96/29 ip4:208.66.40.224/27 ip6:2001:1890:1112:1::0/64
-all
```

V příkladu vidíme jasně nadefinované podsítě, které jsou oprávněny odesílat e-maily s adresou, obsahující doménu ietf.org.

2.2.5 DNSBL

Předchozí technologie řešila evidenci a distribuci seznamu povolených IP adres, které mohou rozesílat e-maily jménem konkrétní domény. Dnes se však filtrování zdrojů spamů většinou řeší opačným způsobem – evidují se a distribuují seznamy IP adres, které jsou zdrojem spamu a které by měly být blokovány. Většina SMTP serverů nějaké takové blacklisty má – buď mohou být definovány administrátorem serveru, který je však nucen je ručně spravovat a doplňovat, anebo SMTP servery komunikují s blacklist servery, o které se starají buď dobrovolníci anebo poskytování seznamu IP adres spammerů může být založeno na komerčním principu.

Hlavní otázkou je, jak vlastně takové seznamy blokovaných IP adres, často velmi obsáhlých, distribuovat. Buď si může SMTP server k sobě celý seznam stáhnout, používat jej a periodicky provádět aktualizaci tohoto seznamu (takto fungují spíše ty komerční blacklisty), anebo se může SMTP server dotazovat nějakého blacklistu na konkrétní IP adresu až v okamžiku potřeby (při přijetí SMTP spojení).

Poslední zmíněnou variantou se budeme zabývat. Pro ni je nejvíce rozšířena technologie DNSBL (DNS-based Blackhole List). Jak již název napovídá, přítomnost nějaké IP adresy v blacklistu se ověřuje pomocí DNS. Existuje mnoho serverů, které tuto službu zdarma a veřejně nabízí. Stačí zaslat speciálně formulovaný DNS dotaz a dozvíme se, zda danou IP adresu evidují. Jako příklad si vezměme IP adresu 1.2.3.4 a použijeme DNSBL server zen.spamhaus.org. DNS název, na který se budeme ptát, se sestaví tak, že se vezme IP adresa v opačném pořadí a za ní se připojí název DNSBL serveru. A na takto získaný název vzneseme DNS dotaz na A záznam.

```
4.3.2.1.zen.spamhaus.org
```

Pokud je tato IP evidována, získáme jeden nebo více A záznamů. Můžeme se spokojit již s existencí těchto A záznamů anebo můžeme dále zkoumat jejich hodnotu a zjistit, z jakého důvodu je tato IP adresa na daném blacklistu. Jako hodnoty se používají speciální lokální IP adresy 127.0.0.X. Jejich význam však není standardizován a každý provozovatel DNSBL je definuje jinak.

Zde je příklad dotazu na IP adresu 88.101.70.219, která je blokována, protože je z rozsahu ADSL:

```
;; QUESTION SECTION:
;219.70.101.88.zen.spamhaus.org.        IN      A
;; ANSWER SECTION:
219.70.101.88.zen.spamhaus.org. 900 IN  A       127.0.0.11
```

Ze stránek www.spamhaus.org se dočteme, že jejich databáze, pojmenovaná PBL (Policy Block List), obsahuje seznam IP adres koncových uživatelů, kteří by neměli zasílat kamkoliv e-maily bez použití autorizace a právě tomu odpovídá „návratová hodnota“ 127.0.0.11.

DNSBL lze teoreticky provozovat s použitím libovolného DNS softwaru a v něm vytvářet zóny, obsahující názvy, které odpovídají zakázaným IP adresám. To však není dobré řešení, protože často bývají zakázané celé velké rozsahy IP adres a to může vést k velmi rozsáhlým a špatně udržovatelným souborům. Proto se na to používá speciální

software, které eviduje IP adresy lépe, ale navenek se chová jako DNS server. DNSBL samozřejmě těží z mnoha výhod, které DNS přináší – snadno lze rozložit zátěž, delegovat části map IP adres na jiné skupiny DNS serverů atd. Avšak nejpodstatnější výhoda spočívá v tom, že dotazy na DNSBL servery jsou, stejně jako jakékoliv jiné dotazy do DNS, cachovány. A tak pokud se náš mailserver neustále dokola ptá na jednoho odesílatele, měl by číst výsledek z nějaké lokální cache a DNSBL server zbytečně nezatěžovat. Na druhou stranu to znamená, že pokud dosáhneme odstranění naší IP adresy z blacklistu, budeme muset počkat na vypršení TTL oněch A záznamů. Např. spamhaus.org uvádí TTL hodnotu 30 minut.

Největším problémem DNSBL systémů je otázka spolehlivosti evidovaných seznamů IP adres. Pokud někdo skutečně spamuje, tak se nepochybně na jednom nebo více DNSBL serverech brzy objeví. Může se tam však samozřejmě objevit omylem také IP adresa, která ve skutečnosti zdrojem spamu není. Některé DNSBL servery fungují na dobrovolnickém principu, kdy lze spamující IP adresy oznámit dokonce anonymně. Jiné DNSBL zase přes vložením IP adresy do seznamu provádí podrobná ruční šetření.

Další metodou získávání totožnosti spammerů a útočníků jsou tzv. honeypots[12], což jsou obecně počítače či systémy, které se tváří jako skutečné a jsou úmyslně nechráněné či jinak zneužitelné. Ve skutečnosti jsou to však návnady, které hodní uživatelé nikdy nepoužijí, ale útočníci ano a tím padnou do pasti. V našem případě to může být mailserver pro doménu, která nikdy neměla a ani nemá nikde uvedenu jakoukoliv svou e-mailovou adresu. Pokud na ní nějaký e-mail přesto přijde, buď to znamená, že se někdo zmýlil, anebo že se jedná o spammera, který zkouší všechny možné příjemce. Takových honeypots bývá více a teprve poté, co zpráva od jednoho zdroje dorazí na více takových uzlů, je odesílatel prohlášen za zdroj spamu.

Často jsou na některých DNSBL evidovány celé rozsahy IP adres preventivně, aniž by z nich někdy přišel jakýkoliv spam – např. rozsahy používané pro dynamické přidělování IP ADSL klientům, u nichž se předpokládá, že nebudou nikdy posílat e-maily přímo, ale budou využívat oficiální SMTP brány svého ISP. Avšak zásadní problém nastane, když si bude chtít někdo např. přes ono ADSL zapojit svůj vlastní SMTP server a pak zjistí, že je všude zablokován. Pokud je někdo zablokován, má každý DNSBL server postupy,

jak provést odblokování, většinou však nejsou IP vymazány automaticky, ale musí o to být zažádáno.

2.3 Služba SIP a SRV záznamy

SIP (Session Initiation Protocol) je internetový protokol určený pro přenos signalizace v internetové telefonii. Normálně používá UDP port 5060, ale může fungovat i nad TCP/5060. [5] Tento protokol a další související věci definuje RFC 3261[6] a je dále upraven dalšími RFC. V této práci se nebudeme zabývat přímo tímto protokolem, ale uvedeme jej do souvislosti s doménami a DNS.

Chceme-li s někým telefonovat po Internetu pomocí nějaké VoIP (Voice over IP) technologie, používají se k identifikaci příjemce SIP adresy (SIP URI) ve tvaru sip:jmeno@domena. Je to podobné jako s e-maily, adresa obsahuje doménu, která nabízí potřebné telefonní služby (jedná se např. o poskytovatele telefonních služeb), a nějaké lokální jméno, které rozpozná příslušný VoIP server. Doména pak obsahuje v DNS potřebné záznamy, které jsou obdobou MX záznamů pro e-maily, ze kterých se dozvíme, na jaký server se máme obrátit, pokud chceme kontaktovat někoho z této domény. Díky tomu je možné tyto adresy oddělit od poskytovatele VoIP služeb (při změně poskytovatele si jen majitel domény změní příslušný záznam v DNS, který bude poté směřovat na servery nového poskytovatele). Konkrétně se jedná o záznamy typu SRV, které slouží nejen pro účely SIP, a informace pro SIP se nachází na speciálních subdoménách _sip._udp.domena, případně _sip._tcp.domena. Záznam typu SRV je specifikován v RFC 2782[7].

Záznam typu SRV obsahuje následující údaje:

- priorita – podobně jako u MX záznamů, spojení se navazuje nejprve se serverem s nejnižší hodnotou priority a teprve při nemožnosti spojení se zkouší další
- váha – relativní váha záznamu pro záznamy se stejnou prioritou
- číslo portu
- cílový hostitel (doménový název)

Výše zmíněný údaj váha má za úkol určit, jaké má být relativní rozložení zátěže pro servery, které mají stejnou prioritu. Má-li např. první záznam váhu 10 a druhý záznam

váhu 90, bude druhý server kontaktován přibližně v 90% případů a druhý ve zbytku. Nevýhoda tohoto rozložení je, že se jedná o statický údaj, který nezohledňuje skutečné aktuální vytížení oněch serverů. Navíc toto rozložení je relativní pouze v rámci jednoho kontaktujícího klienta, protože ten nemůže vědět, jaká je četnost komunikace od jiných klientů.

Aby to bylo zřejmé, v příkladu si ukážeme, jak bychom mohli někoho kontaktovat přes SIP protokol na doméně nic.cz. Zašleme-li na DNS server této domény dotaz na subdoménu „_sip._udp“ a typ záznamu SRV, obdržíme následující odpověď:

```
;; QUESTION SECTION:
;_sip._udp.nic.cz.         IN      SRV

;; ANSWER SECTION:
_sip._udp.nic.cz.   1800    IN      SRV     100 100 5060 sip.nic.cz.
```

Z odpovědi se dozvíme, že máme kontaktovat SIP server sip.nic.cz na UDP portu 5060. SRV záznamy mají mnoho dalších použití. Např. jej Microsoft Windows používají k hledání doménového řadiče a využívají jej další protokoly (XMMP, Kerberos, LDAP aj.).

Kapitola 3 – Bezpečnost DNS

3.1 Problémy protokolu DNS

Protokol DNS, stejně jako všechny ostatní protokoly rodiny TCP/IP, byl v rané době Internetu navržen tak, aby byl velmi jednoduchý a rychlý. Nebylo do těchto protokolů zahrnuto téměř žádné zabezpečení, protože tou dobou nebylo potřeba. Internet používal jen malý okruh lidí, kteří se navzájem znali a důvěřovali si. Podstatná byla jednoduchost, rychlost a efektivita na tehdy výrazně pomalejších linkách a méně výkonných počítačích. To si s sebou tyto protokoly nesou dodnes, kdy je celosvětová síť plná lidí, kteří se jí snaží škodit, ať je to z osobních, finančních, politických či jiných důvodů.

Pro pochopení útoků je potřeba připomenout některé aspekty protokolu. DNS klient posílá DNS serveru požadavek na překlad doménového názvu protokolem UDP. Aby bylo možné rozlišit, která odpověď patří ke kterému dotazu (což nestavový transportní protokol UDP sám o sobě neumí), vkládá klient do paketu s dotazem položku „ID“, což je 16-bitové číslo. Server do paketu s odpovědí vloží stejné ID a podle toho klient odpověď rozpozná. A právě v tomto číslování požadavků a odpovědí a nepřítomnosti šifrování či autentizace je hlavní slabina protokolu DNS.

3.2 Útoky proti DNS

V souvislosti s DNSSEC se přehledem slabin a možných útoků na DNS zabývá dokument RFC 3833 - „Threat Analysis of the Domain Name System (DNS)“[19]. Následuje shrnutí základních útoků proti DNS. Podrobné informace o některých útocích včetně praktické ukázky útoku je možné najít v mé práci „Útoky s využitím protokolu DNS“, která vznikla na MFF UK v rámci předmětu Kybernalita.

3.2.1 DNS spoofing

Tento útok spočívá v podvržení falešných odpovědí. Klienti pracují tak, že za odpověď považují první paket s danou IP odesílatele a daným ID. A nic nebrání k tomu, aby jakékoliv zařízení v síti vygenerovalo falešnou odpověď a podvrhlo IP adresu odesílatele (správného DNS serveru) a ID dotazu. Když bude rychlejší než skutečný DNS server, vnutí klientovi svou odpověď. Jen je potřeba falešnou odpověď načasovat na okamžik, kdy ji klient očekává.

Cílem je tedy klientovi podvrhnout např. jinou IP adresu webových stránek. Klient nemá možnost poznat, že dostal špatnou odpověď, a bere získanou IP adresu za správnou, a např. jeho webový prohlížeč pošle HTTP požadavek na zobrazení nějaké stránky na útočníkův WWW server. Je to pokročilejší technika phishingu, při které nelze poznat, že došlo k nasměrování HTTP požadavků na nesprávný server.

Dalšími možnými cíli může být přesměrování e-mailové komunikace domény oběti na mailserver útočníka (a čtení e-mailů nebo jejich pozměnění a přeposlání na správný mailserver), obejití antispamové ochrany, přesměrování (odposlech) telefonních hovorů v systému ENUM aj.

Potíž je, že DNS protokol neumožňuje, aby si klient mohl ověřit pravdivost získaných DNS záznamů. DNS odpovědi nemají žádný elektronický podpis či jiný mechanismus pro ověření původu dat. To přináší až rozšíření DNSSEC, o kterém bude řeč dále v této práci.

ID spoofing s odposlechem sítě – útočník má v moci nějaké zařízení v lokální síti oběti anebo nějaký router na cestě mezi obětí a skutečným DNS server a pasivně odposlouchává síťovou komunikaci. Může tedy přímo zachytávat pakety s DNS dotazy a generovat falešné odpovědi. Pokud má navíc možnost pakety s dotazem zastavit a neposlat dál skutečnému DNS serveru, nebudou se v síti vyskytovat dvojité odpovědi na jeden dotaz a problém nelze dobře detekovat.

ID spoofing s hádáním ID dotazu – útočník nemá přístup do sítě tak, aby mohl přímo odposlouchávat DNS dotazy své oběti a chybí mu potřebné informace k podvržení odpovědi. Chybí mu znalost ID odpovědi a UDP portu, na kterém klient očekává

odpověď. Ovšem to je „jen“ 2^32 možných kombinací, což je na dnešní poměry pro potřeby brutal force útoku málo. Navíc DNS klienti často používají jen několik nebo dokonce pouze jediný UDP port, na kterém očekávají odpověď, což práci útočníka výrazně usnadní. Podstatou tohoto útoku je to, že některé DNS klienti (případně servery) používají sekvenční či jinak predikovatelné číslování dotazů. Útok zde nelze provést přímo na koncový počítač oběti, protože nemůžeme zjistit jeho způsob číslování dotazů a porty, ale útok se zaciluje na rekurzivní cachovací DNS server, využívaný obětí, který si může útočník „osahat“ korektními dotazy na svou vlastní doménu. Zde je postup již komplikovanější, ale výsledkem je to, že falešná odpověď je podvržena rekurzivnímu DNS serveru, který ji pak poskytne oběti. Pokud nám práci komplikuje pravý DNS server, který posílá korektní odpovědi, umlčíme ho nebo alespoň zpomalíme jeho reakce DoS útokem. Podrobnější informace lze nalézt v mé již zmíněné práci.

Poslední zmíněný útok naznačuje, jak efekt podvržení odpovědi znásobit – nesnažíme se podvrhnout odpověď konkrétnímu koncovému počítači, ale podstrčíme ji cachovacímu DNS serveru, který je využíván např. celou příslušnou společností pro všechny její pracovní stanice. A povede-li se nám podvrhnout informaci DNS serveru nějaké ISP společnosti, hned tuto informaci mohou vidět desítky tisíc jeho klientů.

3.2.2 DNS cache poisoning

Jedná se o variantu podvržení informací, která využívá chyb v implementaci DNS serveru, spočívající v nedokonalé kontrole Additional Section v DNS odpovědích. Do nich DNS servery vkládají dodatečné záznamy, které nějak souvisí s původním dotazem a příslušnou odpovědí. Pokud si oběť špatně nebo vůbec nekontroluje záznamy v této dodatečné sekci, může si je ochotně uložit a používat je.

3.2.3 DNS Amplification Attack

V tomto případě již nejde o podvržení informací, ale o distribuovaný útok s cílem zahlcení sítě oběti s použitím DNS protokolu se zesilujícím efektem. K provedení útoku potřebujeme jeden nebo více veřejných DNS relay serverů. Těm začneme zasílat velké množství vhodných DNS dotazů a jako zdrojovou IP adresu uvádíme IP oběti. DNS servery pak odpovědi posílají k oběti. Vůbec nezáleží na tom, že oběť takové DNS

odpovědi neočekává. Zde už nejde o DNS, ale o zahlcení linky oběti velkým tokem jakýchkoliv dat. Podstatným trikem je správná volba DNS dotazů tak, aby námi odeslaný dotaz byl malý a odpověď DNS serverů velká. Pro tyto účely si můžeme připravit nějaké objemné DNS záznamy u naší domény. Údajně lze dosáhnout až 73násobného zesílení útoku při použití rozšíření EDNS, které umožňuje posílat větší odpovědi přes UDP, které nejsou limitovány 512 byty.

3.2.4 DoS útok na DNS

Neméně zajímavou cestou k likvidaci oběti je zahlcení jeho DNS systému tak, aby přestal reagovat a přestalo tak fungovat překládání názvů na IP adresy, čímž přestane být oběť schopna komunikovat. Případně takto můžeme odstavit doménu tím, že zahltíme její autoritativní DNS servery. Často je to mnohem jednodušší než útočit přímo na oběť.

3.3 DNSSEC

Lidstvo si bezpečnostní aspekty DNS a jejich možné dopady již dlouze uvědomovalo. Nabízí se několik možných řešení, například symetrické šifrování DNS paketů, tedy zabezpečení přenosových cest a znemožnění pozměnění přenášených dat – zde je však hlavní komplikací klasický problém symetrického šifrování – distribuce klíče oběma stranám komunikace, aniž by mohl být klíč podvržen. Navíc by to znamenalo provádět šifrování a dešifrování při každém DNS dotazu, což je z hlediska požadovaného výkonu nereálné zvláště pro velmi vytížené DNS servery (např. kořenové servery). Problémem by také bylo zajistit to, aby si pravdivost získaných DNS záznamů mohl jejich příjemce (aplikace na počítači koncového klienta) ověřit přímo od producenta (administrátor příslušné zóny). Se symetrickým šifrováním by fungovalo jen ověření mezi uzly, které si přímo DNS informace vyměňují (hop-by-hop) a stále by nebylo problém pozměnit nějaká data na mezilehlém DNS uzlu.

Naproti tomu DNSSEC k celé věci přistupuje odlišně – data se přenáší stále stejným nezabezpečeným způsobem a využívá se asymetrické kryptografie pro vytváření elektronických podpisů DNS záznamů tak, aby si mohl jejich pravdivost ověřit přímo jejich konzument. Tyto podpisy a související data jsou ukládána opět ve formě DNS

záznamů. DNSSEC zavádí pro tyto potřeby několik nových druhů záznamů. Některé obsahují elektronický podpis stávajících záznamů, jiné nesou informace pro ověření neexistence záznamů, které v zóně domény nejsou.

DNSSEC je definováno v několik dokumentech z roku 2005. RFC 4033 (DNS Security Introduction and Requirements)[21] teoreticky popisuje základní principy a pojmy, RFC 4034 (Resource Records for the DNS Security Extensions)[22] zavádí nový typy DNS záznamů a jejich význam a nakonec RFC 4035 (Protocol Modifications for the DNS Security Extensions)[23] popisuje další změny DNS související se zavedením DNSSEC. První RFC o DNSSEC sice vyšlo již v roce 1999, ale tato prvotní definice byla v praxi shledána jako nevyhovující, proto byla v podstatě od základů předělána.

DNSSEC je rozšíření stávajícího DNS systému, nikoliv jeho náhrada. Vše tedy funguje stejně jako dříve, jen vzniklo navíc několik nových druhů záznamů. Záleží na klientovi, zda novým druhům záznamům rozumí, zda je využije a zda provede ověření či nikoliv. Klient, který DNSSEC nezná, akceptuje požadované záznamy, nové typy záznamů ignoruje a neprovádí žádnou kontrolu. Modernější klient však použije nové záznamy k ověření těch ostatních a data získaná ze serveru akceptuje pouze v případě, že všechny elektronické podpisy sedí. Z toho vyplývá, že nestačí, aby DNSSEC ovládaly DNS servery, ale musí se aktivně účastnit i klienti. Tedy aby byl celý DNS systém naprosto odolný proti útokům, kterým DNSSEC umí zabránit, musí DNSSEC rozumět a používat všechny DNS servery i klienti. Cachovací DNS servery samozřejmě také musí uchovávat a předávat dál i příslušné DNSSEC záznamy.

Ještě je potřeba upřesnit, co se zde myslí pojmem „klient“. Kontrola získávaných záznamů z DNS může probíhat na několika úrovních. Kontrolu záznamů a jejich podpisů může provádět přímo koncový počítač. Může to však dělat také cachovací DNS server organizace, kterému počítače uvnitř firemní sítě důvěřují, přebírají od něj DNS záznamy a již neprovádí jejich kontrolu, díky tomu nejsou zatíženy ověřováním podpisů. Podobně to může dělat cachovací DNS server poskytovatele připojení, který na počítače svých zákazníků nepropustí nedůvěryhodná data.

3.3.1 Záznam typu RRSIG

Tento záznam se do zóny přidává pro každou množinu záznamů stejného názvu a typu a obsahuje jejich elektronický podpis. Pomocí něj lze tedy ověřit, zda nebyly tyto získané záznamy pozměněny, a navíc také zda nějaký záznam z této množiny nechybí či nepřebývá. Není potřeba přidávat RRSIG pro každý jednotlivý záznam, protože jsou stejně vždy DNS serverem vraceny všechny daného jména a typu.

V praxi tedy ke každé množině záznamů stejného jména a typu přibude jeden RRSIG záznam. Týká se to úplně všech záznamů, včetně SOA a NS. To tedy znamená až dvojnásobně velký počet záznamů v zóně oproti stavu před zavedením DNSSEC. Tím to však nekončí, jak bude uvedeno u záznamu NSEC.

Záznam RRSIG obsahuje tyto informace:

- typ podepisovaného záznamu
- číselné označení použitého algoritmu pro elektronický podpis (např. RSA/SHA1)
- číslo úrovně podepisovaného záznamu (kořenová zóna se nezapočítává), tedy např. jméno www.mff.cuni.cz je úrovně 4
- údaj TTL podepisovaného záznamu (všechny záznamy stejného jména a typu musí mít TTL stejné, proto stačí jeden údaj pro celou množinu)
- doba konce platnosti podpisu, uvádí se ve formátu YYYYMMDDhhmmss
- doba počátku platnosti popisu (stejný formát)
- číselné označení klíče, kterým bylo podepsáno
- doménové jméno podepisujícího
- vlastní podpis v Base64 kódování

3.3.2 Záznam typu DNSKEY a veřejné klíče

Podpis v záznamu typu RRSIG je vytvořen pomocí privátního klíče, který drží majitel příslušné domény u sebe v tajnosti mimo systém DNS, a lze ověřit veřejným klíčem, který se nachází ve stejné zóně v záznamu typu DNSKEY. Záznam DNSKEY musí být, stejně jako všechny ostatní záznamy, podepsán svým privátním klíčem (podepisuje sám sebe) v odpovídajícím záznamu RRSIG.

V praxi je doporučeno v doméně používat 2 veřejné klíče (tedy vloží se 2 záznamy typu DNSKEY):

- ZSK (zone signing key) – slouží pro podepisování ostatních záznamů domény – měl by být „jednodušší“, protože je intenzivně používán pro podepisování a zejména pro ověřování a také může být často měněn, typicky má tedy kratší platnost
- KSK (key signing key) – slouží pro podepisování klíčů (záznamů typu DNSKEY, tedy sebe sama), mění se méně často (protože změna nelze provést jen tak, musí se vyjednat i v nadřazené zóně), typicky má delší platnost

Záznam DNSKEY obsahuje:

- příznaky (zatím definován jen jeden s hodnotou 256, zbytek rezervován pro budoucí použití)
- protokol (opět zatím definován jen jeden s hodnotou 3)
- označení algoritmu klíče (ZSK používá „složitější“, KSK „jednodušší“)
- hodnota klíče v Base64 kódování
- číselné označení klíče

Na každém místě, kde je klíč definován nebo použit, se vyskytuje jeho číselné označení. Toto číslo (unikátní v rámci domény) slouží pro jeho snazší identifikaci pro potřebu klientů a je zde zejména proto, že zóna může teoreticky obsahovat libovolné množství ZSK klíčů (každý může podepisovat některou skupinu záznamů) a je potřeba mezi nimi rozlišovat. Podle tohoto ID také klienti poznají, že se záznamy začaly podepisovat jiným klíčem.

Ve výsledku má doména 2 záznamy typu DNSKEY (pro ZSK a KSK) a k nim odpovídající záznam RRSIG (podpis privátním klíčem KSK). Toto rozdělení však není povinné, doména může pro oba účely použít jeden klíč (jen jeden záznam DNSKEY a všechny RRSIG s použitím jednoho privátního klíče).

3.3.3 Záznam typu DS a řetězec důvěry

Je samozřejmě nutné, aby klient měl možnost si veřejný klíč KSK ze záznamu DNSKEY příslušné domény ověřit. Pro tyto účely se do nadřazené zóny vkládá pro

danou doménu záznam typu DS (Delegation Signer), který obsahuje otisk onoho veřejného klíče, a ten je podepsán privátním klíčem (ZSK) této nadřazené zóny. To vše samozřejmě pouze v případě, že všechny domény na cestě ve stromu podporují DNSSEC.

DS záznam obsahuje tyto údaje:

- číselné označení klíče (pro jeho snadné nalezení)
- označení algoritmu, pro který se klíč používá
- typ digestu (otisku)
- digest (otisk veřejného klíče)

Celý proces ověření by se tedy dal popsat takto: požadovanému záznamu lze důvěřovat, pokud je podepsán ZSK klíčem dané domény (v příslušném RRSIG). ZSK klíči lze důvěřovat, jestliže je podepsán KSK klíčem stejné domény. Pravost KSK klíče si ověříme v DS záznamu nadřazené domény, tento DS je podepsán ZSK klíčem oné nadřazené domény. Tím jsme přešli na úroveň výše. Ověřování se zastaví v okamžiku, kdy narazíme na veřejný klíč, kterému klient důvěřuje. To je buď veřejný klíč nějaké domény, kterému důvěřuje na základě své konfigurace. V nejzazším případě se dostaneme až do kořenové zóny, její klíč by měl být všeobecně známý a důvěryhodný.

Aby se nemusela při každém DNS dotazu provádět takto komplikovaná kontrola a stahování a ověřování klíčů a podpisů ze všech domén až ke kořenové zóně, klienti a cachovací DNS servery si samozřejmě získané DNSSEC záznamy pamatují a navíc již ověřené certifikáty až do konce jejich platnosti znovu neověřují.

3.3.4 Záznam typu NSEC, negativní odpovědi

Tento záznam řeší trochu jinou situaci než záznamy RRSIG, které potvrzují existenci nějaké množiny záznamů a jejich obsah. Neméně důležité je mít možnost ověřit, že nějaké doménové jméno či záznam vůbec neexistuje. K tomu slouží záznam typu NSEC, ten se přidává do zóny ke každé množině záznamů, které mají stejný název (ale už se nerozlišují typy záznamů), a obsahuje dvě informace:

- seznam typů záznamů, které zóna obsahuje pro toto jméno
- následující existující jméno ve stejné doméně

Poslední jméno v doméně odkazuje na první (uspořádání jmen přesně definuje příslušné RFC). Tím propojuje všechna jména domény do řetězce. Samozřejmě i tento záznam má svůj RRSIG, který jej podepisuje.

Pomocí tohoto záznamu si tedy můžeme ověřit 2 situace.

1. Dané jméno v doméně existuje, ale nikoliv tento typ záznamu – DNS server klientovi zašle prázdnou odpověď a do Additional Section vloží NSEC záznam pro toto jméno, ve kterém si neexistenci daného typu záznamu ověříme. Samozřejmě obdržíme i RRSIG záznam, abychom ověřili důvěryhodnost NSEC záznamu.
2. Dané jméno v doméně neexistuje – DNS server nám zašle NSEC záznam posledního existujícího jména, které abecedně předchází to ověřované, a tento záznam obsahuje jméno následujícího existujícího jména. To je důkaz, že mezi těmito existujícími jmény žádné další není.

3.3.5 Zavedení do praxe

O zavedení DNSSEC do praxe se stará pracovní skupina Deployment of Internet Security Extensions (DISI) organizace RIPE. Poskytuje dokumenty a návody k nastavení DNSSEC na konkrétních implementacích DNS serverů a klientů. DNSSEC je podporován v nejnovějších verzích softwaru BIND, NSD. Zajímavým projektem jsou také stránky www.dnssec.net, které slouží široké veřejnosti jako propagace DNSSEC, obsahují obsáhlé informace a odkazy na všechny související dokumenty, přednášky, výzkumy aj.

Problémem je také podpis kořenové zóny. Kdo ji má podepsat a kdo má držet její privátní klíč? V současné době to není ještě zcela vyřešeno, kořenová zóna je prozatím podepsána jen zkušebně na DNS serveru ns.iana.org.

Jak bylo již řečeno, nasazení DNSSEC do praxe bude dlouhé a komplikované proto, že jej musí podporovat všechny DNS servery a klienti, tedy musí jej implementovat veškerý DNS software a musí jej nakonfigurovat všichni administrátoři. To znamená, že úplné nasazení DNSSEC může trvat mnoho let až desítky let. Avšak ne všichni se staví

k DNSSEC s nadšením a pokládají si otázku, zda je bezpečnost DNS opravdu tak kritická, zda dochází k opravdu vážným incidentům způsobeným útoky na DNS, zda za to všechno nárůst objemu dat a náročnosti zpracování DNS dotazů stojí a zda by se nedalo vymyslet lepší řešení než DNSSEC.

3.3.6 Shrnutí

DNSSEC samozřejmě přináší mnoho nového do DNS a otázek jeho bezpečnosti. Klient si může na základě důvěry ke kořenové zóně a jejímu certifikátu postupně ověřit samotnou zónu a její nadřazené. To však přináší nějaké oběti, zejména v podobě až čtyřnásobného nárůstu počtu záznamů v zóně. Avšak jelikož nové záznamy jsou mnohem více objemné, velikostně zóna naroste ještě více (cca desetkrát). Zvyšuje se tedy objem dat na DNS serverech, ale také objem dat přenášených po síti.

Nezanedbatelným faktem je také nárůst nároků na výpočetní výkon klientských zařízení, která musí provádět ověřování elektronických podpisů asymetrickou kryptografií s použitím veřejných klíčů. Také dochází k dalším dodatečným DNS dotazům v případě, že klient příslušnému certifikátu zóny nedůvěřuje a chce si ho ověřit v nadřazené zóně. Tak se může postupně dotazovat až do kořenové zóny, aby se celý řetězec důvěry uzavřel a klient prohlásil data za pravá.

Měli bychom se též zamyslet nad otázkou, proč se např. podepisují jen jednotlivé záznamy a proč se nevytvoří jeden elektronický podpis pro všechny záznamy domény. Důvodů je zřejmě několik – aby mohl klient ověřit pravost získaných dat, musel by mít k dispozici obsah celé zóny domény, aby mohl provést kontrolu podpisu. To samozřejmě není možné, ať už z důvodu objemu dat či nutnosti prozrazení všech záznamů ze zóny libovolnému klientovi. Navíc při jakékoliv malé změně by se musel přepočítávat podpis celé zóny, takto stačí přepočítat podpis jednoho či několik málo záznamů. Podpisy se samozřejmě nesestavují v okamžiku dotazu klienta, ale jen při změně v zóně na DNS serveru a pak jsou uloženy s ostatními záznamy a jsou jen pasivně čteny.

Je také podstatné si přesně uvědomit, co DNSSEC umí a poskytuje, ale co naopak neumí a co nám nezajistí, abychom nezískali pocit, že řeší všechny bezpečnostní problémy.

DNSSEC řeší:

- ověření správnosti získaných DNS záznamů
- získání důkazu o neexistenci nějakého doménového jména nebo nějakého jejího záznamu

DNSSEC neřeší:

- útoky na DNS systém klienta, tedy podvržení údajů na klientském počítači (aplikace nad operačním systémem s DNSSEC již nepracují)
- zabezpečení přenosových cest protokolu DNS – vše nadále putuje otevřeně a veřejně, DNS dotazy se mohou odposlouchávat (a lze takto zjišťovat, co kterého klienta zajímá a kde „surfuje“)
- omezení přístupu k záznamům domény, autentizaci klientů
- útoky na DNS servery za účelem jejich odstavení, vhodný DDoS útok příslušný autoritativní DNS server odřízne od světa a k záznamům domény se v tu dobu nikdo nedostane
- podvodné přesměrování či podobný útok na úrovni IP adres – z DNS systému se sice bezpečně dozvíte správnou IP adresu, na kterou se potřebujete spojit, ale DNSSEC nezabrání tomu, aby vás při následné komunikaci s cílem “man-in-the-middle“ nenasměroval jinam

Kapitola 4 – Rozšíření analýzy

V této kapitole si uvedeme, které další informace o zkoumaných doménách a souvisejících službách si budeme během analýzy zjišťovat a jak budeme ověřovat jejich případnou správnost. V bakalářské práci byly stanoveny 3 úrovně zjištěných problémů – chyba, varování a upozornění. V nové rozšířené analýze se však budou vyskytovat též testy, jejichž výsledkem nemusí být zjištění nějakého problému, ale rozlišení různých situací (např. podporovaných služeb) či zajímavých informací, které jsou všechny korektní a nikdy nekončí chybou. Proto zavedeme další typ výsledku: informace. Bude se jednat o příznaky, které poté využijeme při sestavování celkových výsledků analýzy celé sady domén. Celkově tedy máme následující možné výsledky každého testu:

- **V pořádku** – v testu byla zkoumána správnost nějakého údaje či stavu a vše je v pořádku – tj. funkční či přesně v souladu s nějakým standardem či doporučením.
- **Chyba** – nesplnění zkoumané věci způsobuje úplnou nebo částečnou nefunkčnost či omezení provozu dané služby, která tak neplní správně svůj účel. Systém nefunguje buď vůbec nebo funguje pouze část systému, případně má systém zásadní bezpečnostní problémy, nedostatky ve svém nastavení anebo zcela odporuje nějakému standardu či doporučení.
- **Varování** – nesplnění nezpůsobuje bezprostřední nefunkčnost systému, avšak jedná se o jeho nesprávný stav, který může v nefunkčnost či jiný problém vyústit, a tak je nutno tomuto varování věnovat pozornost.
- **Upozornění** – sledovaný parametr je po funkční stránce v pořádku, avšak není nastaven tak, jak je všeobecně doporučené. Může se jednat o chybu v nastavení (překlep administrátora, neznalost problematiky) anebo to může být také speciální účelové nastavení.
- **Informace** – nejedná se o testování správnosti nějakého údaje, pouze se nějaké zajímavé informace zjišťují a analyzují. Výsledek je čistě informativní.

Každý test může vyprodukovat více výsledků, pokud se vzájemně nevylučují. Celkový výsledek testu se odvíjí od nejzávažnějšího ze zjištěných problémů.

4.1 Dostupnost a informace o WWW službách

Zřejmě nejpoužívanější službou na Internetu je World Wide Web, tedy internetové (webové) stránky, které se staly samozřejmou součástí našeho života. Denně je používáme k hledání informací pro naši práci, studium či zábavu. Firma, která dnes nemá svou WWW prezentaci, jako by neexistovala. Místo do kamenných obchodů chodíme nakupovat do e-shopů.

První rozšíření aplikace pro analýzu domén se tedy zaměřuje na WWW stránky a technické aspekty jejich provozu. Bude nás zajímat, na jakých serverech jsou stránky provozovány, jaké stránky jsou mimo provoz, jaké stránky nejsou korektně nastavené apod. K tomuto zkoumání budeme samozřejmě používat HTTP protokol, kterým se budeme spojovat s WWW servery a budeme z nich získávat data a zkoumat je.

4.1.1 Kontaktování WWW serveru

V tomto testu provedeme pokus o TCP spojení na portu 80 na IP adresu z A záznamu, uvedeného u subdomény „www“ dané domény, což je neobvyklejší adresa WWW stránek. Pokud doména nemá A záznam pro subdoménu „www“, použijeme A záznam pro samotnou doménu (bez subdomény). Jestliže takových A záznamů existuje více, vezme se některý z nich a ostatní se ignorují. Jestliže naopak doména nemá ani jeden z těchto dvou A záznamů, tento test i všechny následující testy WWW přeskočíme s konstatováním, že doména nenabízí WWW služby na běžných adresách. To samozřejmě nemusí znamenat, že nenabízí WWW služby vůbec či že nenabízí žádné služby – mohou běžet na nějakých speciálních subdoménách a/nebo TCP portech.

Pokud se nám spojení zdaří, odešleme následující požadavek:

```
HEAD / HTTP/1.1
Host: <doména s www nebo bez www>
```

Následně vyčkáme na odpověď a tím ověříme, že skutečně komunikujeme s WWW serverem přes protokol HTTP. Posíláme příkaz HEAD, zajímají nás tedy pouze hlavičky odpovědi a tělo cílové stránky stahovat vůbec nebudeme. Timeouty pro navázání TCP spojení a přijetí odpovědi jsou 5 sekund. Pokud obdržíme zpět odpověď s

kódem 3xx, následujeme přesměrování. Pro případ zacyklení povolíme max. 3 přesměrování. Budou nás však zajímat pouze přesměrování, která jsou v rámci stejné domény. Pokud obdržíme přesměrování na jinou doménu 2. úrovně, poznamenáme si ji a testy WWW ukončíme. Může se také stát, že nás WWW server přesměruje na zabezpečení připojení, takové také nebudeme následovat.

Pro potřeby dalších testů si uložíme všechny získané hlavičky z odpovědi.

Správný výsledek		Spojení s WWW přes HTTP protokol je v pořádku
101-1	informace	Žádný A záznam s www či bez www
101-2	chyba	Nepodařilo se navázat spojení s WWW serverem
101-3	chyba	Chybná odpověď WWW serveru, není protokolem HTTP
101-4	varování	Příliš mnoho přesměrování při komunikaci s WWW serverem
101-5	varování	Timeout při čekání na odpověď od serveru
101-6	informace	Přesměrování na jinou doménu při komunikaci s WWW serverem
101-7	informace	Přesměrování na HTTPS při komunikaci s WWW serverem
101-8	informace	A záznam má neveřejnou IP adresu

Jestliže doména žádný příslušný A záznam nemá anebo tento test skončil chybou, další testy, týkající se HTTP, se přeskočí, protože se nám nepodařilo žádné další informace zjistit. Chyba navázání spojení může být způsobena tím, že v A záznamu je chybná IP adresa (potom cílový WWW server buď vůbec neexistuje nebo nenabízí WWW služby), cílový server nabízí WWW služby na jiném portu, cílová síť je nedostupná, server je vypnutý nebo porouchaný. Na rozdíl od nedostupnosti DNS serverů či mailserverů se však nejedná až o tak závažný problém – nedostupnost WWW znamená většinou jen to, že se nyní lidé na stránky podívat nemohou a podívají se na ně později, avšak to neznamená nefunkčnost dalších systémů či ztrácení nějakých informací. Horší by bylo, kdyby WWW server sloužil jako brána pro Web Services, následkem čehož by i jiné systémy, na tomto závislé, přestaly fungovat.

Další speciální situace nastane, jestliže IP adresa v A záznamu patří mezi neveřejné IP adresy, které nejsou určené pro použití v Internetu tak jak jsou vyjmenovány v dokumentu RFC 3330[10]. To zřejmě znamená, že WWW služby dané domény jsou určeny jen pro nějaký Intranet na lokální síti majitele domény a nejsou dostupné ze zbytku Internetu. V takovém případě ani nemá smysl se o spojení na WWW server

pokoušet, všechny HTTP testy budou přeskočeny. Může to také samozřejmě znamenat chybu, kdy je neveřejná IP adresa v A záznamu omylem a provozovatel domény si myslí, že všechno funguje, protože se na web server dívá ze stejné lokální sítě, ale tuto situaci my rozlišit nedokážeme.

Chyba 101-3, tedy chyba v protokolu HTTP, znamená, že se serverem sice bylo navázáno spojení a byl mu odeslán požadavek, ale formát odpovědi ze serveru nebyl rozpoznán. To zřejmě znamená, že na daném portu sice běží nějaká služba, ale nejedná se o WWW server komunikující na HTTP protokolu. To však budeme považovat za chybu, protože je to skutečně neobvyklé, aby na portu 80 bylo něco jiného než WWW server.

Podobně je problém situace, kdy dojde k příliš mnoho přesměrování. Jak bylo řečeno, akceptujeme max. 3 přesměrování, a pokud se ani poté nedostaneme k výsledné stránce, budeme to považovat za upozornění, protože velký počet přesměrování může způsobit problémy některým starším prohlížečům (mohly by se rozhodnout tolik přesměrování nenásledovat a uživatel by se na stránku nedostal) a také vyhledávačům, který by se to nemuselo líbit. Smysl má pouze jedno přesměrování, které přesune uživatele přímo do cíle. Avšak důvodem této chyby může být i to, že přesměrování je natolik špatně nastavené, že je v něm cyklus a nikdy bychom se k cílové stránce vůbec nedostali.

4.1.2 Kód odpovědi HTTP

Pokud se test spojení s WWW serverem zdařil, zaevidujeme kód HTTP odpovědi. Pokud byla odpověď negativní, zobrazíme uživateli (v případě testu jedné domény) vysvětlivky, jaká odpověď byla zjištěna a co přesně znamená.

Správný výsledek		Odpověď 200, přesměrování na jinou doménu či na SSL
102-1	informace	Přístup zamítnut (403 Forbidden)
102-2	chyba	Chyba na straně serveru (odpověď 5xx)
102-3	informace	Vyžadována autorizace (odpověď 401)
102-4	upozornění	Stránka nenalezena (odpověď 404)
102-5	varování	Jiná chyba (jiný kód 4xx nebo neznámý kód)

4.1.3 Zjištění softwaru serveru

Pokud se test spojení s WWW serverem zdařil, provedeme analýzu hlaviček v odpovědi. Často se v odpovědi vyskytuje hlavička **Server**. V té bývá uveden software WWW serveru a jeho verze, popř. jeho další rozšiřující moduly. V případě Apache serveru můžeme vidět např. následující:

```
Apache/1.3.41 mod_ssl/2.8.31 OpenSSL/0.9.7a CSacek/2.1.9 PHP/4.4.8
```

Z toho se dozvíme dokonce i operační systém serveru a verzi skriptovacího jazyka PHP. Hlavička X-Powered-By je někdy vkládána do odpovědi skriptovacími jazyky:

```
X-Powered-By: PHP/4.4.8
```

Abychom mohli všechen zjištěný software evidovat, budeme rozdělovat získané hodnoty z hlaviček Server a X-Powered-By na jednotlivé části podle mezer, vyloučíme duplicity, získané řetězce dále rozdělíme podle lomítek, abychom oddělili název softwaru od jeho verze, a vše si uložíme k výsledkům testu domény. Je nutné zdůraznit, že informace z hlaviček nemusí být pravdivá, je to pouze tvrzení daného WWW serveru, který může z nějakého důvodu úmyslně zasílat nesprávné informace. Navíc mnoho serverů informace o svém softwaru neposkytují vůbec, nebo alespoň skrývají čísla verzí, protože tato znalost by mohla potenciálnímu útočníkovi napomoci k použití známých bezpečnostních děr.

Tento test neposkytuje žádné chybové návratové kódy, jeho výsledkem je pouze množina software a verzí, které byly zjištěny z hlaviček, popř. informace, že se žádný software zjistit nepodařilo.

103-1	informace	Software WWW serveru nebyl zjištěn
103-2	informace	Byl zjištěn software z hlaviček odpovědi WWW serveru

4.1.4 Přítomnost AAAA záznamů

Jak známo, zásoby IPv4 adres se ztenčují a Internet se bez přechodu na IPv6 brzy neobejde. Přestože je protokol IPv6 připraven již dlouho, masivně se na něj zatím

nepřechází, protože to samozřejmě ISP společnostem přináší nemalé výdaje a technické komplikace.

Tento test má za úkol zjistit, zda se u domény nachází AAAA záznamy (pro subdoménu www nebo alespoň pro samotnou doménu bez subdomény www) a zda jsou tedy její WWW stránky potenciálně dostupné přes IPv6. Z hlediska WWW to samozřejmě záleží hlavně na podpoře IPv6 u webhostingových a serverhousingových společností, v ČR je však podpora IPv6 v tomto směru mizerná[2].

104-1	informace	Doména obsahuje A záznamy a také AAAA záznamy
104-2	informace	Doména obsahuje A záznamy, neobsahuje AAAA záznamy
104-3	informace	Doména neobsahuje A záznamy, ale obsahuje AAAA záznamy
104-4	informace	Doména neobsahuje A ani AAAA záznamy

V tomto testu nás velmi zajímají kombinace A a AAAA záznamů. Obvyklá situace je, že jsou k dispozici A záznamy a nikoliv AAAA záznamy. Velmi neobvyklé však může být, pokud jsou WWW stránky domény dostupné jen přes IPv6 a nikoliv přes IPv4 (tedy má jen AAAA záznamy a nikoliv A záznamy).

4.1.5 TTL hodnoty u A záznamů

Stejně jako v bakalářské práci, kde jsme kontrolovali shodu TTL u NS záznamů, zde budeme hlídat shodu TTL hodnot u všech typů záznamů, které budeme zkoumat. Připomínám, že standard DNS stanovuje, že obecně jakékoliv záznamy, které se shodují názvem domény a typem, musí mít stejné hodnoty TTL. V opačném případě by měly být tyto záznamy považovány za neplatné. Tento test má smysl samozřejmě jen v případě, že u domény je více než jeden A záznam. Zkoumají se obě sady A záznamů, tedy pro subdoménu „www“ a pro samotnou doménu bez subdomény.

Správný výsledek		TTL hodnoty u A záznamů se shodují
105-1	chyba	Neshoda TTL u A záznamů bez „www“
105-2	chyba	Neshoda TTL u A záznamů s „www“

- Reference: RFC 2181[11] (5.2)

4.1.6 TTL hodnoty u AAAA záznamů

Stejný test jako předchozí, tentokrát zkoumáme TTL u AAAA záznamů s „www“ i bez „www“.

Správný výsledek		TTL hodnoty u AAAA záznamů se shodují
106-1	chyba	Neshoda TTL u AAAA záznamů bez „www“
106-2	chyba	Neshoda TTL u AAAA záznamů s „www“

4.1.7 Reverzní záznamy WWW serveru

Podobně jako jsme u DNS serverů zjišťovali, zda nalezením reverzních záznamů jejich IP adres a následným převodem reverzních záznamů získáme opět stejné IP adresy, budete toto ověřovat i u A záznamů pro WWW. Opět si také všímáme situace, kdy nějaká IP adresa žádný reverzní záznam nemá anebo zjištěný reverzní záznam nelze přeložit na žádnou IP adresu (vyhodnoceno jako neshoda). Zkoumají se dohromady záznamy s „www“ i bez „www“, tento test nebyl v této práci implementován pro AAAA záznamy.

Je sice slušnost mít v pořádku všechny reverzní záznamy, avšak u A a AAAA záznamů pro WWW služby se to nepovažuje za velký problém, protože se nekontrolují (na rozdíl od reverzních záznamů pro IP adresy MX záznamů, jak je uvedeno dále).

Správný výsledek		Reverzní záznamy se shodují s jejich IP adresami
107-1	upozornění	Reverzní záznamy se neshodují u všech A záznamů
107-2	upozornění	Reverzní záznamy se neshodují u některých A záznamů (nikoliv u všech)
107-3	upozornění	Reverzní záznamy u některých A záznamů neexistují

4.2 Dostupnost a informace o e-mailových službách

Druhou nejdůležitější službou Internetu je zřejmě elektronická pošta. Další rozšířením analýzy, které následuje po WWW, je tedy logicky zkoumání dostupnosti e-mailových služeb u domény. Bude nás zajímat počet a korektnost MX záznamů, dostupnost primárních i sekundárních mailserverů, služby nabízené mailservery, použití

bezpečnostních prvků pro ochranu před spamem aj. Samozřejmě budeme využívat protokolu SMTP pro komunikaci s těmito servery.

4.2.1 Počet MX záznamů

Jestliže doména nemá žádný MX záznam, nemusí to znamenat chybu. Doména prostě není určena pro to, aby byly na její adresy doručovány zprávy. Jestliže se někdo pokusí zaslat zprávu na takovou doménu, SMTP servery při nenalezení MX záznamu zkusí zprávu doručit na IP adresu, která je uvedena v A záznamech domény. To je však zavržené řešení, jelikož to např. znamená, že WWW i e-mailové služby jsou provozovány na stejném serveru. Pokud tento zastaralý způsob doména používá, to nedokážeme bohužel správně rozlišit. Nepřítomnost MX záznamů tedy skončí pouze informativní hláškou s příslušným konstatováním a další testy, které souvisí s MX záznamy, prováděny nebudou.

Pokud má doména jeden MX záznam, již zřejmě nabízí doručování zpráv, ale problém může nastat při nedostupnosti tohoto jednoho serveru. V takovém případě odesílající SMTP servery zprávu uschovají a budou se nějakou dobu opakovaně pokoušet o doručení, jak to vyžaduje RFC 5321[4]. Avšak pokud SMTP server odesílatele RFC standardy nedodržuje a o další doručení se nepokusí, zpráva se nenávratně ztratí. Proto by doména měla mít alespoň jeden další MX záznam se záložním mailserverem.

Jeden MX záznam může také samozřejmě znamenat, že se za ním schovává více serverů, mezi něž se rozkládá zátěž a které se zastupují v případě poruchy. To bohužel nerozpoznáme. V každém případě je mnohem lepší, použijeme-li více různých MX záznamů, které budou odkazovat na různé na sobě zcela nezávislé servery či skupiny serverů.

201-1	informace	Doména neobsahuje žádné MX záznamy
201-2	informace	Doména obsahuje alespoň 2 MX záznamy
201-3	upozornění	Doména obsahuje pouze 1 MX záznam

V této souvislosti by také bylo dobré prozkoumat autonomní systémy a podsítě, v nichž se mailservery domény nachází, podobně jak se to zkoumá u DNS serverů, a doporučit

použití alespoň 2 podsítí, či ještě lépe alespoň dvou autonomních systémů. Výrazně to zvýší dostupnost e-mailových služeb při nějakých poruchách. To však nebylo v této práci realizováno.

4.2.2 Kontrola syntaxe MX záznamů

Chybou bude, když MX záznam bude mít nesprávnou syntaxi (musí obsahovat číselné vyjádření priority, mezeru a následuje doménový název). Pokud takový problém nalezneme, je to fatální chyba a nebudeme v dalších testech MX záznamů pokračovat. Je otázka, jestli vůbec DNS servery umožní zapsání chybného MX záznamu do domény a pak je distribuují. I s tím však musíme počítat, ať už je to chyba implementace DNS serveru anebo chyba administrátora.

Chybou bude také např. to, pokud nějaký záznam je sice syntakticky správně, ale namísto doménového názvu mailserveru obsahuje přímo jeho IP adresu. V takovém případě nahlásíme chybu, ale budeme pokračovat v dalších testech.

Správný výsledek		Všechny MX záznamy mají správnou syntaxi
202-1	chyba	Některý MX záznam má chybnou syntaxi
202-2	chyba	Některý MX záznam obsahuje IP adresu

- reference: RFC 974[24]

4.2.3 Překlad MX záznamů na IP adresy

Pokusíme se všechny doménové názvy z MX záznamů přeložit na IP adresy.

Správný výsledek		Všechny názvy z MX záznamů lze přeložit na IP adresu
203-1	chyba	Některé názvy z MX záznamů nelze přeložit na IP adresu
203-2	chyba	Žádný název z MX záznamů nelze přeložit na IP adresu

Pokud některý název na IP přeložit nelze, samozřejmě není možné se s daným serverem spojit. V takovém případě by měl klient zkusit další MX záznam. Nelze-li přeložit žádný název, nelze doručovat e-maily pro doménu vůbec.

4.2.4 Duplicita MX záznamů

Zkontrolujeme, zda nejsou názvy duplicitní či zda nejsou po překladu na IP adresy duplicitní. Není to sice chyba, ale v takové situaci je zbytečné mít MX záznamů více, když ve skutečnosti směrují na stejný server.

Správný výsledek		MX záznamy nejsou duplicitní
204-1	chyba	MX záznamy obsahují duplicitní doménový název
204-2	upozornění	Některé MX záznamy směřují na stejnou IP adresu

4.2.5 Kontrola TTL MX záznamů

Opět kontroly shody hodnot TTL u všech MX záznamů domény.

Správný výsledek		TTL hodnoty u MX záznamů se shodují
205-1	chyba	Neshoda TTL u MX záznamů

4.2.6 Reverzní záznamy MX záznamů

U MX záznamů jsou oproti A záznamům reverzní záznamy již mnohem důležitější. Jsou totiž bedlivě kontrolovány většinou mailserverů, a pokud nejsou reverzní záznamy správné (tj. reverzním překladem IP adresy z názvu MX záznamu a následným překladem zpět na IP se nedostaneme na původní IP), e-mailové zprávy bývají dokonce odmítány. Případné zjištěné problémy tedy budeme považovat za chybu, nikoliv pouze jen za varování.

Správný výsledek		Reverzní záznamy se shodují s jejich IP adresami
206-1	chyba	Reverzní záznamy se neshodují u všech MX záznamů
206-2	chyba	Reverzní záznamy se neshodují u některých MX záznamů (nikoliv u všech)
206-3	chyba	Reverzní záznamy u některých MX záznamů neexistují

4.2.7 Kontaktování primárního e-mailového serveru

Pokusíme se navázat spojení s primárním mailserverem, tedy s tím, který má v MX záznamech nejvyšší prioritu (nejmenší číslo priority). Samozřejmě že nelze s jistotou

tvrdit, že první server v seznamu je primární. Za primární považujeme ten, na kterém e-mailové zprávy pro danou doménu končí a odkud si je pak příjemce vyzvedává. Priority MX záznamů ale určují jen to, na které (resp. přes které) servery se mají zprávy doručovat. Primární server v MX záznamech ani být nemusí, servery v MX mohou sloužit pouze jako brány a cílový server může být umístěn např. uvnitř lokální sítě a není dostupný přes veřejnou IP adresu. V každém případě je nejdůležitější správná funkčnosti právě toho serveru z MX záznamů s nejvyšší prioritou, protože jeho kontaktují všichni klienti bez ohledu na to, co se se zprávou děje dál a kde nakonec doopravdy skončí.

Správný výsledek		Spojení se primárním mailserverem je v pořádku a akceptuje e-maily pro tuto doménu
207-1	informace	Žádný platný MX záznam
207-2	chyba	MX záznam nelze přeložit na IP adresu
207-3	upozornění	MX záznam má neveřejnou IP adresu
207-4	chyba	Nepodařilo se navázat spojení se serverem
207-5	chyba	Chybná odpověď serveru, není protokolem SMTP
207-6	varování	Timeout při čekání na odpověď od serveru
207-7	chyba	Ztráta spojení se serverem během komunikace
207-8	informace	Server s námi odmítl komunikovat či odmítl našeho odesílatele, test nemohl být dokončen
207-9	chyba	Server neakceptuje e-maily pro postmaster@domena

Podobně jako u A záznamů, chyba navázání spojení může být způsobena tím, že v MX záznamu je chybný doménový název SMTP serveru anebo je chybná jeho IP adresa (potom cílový SMTP server buď vůbec neexistuje nebo nenabízí e-mailové služby), server je vypnutý nebo porouchaný. Je zcela nepravděpodobné, že by SMTP server nabízel služby na jiném portu než 25. V e-mailová adresa totiž nelze, na rozdíl od WWW adresy, uvést jiný port pro komunikaci se serverem.

Opět zkoumáme také situaci, kdy IP adresa SMTP serveru patří mezi neveřejné IP adresy. To zřejmě znamená, že SMTP služby dané domény jsou určeny jen pro nějaký Intranet na lokální síti majitele domény a nejsou dostupné ze zbytku Internetu. Pokud tato situace nastane, nemá smysl se pokoušet SMTP server kontaktovat. Dále podobně

zkoumáme chybu v protokolu SMTP. Při jakémkoliv zmíněném problému budou další testy SMTP serveru přeskočeny.

Pokud se nám podaří úspěšně navázat spojení a přečíst pozdrav serveru, tento pozdrav si uschováme, protože se z něj pokusíme v dalším testu zjistit software mailserveru. Následně zašleme příkaz EHLO. Jestliže bude odpovědí chyba o neznámém příkazu, víme, že se zřejmě jedná o starší verzi SMTP serveru, který nepodporuje rozšíření ESMTP. V takovém případě odešleme pozdrav HELO. Odpověď na pozdrav EHLO si opět schováme, budeme ji poté analyzovat za účelem zjištění podporovaných rozšíření.

Po úspěšných pozdravech vyzkoušíme, zda mailserver akceptuje poštu pro zkoumanou doménu. Zjistíme to tak, že se pokusíme doručit zprávu pro adresu postmaster@domena. Tato speciální e-mailová schránka musí být funkční pro každou doménu, jak je to vyžadováno v RFC 822[3] (6.3). Slouží ke kontaktování osoby, která má na starost správu dané domény a slouží také pro případ, že neznáme žádnou jinou schránku na dané doméně, ale potřebujeme někoho kontaktovat. Jiné e-mailové adresy pro jednotlivé domény nám známy nejsou, proto nám musí stačit postmaster. Po předání adresy příjemce příkazem RCPT TO komunikaci se serverem ukončíme, nebudeme ve skutečnosti nikdy žádnou zprávu doručovat.

Pokud nám server tuto adresu příjemce odmítne, může to znamenat několik věcí. Běžnější příčina spočívá v tom, že mailserver je sice uveden v MX záznamu domény, ale ve skutečnosti e-mailové služby pro tuto doménu neprovozuje (není pro tuto doménu nakonfigurován). Důvody mohou být různé, většinou je to však chyba správce domény, který uvedl nesprávné MX záznamy. Může se však také stát, že mailserver je pro doménu nastaven, ale odmítá jako příjemce adresu postmaster. To by však znamenalo porušení RFC. Bohužel nejsme schopni tyto dvě příčiny rozlišit. V každém případě to budeme považovat za chybu. Kód odpovědi si schováme k dalším testům.

Jiný problém nastane, pokud nám server odmítne naší adresu odesílatele (MAIL FROM). Je nutné používat takovou e-mailovou adresu, která skutečně existuje, protože servery si její správnost ověřují. Pokud je odmítnuta, nejsme schopni další testy provádět a ani se nedostaneme k tomu, abychom serveru předali adresu příjemce.

4.2.8 Kontaktování dalších mailserverů

V tomto testu se pokusíme kontaktovat případné servery z dalších MX záznamů, pokud nějaké další jsou. Opět požadujeme korektní navázání spojení a akceptování e-mailu na adresu postmaster@domena. Tím otestujeme, zda daný mailserver zná příslušnou doménu a je pro ni nastavený jako záložní mailserver.

Správný výsledek		Spojení se všemi záložními mailservery je v pořádku a akceptují e-maily pro tuto doménu
208-1	upozornění	Žádný další platný MX záznam, doména nemá záložní mailserver
208-2	chyba	Nepodařilo se navázat spojení s některým záložním SMTP serverem
208-3	chyba	Chybná odpověď některého serveru, není protokolem SMTP
208-4	varování	Timeout při čekání na odpověď od některého serveru
208-5	upozornění	Některý další MX záznam má neveřejnou IP adresu
208-6	chyba	Některý server neakceptuje e-maily pro postmaster@domena
208-7	chyba	Žádný další server neakceptuje e-maily pro postmaster@domena
208-8	informace	Některý ze serverů s námi odmítl komunikovat či odmítl našeho odesílatele, test nemohl být dokončen

Pokud se serverem v pořádku komunikujeme, ale neakceptuje nám příjemce postmaster@domena, opět to může znamenat, že server není pro danou doménu nakonfigurován anebo v rozporu s RFC neakceptuje schránku postmaster. Jestliže není pro doménu nastaven, není to fatální chyba, ale v případě výpadku primárního mailserveru a nemožnosti zprávu předat některému ze záložních mailserverů se mohou zprávy nenávratně ztratit. Za nejhorší chybu budeme považovat situaci, kdy se nám nezdařily testy pro žádný mailserver (e-mail nelze vůbec doručit).

4.2.9 Zjištění software serveru

Zde vycházíme z komunikace s primárním mailserverem. Z jeho prvního pozdravu se pokusíme zjistit, jaký software používá. Nemusí se nám to však z mnoha důvodů povést. Některé mailservery o sobě neprozradí nic nebo o sobě mohou úmyslně tvrdit mylné informace (ze stále opakovaných důvodů – při znalosti softwaru a jeho verze má případný útočník více informací o potenciálních slabých místech). Tento test samozřejmě nemá smysl, pokud se nám komunikace s primárním mailserverem

nezdařila (neobdrželi jsme prvotní pozdrav). Údaje získané z komunikace z případných dalších serverů zde nebereme v úvahu.

Navíc máme mnohem komplikovanější situaci než v případě zjištění software WWW serveru, protože v SMTP protokolu není stanoven žádný formát informace o softwaru. Tato informace se může a nemusí objevit v libovolném formátu a znění v prvotním pozdravu, ale může se k němu zamíchat i další text (uvítací hláška, název firmy provozovatele – v podstatě libovolná textová hláška nastavená administrátorem). Proto nemůžeme daný řetězec parsovat a vyndavat z něj možný název softwaru a jeho verzi, ale naopak musíme disponovat předem připraveným seznamem všech možných řetězců, které budeme hledat.

Další komplikací je samozřejmě to, že žádný seznam identifikačních řetězců mailserverů nemáme k dispozici, musíme si jej vytvořit. Je tedy nutné spustit testy na velký počet domén a zaznamenávat si řetězce, ze kterých se nepodařilo nic identifikovat. Pak se musí tyto logy ručně procházet a doplňovat nová známá označení. Tím bychom měli dosáhnout toho, že velmi rychle obsáhneme nejrozšířenější software a postupně budeme doplňovat méně rozšířenými. Časem dosáhneme kvalitního seznamu.

správný výsledek		Zjištěn software mailserveru
209-1	informace	Primární mailserver nezjištěn nebo chyba při komunikaci s ním
209-2	informace	Nepodařilo se zjistit software mailserveru

4.2.10 Zjištění dostupných rozšíření

Pokud mailserver pozitivně reaguje na příkaz EHLO, vypíše nám seznam rozšíření protokolu SMTP, která nabízí. V tomto testu výpis zpracujeme a vypíšeme zjištěný seznam rozšíření. Abychom mohli uživateli tohoto nástroje poskytnout více informací než jen strohý výpis označení a zkratek rozšíření, musíme si podobně jako v minulém testu postupně vybudovat seznam nám známých rozšíření a k nim připravit nějaký krátký popis, který všechna zjištěná rozšíření ozřejmí. Opět řešíme pouze rozšíření primárního mailserveru, případné záložní ignorujeme.

správný výsledek		Mailserver nabízí rozšíření (+ jejich výpis)
210-1	informace	Primární mailserver nezjištěn nebo chyba při komunikaci s ním
210-2	informace	Mailserver nepodporuje rozšíření (ESMTP)
210-3	informace	Mailserver podporuje rozšíření, ale žádná nenabízí

4.2.11 Greylisting

Jestliže nám server na příkaz RCPT TO odpoví kódem 451 (anebo jiným podobným, který značí odmítnutí z dočasných důvodů), znamená to, že cílová schránka sice existuje, ale je dočasně nedostupná. Může to ale také být zastírací manévr pro greylisting. Avšak je problém tyto 2 situace rozlišit, protože to žádný greylisting být nemusí a schránka je nyní opravdu nedostupná (přeplněna, má dočasně zakázán příjem zpráv aj.). Některé mailservery nám to ulehčí a v popisu chyby je rovnou zmínka o tom, že se jedná o greylisting:

```
451 This server employs greylisting as a means of reducing spam. Your
message has been delayed and will be accepted later.
```

Pokud se nám tedy podaří v textu odpovědi najít řetězec „grey“, máme jistotu. Jiné se o tom nezmíní a jistotu nemáme. Proto výsledkem tohoto testu jsou následující možné informativní výsledky. Žádný z výsledků není správný ani chybný, používání greylistingu je zcela na rozhodnutí administrátora serveru.

211-1	informace	Mailserver zřejmě podporuje greylisting
211-2	informace	Mailserver podporuje greylisting
211-3	informace	Mailserver nepodporuje greylisting

Aby při opakovaném testování stejné domény nedošlo k tomu, že server má greylisting aktivní, ale my jsme si svého odesílatele „povolili“ při nedávném předchozím testu, je nutné jako odesílatele v příkazu MAIL FROM uvádět vždy unikátní adresu, např. nějakou, která v sobě obsahuje aktuální časové razítko.

4.2.12 Přítomnost AAAA záznamů

Zajímá nás, zda doménové názvy mailserverů z MX záznamů mají také AAAA, tj. zda by mohly být dosažitelné přes IPv6. Spojení přes IPv6 nenavazujeme, budeme se muset spokojit s tím, že když je uvedena IPv6 adresa, tak je velmi pravděpodobně i správně, a že správce domény nemá důvod ji uvádět, pokud by byla chybná či nepoužitelná. Stačí nám, že alespoň jeden z mailserverů má IPv6 adresu. Ještě lepší samozřejmě bude situace, kdy ji budou mít všechny. Pokud ji nemá žádný, nebude nám to nijak vadit, zatím je to zcela běžné.

správný výsledek		Všechny mailservery mají IPv6 adresu (AAAA záznam)
212-1	informace	Alespoň jeden mailserver má IPv6 adresu (AAAA záznam)
212-2	informace	Žádný mailserver nemá IPv6 adresu (AAAA záznam)

4.2.13 Přítomnost SPF záznamů

Nakonec zjistíme, zda doména obsahuje SPF záznam, tedy zda definuje pravidla, kdo je oprávněn odesílat e-mailové zprávy s uvedením této domény v odesílateli. Z důvodů, uvedených v teoretických informací o SPF, se pokusíme zjistit jak přítomnost záznamu typu SPF, tak také TXT záznamu, který obsahuje SPF pravidla. Pokud alespoň jeden takový záznam nalezneme, bude to pro nás pozitivní výsledek tohoto testu. Nejlepší je samozřejmě situace, kdy existují záznamy obojího typu. Záznamy nebudeme dále nijak zkoumat.

213-1	informace	Doména nemá SPF ani TXT (v=spf1) záznam
213-2	informace	Doména má SPF záznam
213-3	informace	Doména má TXT (v=spf1) záznam

4.2.14 Mailservery jako open relay

Pokud je mailserver chybně nakonfigurován, akceptuje zprávy od jakéhokoliv klienta pro libovolného příjemce a dále je doručuje do správného cíle. To je však velmi závažný problém, protože takový server může být zneužit ke spamování. Navíc se server velmi rychle objeví v blacklistech a přestanou se z něj doručovat zprávy i od odesílatelů s dobrými úmysly. Existují samozřejmě částečné relay servery, např. SMTP

servery poskytovatelů připojení, kteří nabízí SMTP server pro své zákazníky (odesílatele ze své sítě).

Abychom mohli otestovat, zda se jedná o open relay, zkusíme doručit zprávu pro příjemce s adresou existující domény, to proto, že SMTP servery si mohou již během komunikace s námi ověřit, zda doména, na kterou chceme e-mail poslat, existuje a má MX záznam. Kdybychom použili neexistující doménu, mohly by nám servery dát negativní odpověď, i když ve skutečnosti open relay jsou. Pokud nám ji server akceptuje, jedná se o open relay server a budeme to považovat za chybu. Ovšem je potřeba dát si pozor, abychom jako open relay neoznačili mailserver námi použité domény příjemce. Test provedeme pro všechny mailservery dané domény.

Výsledky testování je vhodné cachovat, protože při testování velkého počtu domén máme velkou šanci, že více domén používá společný mailserver. To, zda je server open relay či nikoliv, nijak nesouvisí s testovanou doménou. Klíčem pro cachování výsledku stačí IP adresa, není nutné doménové jméno mailserveru.

správný výsledek		Žádný z mailserverů není open relay
214-1	chyba	Některé mailservery jsou open relay
214-2	chyba	Všechny mailservery jsou open relay

4.2.15 Mailservery v DNSBL seznamech

V tomto testu zjistíme, zda se některá z IP adres mailserverů této domény nachází na nějakém DNSBL seznamu. Pokud by tomu tak bylo a příslušný server by majitelé domény využívali i jako SMTP pro odchozí poštu, mohla by tato pošta být dalšími mailservery odmítána. Pro tyto účely je potřeba vybrat pouze takové DNSBL servery, které jsou skutečně blacklisty (tj. na blokované IP adresy pošlou odpověď a neblokované IP adresy neznají), protože podobný systém dotazování se občas používá i pro whitelisty. Seznamy DNSBL lze získat z různých zdroj, např. ze seznamu na Wikipedii [8]. Nutno dodat, že přítomnost IP adresy mailserveru v některém z DNSBL seznamů by neměla mít žádný vliv na přijímání e-mailů. V DNSBL se kontrolují IP adresy odesílatelů zpráv, nikoliv příjemců.

Pokud nám alespoň jeden DNSBL server vrátí alespoň jeden odpovídající A záznam, prohlásíme, že mailserver je blokován. Nebudeme dále zkoumat, z jakého DNSBL výsledek pochází a jaké je hodnota v A záznamu (podle kterého by se dala rozeznat příčina blokování). Navíc nás bude zajímat, zda je blokován jen některý server této domény či všechny. Uvedení v nějakém blacklistu může znamenat, že je mailserver provozován přes nějaké domácí internetové připojení (např. ADSL), což není dobrý nápad. Domácí připojení nemusí být spolehlivá natolik, aby byla stále dostupná pro přijímání e-mailů.

Pro dotazování na DNSBL je vhodné využít nějaký z lokálních cachovacích DNS serverů, aby byly odpovědi co nejvíce cachované. Při velkém počtu testovaných domén je velmi pravděpodobné, že se budou mailservery stále opakovat (velké množství domén hostuje u malého množství ISP), a tak testování DNSBL by mělo být velmi rychlé.

správný výsledek		Žádný z mailserverů není uveden v žádném DNSBL
215-1	varování	Některý z mailserverů je uveden v DNSBL
215-2	varování	Všechny mailservery jsou uvedeny v DNSBL

Součástí výstupu testu je seznam DNSBL serverů, kde byl nějaký záznam nalezen. Získané údaje je však potřeba brát trochu s rezervou. Různé DNSBL servery se zaměřují na různé druhy IP adres. Některé DNSBL prověřují zdroje spamu ručně, a tak jsou zřejmě korektní, jiné mohou seznamy IP získávat nějakým způsobem automaticky či dokonce na základě anonymního oznámení bez důkladného prověření, a tak uvedení v seznamu nemusí být odůvodněné. Tak či onak, uvedení IP adresy v kterémkoliv seznamu může někdy přinést problémy, protože nikdy nevíme, které seznamy jsou využívány kterými SMTP servery.

4.3 Dostupnost a informace o telefonních službách

V této části provedeme jen několik velmi jednoduchých testů, které nám poskytnou základní informace o podpoře SIP služeb na dané doméně. Se SIP serverem komunikovat nebudeme, to už bychom v analýze služeb poskytovaných domén zašli hodně daleko. Provedeme pouze jednoduché zkoumání příslušných DNS záznamů.

4.3.1 Zjištění SRV SIP záznamů

První otázkou je, zda daná doména vůbec služby SIP nabízí. Dotážeme se tedy DNS serveru na záznamy typu SRV s názvem „_sip._udp.domena“ a „_sip._tcp.domena“. Pokud dostaneme alespoň jednu pozitivní odpověď, doména SIP služby nabízí.

301-1	informace	Doména nenabízí služby SIP
301-2	informace	Doména nabízí služby SIP nad UDP protokolem
301-3	informace	Doména nabízí služby SIP nad TCP protokolem
301-4	informace	Doména nabízí služby SIP nad UDP i TCP protokolem

Ve výsledku testu zohledníme záznamy pro TCP a UDP. Jestliže žádné SRV záznamy nezískáme, další testy, které souvisí se SIP, provádět nebudeme.

4.3.2 Kontrola syntaxe SRV záznamů

Pokud jsme v předchozím testu získali nějaké SRV záznamy, provedeme nejprve jejich syntaktickou kontrolu. Sice je nepravděpodobné, že by DNS server umožnil do zóny domény vložit chybné záznamy SRV, ale pro raději se ujistíme. Opět nás také zajímá, zda namísto názvu SIP serveru není vložena přímo jeho IP adresa. Jestliže jsou nějaké SRV záznamy syntakticky chybně, další související testy neprovádíme.

Správný výsledek		Všechny SRV záznamy mají správnou syntaxi
302-1	chyba	Některý SRV záznam má chybnou syntaxi
302-2	varování	Některý SRV záznam obsahuje IP adresu

4.3.3 Překlad názvů SIP serverů v SRV záznamu na IP adresy

Pokud jsme v předchozím testu získali nějaké SRV záznamy a jsou syntakticky správně, prozkoumáme jejich obsah. Budou nás zajímat obsažené doménové názvy SIP serverů. Jako obvykle je zkusíme přeložit na IP adresu.

Správný výsledek		Všechny názvy ze SRV záznamů lze přeložit na IP adresu
303-1	chyba	Některé názvy ze SRV záznamů nelze přeložit na IP adresu
303-2	chyba	Žádný název ze SRV záznamů nelze přeložit na IP adresu

4.3.4 Duplicita SRV záznamů

Je zbytečné, aby administrátor domény uváděl jeden SIP server vícekrát v rámci stejného transportního protokolu (TCP a UDP). Smysl má pouze více SRV záznamů s různými SIP servery pro rozložení zátěže a zálohování v případě poruchy. Provedeme tedy kontrolu duplicity SIP serverů, pro každý transportní protokol zvlášť. Podobně provedeme i porovnání v rámci IP adres těchto SIP serverů.

Správný výsledek		SRV záznamy nejsou duplicitní
304-1	chyba	SRV záznamy obsahují duplicitní doménový název
304-2	varování	Některé doménové názvy v SRV záznamech směřují na stejnou IP adresu

4.3.5 Kontrola TTL SRV záznamů

Opět kontroly shody TTL u všech záznamů, odděleně pro subdoménu _sip._tcp a _sip._udp.

Správný výsledek		TTL hodnoty u SRV záznamů se shodují
305-1	chyba	Neshoda TTL u SRV záznamů pro TCP
305-2	chyba	Neshoda TTL u SRV záznamů pro UDP

4.3.6 Reverzní záznamy názvů SIP serverů

Obvyklá kontrola reverzních záznamů. Případné zjištěné problémy nepovažujeme za vážné chyby, a tak budou klasifikovány jen jako varování. Reverzní záznamy SIP servery zřejmě nijak nekontrolují, ale je slušnost je mít.

Správný výsledek		Reverzní záznamy se shodují s jejich IP adresami
306-1	upozornění	Reverzní záznamy se neshodují u všech SRV záznamů
306-2	upozornění	Reverzní záznamy se neshodují u některých SRV záznamů (nikoliv u všech)
306-3	upozornění	Reverzní záznamy u některých SRV záznamů neexistují

4.3.7 Přítomnost AAAA záznamů pro SIP servery

Nakonec obvyklý test, zda je možné některý ze SIP serverů kontaktovat přes IPv6, tedy zda jeho doménový název má odpovídající AAAA záznam. Správným výsledkem testu je, že jsou všechny SIP servery dosažitelné přes IPv6, ale předpoklad je, že to v současné době budou velké výjimky a že to chce ještě mnoho let.

správný výsledek		Všechny SIP servery mají IPv6 adresu (AAAA záznam)
307-1	informace	Alespoň jeden SIP server má IPv6 adresu (AAAA záznam)
307-2	informace	Žádný SIP server nemá IPv6 adresu (AAAA záznam)

4.4 Dostupnost bezpečnostních prvků DNS

4.4.1 Přítomnost DNSKEY záznamu

V této sekci otestujeme pouze jednu věc, a to přítomnost alespoň jednoho DNSKEY záznamu v doméně. Pokud takový záznam existuje, nebudeme jej nijak zkoumat ani ověřovat správnost. Budeme předpokládat, že pokud existuje, tak je doména skutečně chráněna technologií DNSSEC. Za správný výsledek považujeme, když daný záznam existuje, a zóna domény je tedy zřejmě podepsána, a budeme doufat, že takto budou časem podepsány všechny domény.

Správný výsledek		Doména obsahuje DNSKEY záznam
401-1	informace	Doména neobsahuje DNSKEY záznam

Kapitola 5 – Implementace

5.1 Technologie

Jako v případě bakalářské práce byl celý systém vyvinut v technologii PHP, důvodů je několik – již hotové testy z předchozí práce, bohaté zkušenosti autora s touto technologií a orientace na veřejné rozhraní celé služby. Jako databázový server byl opět zvolen MySQL, který sice někteří kritizují a namítají, že se jedná o amatérský projekt, avšak podle mého názoru doba pokročila a MySQL už je dávno robustní a léta prověřený databázový systém, který nabízí vše co je potřeba. Oproti bakalářské práci se jako úložiště dat používá InnoDB s transakcemi.

Problémem PHP je, že není určeno k vývoji stále běžících procesů, které na pozadí něco zpracovávají. PHP je klasický skriptovací jazyk, jehož skript se spustí, něco vykoná a ukončí se. Ovšem s použitím vhodných technik může PHP skript stále běžet na pozadí jako jakési asynchronní vlákno. Musí se však myslet na problémy, které při běhu jedné aplikace ve více vláknech vznikají – synchronizace, vzájemné vyloučení, restart „spadlého“ vlákna apod.

Celá aplikace opět běží nad PHP frameworkem vlastní výroby, který se stará o komunikaci s databázovým serverem, zpracováním chyb v aplikaci, generování stránek v HTML jazyce, kontrolu syntaxe a formátování řetězců, logování aktivity aj.

5.2 Práce s WWW servery

Komunikace s WWW servery je zcela bezproblémová. Oproti protokolu SMTP nebývá HTTP protokol přímo zneužíván k nějakým nekalým účelům (např. rozesílání spamu). WWW servery tedy nepoužívají žádné automatické mechanismy pro kontrolu IP adres klientů, omezování počtu spojení apod. Ke komunikaci s WWW servery se používá třída net_http, která byla vyvinuta speciálně pro tuto aplikaci, jelikož nabízí práci na nejnižší úrovni a umožňuje získávat jakékoliv detaily o obsahu komunikace. Používá se protokol HTTP verze 1.1, verze 1.0 se neřeší. Je totiž zastaralá a předpokládá se, že již není používaná.

5.3 Práce s mailservery

Komunikace s mailservery přes SMTP je výrazně komplikovanější. SMTP servery se brání zuby nehty, aby přes ně nikdo neposílal spam. Kontrolují zdrojové IP adresy, adresy odesílatele i příjemce, omezují počet spojení za určitý časový interval, automaticky detekují podivná chování aj. S narůstajícím počtem domén k otestování se tedy zvyšuje pravděpodobnost, že nás různé SMTP servery z nějakého důvodu zablokují. To vše i přesto, že ve skutečnosti nikdy žádný e-mail neodešleme (skončíme nejpozději příkazem RCPT TO, na příkaz DATA nikdy nedojde). Nejprve nás zablokují dočasně na nějakou dobu, všímavý administrátor nás však může zablokovat trvale. Pravděpodobnost těchto problémů můžeme snížit zvýšením počtu uzlů, ze kterých se testování provádí.

V příkazu MAIL FROM se používá e-mailová adresa s doménou dns-info.cz a před zavináčem je aktuální UNIXový čas (abychom omylem neprošli greylistingem při opakovaném testování stejné domény). Ukázalo se, že některé mailservery ihned po poslání příkazu MAIL FROM kontrolují, zda doména odesílatele má MX záznam. Proto byl doméně dns-info.cz takový záznam zaveden, i když cílový mailserver ve skutečnosti e-maily pro tuto doménu neakceptuje. Několik serverů však šlo v kontrolách ještě dál a ihned po MAIL FROM kontaktovaly mailserver odesílatele a ověřovaly si, zda e-mailová adresa existuje. Vyvstává otázka, zda při takovém kontrolování nemůže dojít k zacyklení, kdy se 2 mailservery začnou takto vzájemně kontrolovat (doufejme, že jejich implementace s takovou situací počítá).

Největší problém nastal s detekcí softwaru mailserveru. Drtivá většina serverů o sobě nic neprozradí. Podařilo se zjistit pouze 19 programů (mezi nimi např. Postfix, Exim, Kerio Mailserver, Microsoft Exchange aj.). Ale např. velmi používaný software qmail jsme nezjistili nikdy. Je tedy náhoda, pokud se nám software zjistit podaří. Při zkušebních analýzách malého množství domén byla úspěšnost detekce softwaru méně než 1%.

Dalším problémem se ukázalo blokování rozsahů IP adres, které jsou dynamicky přidělovány uživatelům ADSL apod., jak bylo zmíněno v informacích o DNSBL.

Výsledkem bylo to, že nebylo možné SMTP testy zkoušet z počítače z domova, protože většina SMTP serverů se odmítla bavit.

Nakonec nastal problém s detekcí greylistingu a dalších očekávaných chybových stavů. Původní očekávání, že všechny SMTP servery používají několik málo přesně daných návratových kódů, se ukázalo jako mylné. Různé servery používají pro stejný výsledek jiný kód odpovědi. Jen pro odmítnutí příjemce bylo zjištěno 17 různých kódů. Problematickou se ukázala i správná detekce greylistingu, musí se totiž rozlišit různé důvody odmítnutí příjemce (zda příjemce je odmítán stále anebo jen dočasně pro greylisting). Jako nejspolehlivější se ukázalo hledání řetězce „grey“ či „gray“ v textu odpovědi. Jinak si nemůžeme být jisti důvodem.

Celkově shrnuto – se SMTP servery jsou příliš velké problémy a nemá smysl je zahrnout do hromadných analýz. Výsledky by byly bohužel příliš divoké a o ničem by nevypovídaly.

5.4 DNSBL servery

Seznam DNSBL serverů byl sesbírán z mnoha různých zdrojů. Při jejich zkoušení se však ukázalo, že některé neodpovídají vůbec a jiné jsou pomalé. Některé dokonce zasílají pozitivní odpověď na jakoukoliv IP adresu. Nakonec bylo zjištěno, že to bylo z důvodu ukončení provozu těchto DNSBL a vracená IP adresa směřovala na WWW stránku s vysvětlujícím textem. Seznam se nakonec ustálil na 36 DNSBL serverech. Mezi nejvýznamnější patří:

- zen.spamhaus.org
- bl.spamcop.net
- db.wpbl.info
- dnsbl.proxybl.org
- spam.spamrats.com
- bl.spamcannibal.org

5.5 Vzdálené volání procedur (XAPI)

Pro účely tohoto systému byla vytvořena třída pro vzdálené volání procedur, tato implementace RPC byla pojmenována XAPI (external API), tedy rozhraní pro externí volání. Jako transportní protokol se používá HTTP. Serverová část XAPI tedy běží na WWW serveru, „naslouchá“ na určité URL adrese a čeká na příchozí požadavky od klientů. Samozřejmě lze použít šifrování, tedy transportní protokol HTTPS (tj. HTTP nad SSL), to však tato vrstva neřeší, musí se o to postarat WWW server.

Je to obvyklé RPC řešení, tzn. že na straně klienta je dostupná knihovna, která nabízí rozhraní pro vzdálené volání a plně řeší veškeré záležitosti co se týče vyhledání a kontaktování cílového serveru, formátu dat, jejich přenosu po počítačové síti, zpracování odpovědi, řešení nestandardních situací, poruch atd. Podobně je to řešeno na straně serveru, následkem čehož programátor (jakožto uživatel tohoto rozhraní) nemusí podobné věci vůbec řešit. Na straně serveru mu jen stačí nějakým způsobem „vystavit“ metodu třídy, kterou má být možné volat zvenku, a na straně klienta ji vhodným způsobem zavolat přes stanovené rozhraní.

XAPI bylo implementováno samozřejmě v PHP, nicméně jeho principy a protokol žádným způsobem neomezují použití pro komunikaci s aplikacemi nad jinými technologiemi. Nejedná se o opravdové RPC – na straně klienta se negenerují žádné stuby a skeletony, na klientské straně se explicitně volá XAPI rozhraní, na serverové straně jsou metody explicitně upravené pro přijímaní požadavků přes XAPI.

5.5.1 Klient

Na straně klienta je k dispozici statická metoda xapi::call(), té se předávají následující údaje:

- způsob volání
- název cílové aplikace
- název metody
- parametry metody
- dodatečné parametry pro XAPI

XAPI umožňuje volat cílovou metodu v cílové aplikaci v zásadě dvěma způsoby – synchronně a asynchronně. Při synchronním použití se provede volání vzdálené aplikace okamžitě a čeká se na výsledek, který se vrátí volajícímu. Při asynchronním použití je požadavek uložen do fronty a vykoná se později (během několika sekund, max. během minuty), na výsledek se nečeká a ani není umožněno se jej přímo dozvědět. Nic však nebrání tomu, aby nám druhá strana později poslala výsledek našeho požadavku dalším voláním XAPI (v opačném směru). Dále je k dispozici 3. způsob, při kterém se XAPI pokusí požadavek provést synchronně, a pokud se to nezdaří (např. porucha na síti či jiný podobný problém), uloží jej do fronty a později vykoná asynchronně.

Pro provádění asynchronního volání se používá třída async, která byla vyvinuta též pro tyto účely a která do PHP zavádí možnost běhu asynchronního vlákna, které si z fronty průběžně vyzvedává operace, které má vykonávat. Zjednodušeně je to PHP skript, který stále běží na pozadí a v pravidelných intervalech sleduje frontu požadavků, které jsou umístěny v souborovém systému na lokálním disku. Třída async je opět dělána univerzálně, neslouží pouze pro potřeby XAPI, ale pro asynchronní zavolání jakékoliv metody s jakýmikoliv parametry v PHP. Navíc lze stanovit časové zpoždění, udávající po jaké době se může operace nejdříve provést, protože nemá smysl neúspěšný XAPI požadavek opakovat okamžitě, lepší je např. několik minut počkat. Jestliže se asynchronní volání XAPI nezdaří, opět se zařadí do fronty a nastaví se mu nějaké zpoždění. Toto řešení však nijak negarantuje skutečnou časovou prodlevu. Pokud je ve frontě k vyřízení mnoho požadavků, jsou vykonávány v pořadí, v jakém byly do fronty vloženy, a musí čekat, až na ně dojde řada. Není to tedy vhodné na vykonávání časově náročných operací, které by blokovaly všechny ostatní (v takovém případě by bylo lepší vlastní asynchronní vlákno).

Při volání se uvádí název cílové aplikace. Pod tímto názvem se v konfiguraci skrývá zejména URL adresa cílové aplikace, na které XAPI rozhraní serveru „naslouchá“, a dále název výchozí třídy, v níž se má hledat volaná metoda. Předpokládá se totiž, že právě jedna třída na straně serveru příslušné aplikace se stará o přijímání požadavků přes XAPI (lze to však obejít, viz. níže). Dále se uvede název metody a její parametry. Tyto parametry mohou být libovolného datového typu PHP, zadaná data budou totiž serializována a na druhé straně rekonstruována do původní podoby. K tomu se používá

formát dat JSON (JavaScript Object Notation), definován v RFC 4627[20], který byl sice původně určen pro AJAXové WWW aplikace k předávání dat mezi JavaScriptem na straně klienta a aplikace na straně serveru, ale je to nyní již obecně známý formát, podporovaný snad ve všech současných programovacích jazycích a technologiích. Díky tomu není problém XAPI rozhraní implementovat i v jiných technologiích.

Nakonec v dodatečných parametrech pro XAPI lze uvést název třídy, pokud se liší od třídy výchozí. Pokud selže spojení s cílovým serverem, XAPI se o to ihned pokusí znovu (aby se vyloučil případ, že spojení nebylo navázáno z důvodu nějakého náhodného faktoru či chvilkové slabosti sítě či serveru) – i toto lze v dodatečných parametrech zakázat.

5.5.2 Server

Jak již bylo řečeno, serverová část se mimo samotné knihovny sestává z PHP skriptu, který je umístěn na WWW serveru na určité URL adrese a přes HTTP protokol přijímá požadavky. Na vstupu obdrží informace o požadované třídě, název metody, parametry a režijní informace XAPI. Zkontroluje existenci příslušné třídy a metody (veřejná statická metoda třídy) a zavolá ji. Tato metoda na serveru se musí ve skutečnosti jmenovat trochu jinak, než se uvádí v požadavku od klienta – musí mít ve svém názvu navíc prefix „xapi_“, aby bylo zřejmé, že se jedná o metodu adaptovanou pro XAPI rozhraní a aby nebylo možné přes XAPI zavolat (ať úmyslně či neúmyslně) libovolnou metodu (i takovou která to nečeká a mohlo by dojít k nějaké chybě či narušení bezpečnosti). Jako první parametr se metodě předají vstupní data od klienta (po deserializaci). Pokud tedy potřebujeme metodě předat více parametrů, musíme jí je např. předat jako jedno asociativní pole či jako nějaká data v jednom objektu. Metoda obdrží od XAPI i druhý parametr, ve kterém se nachází asociativní pole s režijními daty XAPI rozhraní, ze kterého se lze dozvědět např. který server a která aplikace požadavek zasílá.

Návratová hodnota cílové metody, která opět může být libovolného typu (bude serializována a následně deserializována), se předá zpět jako odpověď a na straně klienta se objeví jako návratová hodnota metody xapi::call(). Pro indikaci chyb při XAPI volání se používají speciální návratové hodnoty.

5.5.3 Protokol

Klientská a serverová část XAPI si vyměňují data tak, že klient zavolá příslušnou URL adresu serveru a jako POST v HTTP předá serializované asociativní pole (formát JSON), které ve svých položkách obsahuje základní informace, režijní údaje a vstupní parametry. Řetězec, který je výsledkem serializace, je ještě před přenosem zkomprimován metodou bzip2. Tím se významně sníží objem přenášených dat (zvláště když se v našem případě jedná zejména o textová data o doménách a přenášíme informace o tisících doménách). Před výsledná data se navíc vloží kontrolní řetězec, podle kterého druhá strana pozná, že se opravdu jedná o data protokolu XAPI. Tento řetězec také obsahuje číslo verze protokolu, díky čemuž lze rozpoznat formát dat a systém může být v budoucích verzích zpětně kompatibilní.

Na straně serveru se v datech z POST nejprve zkontroluje počáteční kontrolní řetězec, dekomprimují se data a provede se jejich deserializace. Tím by měl server získat původní pole se všemi potřebnými údaji o třídě, metodě, vstupních parametrech a další režijní informace. Stejným způsobem probíhá sestavení a zaslání odpovědi s návratovou hodnotou volané metody anebo chybovým kódem.

Protokol XAPI žádným způsobem neřeší autentizaci, autorizaci a šifrování. O zabezpečení se musí postarat správce WWW serveru např. tím, že omezí přístup na danou URL adresu jen z určitých IP adres, umožní komunikaci přes SSL a v URL serveru na straně klienta uvede protokol HTTPS.

5.6 Distribuované zpracování

5.6.1 Úvod do distribuovaných systémů

Software pro analýzu DNS je specifický tím, že většinu času čeká. Stále čeká na odpovědi od DNS serverů. Protože mohou být tyto servery rozmístěny různě po celém světě, odpovědi se vracejí za několik desítek až stovek milisekund. Často také spojení s nějakým serverem selže, protože není dostupný nebo z jiného důvodu nereaguje, tudíž se čeká, až vyprší všechny timeouty pokusů o spojení. Tento nástroj pro analýzu načítá přes Internet mnohem více dat, nejen z DNS serverů – informace o sítích a autonomních systémech z WHOIS serverů, pokouší se kontaktovat zjištěné WWW a mailservery a

další. Když se to všechno sečte, zkoumání jedné domény může trvat od několika stovek milisekund až po mnoho sekund. Tímto tempem otestovat několik stovek tisíc domén či dokonce několik milionů domén je samozřejmě zcela nereálné. A tak musíme analyzovat paralelně, tedy spustit několik vláken, která si mezi sebe práci rozdělí. Tato vlákna mohou běžet na jednom počítači, několik desítek až stovek jich jeden počítač s dostatkem paměti a procesorovým výkonem může zvládnout bez problémů.

Dalším krokem je samozřejmě běh těchto vláken na více počítačích, a to je jeden z cílů této práce. Dostáváme se do oblasti distribuovaného zpracování a pro vytoužený výsledek musíme vyřešit spoustu otázek a problémů, které s tím souvisí. Musíme také najít nějaký kompromis ohledně složitosti výsledného systému, náročnosti implementace, způsobu komunikace, řešení nestandardních situací a poruch aj.

Mějme tedy několik počítačů, které použijeme pro naši analýzu. Musíme je nějak donutit, aby spolu komunikovaly, rozdělily si mezi sebe práci, daly dohromady získaná data do celkového výsledku, získávaly data a zapisovaly výsledky do společného úložiště aj. Naše řešení musí být takové, aby byl celý systém snadno škálovatelný a rozšiřitelný, aby zapojení či odpojení počítače bylo jednoduché. Je potřeba řešit nestandardní situace jako porucha v komunikaci, výpadek jednoho z počítačů, zahájení a ukončení práce a další.

Ideální distribuovaný systém se skládá z nezávislých uzlů, které jsou vzájemně propojeny, a uživateli poskytuje dojem jednotného systému, nemá žádný centralizovaný prvek, uzly se rozhodují samostatně a všechny použité protokoly a algoritmy jsou distribuované a decentralizované. Tohoto ideálu distribuované krásy však není jednoduché docílit. Aby bylo reálné takový systém implementovat, budeme muset z některých požadavků slevit.

Podrobnou teorii distribuovaných systémů lze nalézt např. v materiálech k předmětu Principy distribuovaných systémů, který vyučuje katedra softwarového inženýrství na MFF UK.

5.6.2 Architektura systému

Řešení budeme muset mít centralizované, bude tedy existovat jeden předem pevně daný centrální server, který bude sloužit jako úložiště dat, bude rozdělovat práci mezi uzly, přijímat zjištěná data o doménách od uzlů a sestavovat celkové výsledky. Taktéž bude sloužit k ovládání celého systému administrátorem. Kdybychom chtěli mít decentralizované řešení, museli bychom mít distribuované úložiště dat, řešit synchronizaci uzlů, volbu koordinátora aj. To by však bylo téma na samostatnou diplomovou práci. Naše řešení se velmi podobá distribuovaným výpočtům po Internetu – např. projektu SETI@home (http://setiathome.ssl.berkeley.edu/), platformě BOINC (http://boinc.berkeley.edu/) aj. Způsobu práce těchto systémů se obecně říká grid computing.

Centralizované řešení znamená, že funkčnost celého systému bude závislá na jednom centrálním počítači. To má nevýhody spočívající v problémech, které nastanou při jeho výpadku. Musíme se však zamyslet nad tím, zda nás jeho porucha opravdu tolik zasáhne. Nevyvíjíme systém, který musí mít dostupnost 100% a musí reagovat v reálném čase. Když bude mít server výpadek několik minut či hodin, nic hrozného se nestane, analyzování domén se o nějakou dobu zpozdí anebo se nějakou chvíli nepodíváme na aktuální výsledky. Navíc rozdělování práce mezi uzly lze mnohými zlepšováky upravit tak, aby se uzly bez hlavního serveru nějakou dobu obešly – pokusíme se je tedy udělat co nejméně závislé na serveru a minimalizujeme komunikaci. Uzly tedy budou samostatně funkční inteligentní jednotky.

Další nevýhodou je, že server bude potenciálním úzkým místem co se týče výkonu. Počet uzlů bude možné rozšiřovat do nekonečna, ale server musí zvládat komunikaci se všemi, evidovat všechny úkoly a jejich výsledky a všem rozdělovat práci. Na server tedy mohou být kladeny vysoké nároky co se týče výpočetního výkonu, datového prostoru a propustnosti sítě. Jakmile přestane stíhat server anebo se porouchá či zahltí komunikační síť u serveru, stane se celý systém pomalým či dokonce nepoužitelným. To je však daň za jednoduchost systému a musíme myslet na to, abychom v závislosti na zamýšleném použití a zátěži použili dostatečně výkonný server. Samozřejmě je možné onen jeden fyzický server rozdělit na více serverů, které mezi sebou sdílí databázi a jsou schované např. za nějakým load balancerem – tím se ale zde nebudeme zabývat, principiálně tomu však nebude nic bránit. A pokud budeme analyzovat více

různých nesovisejících dat, můžeme snadno vytvořit více instancí tohoto distribuovaného systému, které budou na sobě zcela nezávislé a každá bude mít svůj centrální server. Tím můžeme např. v našem případě paralelně otestovat domény ve více TLD nezávisle na sobě.

5.6.3 Univerzálnost distribuovaného řešení

Bylo by nevhodné celé distribuované řešení implementovat jen pro použití s analýzou DNS. Bude vhodnější jej implementovat tak, aby bylo univerzální a nezávislé na konkrétním použití. To znamená, že tato distribuovaná „vrstva“ nebude řešit konkrétní obsah úkolů ke zpracování (a ani obsahu úkolů nebude muset rozumět). Její prací bude pouze nějakou sadu úkolů vhodně evidovat, rozdělovat práci mezi uzly, řešit poruchy uzlů a síťové komunikace, shromažďovat výsledky a poskytovat správu celého systému administrátorovi.

Distribuovaný systém bude mít svou serverovou část, která bude přijímat nové úkoly od administrátora (či jiného systému), evidovat je a rozdělovat, a dále klientskou část, která poběží na uzlech, bude stahovat nového úkoly ze serveru, předávat je ke zpracování dalším systémům na uzlu, přijímat výsledky a přeposílat je zpět na server. Ostatní systémy na serveru nebudou řešit, kdo, kdy a jak úkoly zpracuje, nebude je zajímat nějaké rozdělení úkolů mezi servery, síťová komunikace, připojování, odpojování a výpadky uzlů – prostě úkoly předají na server ke zpracování a pak se dozví výsledky, o distribuovaném zpracování nebudou mít ponětí. Podobně ostatní systémy na uzlech nebudou řešit, kde se úkoly vzaly a co se stane s jejich výsledky – dostanou na uzlu jeden úkol k vyřízení a vrátí výsledek.

Také systém pro zpracování konkrétního typu úkolu bude mít svou serverovou a klientskou část. Serverová část předá data úkolů přes určité rozhraní distribuovanému systému a později se přes stejné rozhraní dozví výsledky a provede jejich vyhodnocení. Klientská část bude naslouchat na určeném rozhraní na uzlu a bude od distribuovaného systému dostávat úkoly ke zpracování. Vypracované výsledky předá přes stejné rozhraní zpět.

Dále musíme myslet na to, aby jeden distribuovaný systém umožnil zpracovávat více druhů úkolů zároveň, tedy aby měl, zjednodušeně řečeno, více vstupů a výstupů a nebyl vázán na konkrétní použití. Musí tedy rozlišovat typ úkolů a podle toho na serveru komunikovat s určitým procesem pro tento typ a podobně na uzlu předávat úkoly správným typům procesů.

5.6.4 Zabezpečení

Ideální by bylo, kdybychom do celého systému zavedly prvky autentizace a autorizace, tedy aby bylo možno do sítě zapojovat pouze důvěryhodné uzly na základě prověření jejich totožnosti a kontroly oprávnění. Server by si tak musel vést seznam uzlů a jejich oprávnění či používat jiný mechanismus tak, aby mohl rozhodovat, který uzel do systému vpustí a který nikoliv.

My si to však opět zjednodušíme. Budeme předpokládat, že se do tohoto systému nebudou zapojovat anonymní uzly z celého světa a že se bude jednat o uzavřený systém, jehož všechny uzly budou plně pod naší kontrolou. Systém nebude implementovat žádnou autentizaci a autorizaci a nebude si pro tyto účely udržovat žádný seznam uzlů. Omezení síťové komunikace se ponechá na správci sítě, např. omezení na úrovni IP adres a TCP portů na firewallech.

Ze stejných důvodů budeme předpokládat, že od uzlů budeme vždy dostávat korektní výsledky. Kdyby se jednalo o otevřený systém, mohlo by se stát, že by nám chtěla některá cizí osoba zapojená do systému škodit a mohli bychom od příslušného uzlu dostávat úmyslně zkreslené či zcela špatné výsledky. To by se muselo řešit tak, že jeden úkol by se zaslal ke zpracování více uzlům a získané výsledky by se porovnávaly. Tím by se dal identifikovat škodící uzel. To vše by však spolehlivě fungovalo jen v případě, že výsledek nějakého konkrétního úkolu by byl vždy přesně stejný nezávisle na tom, kdy a kde je spuštěn (např. matematické výpočty). Navíc by to znásobilo čas potřebný k provedení úkolů a spotřebovalo by to mnoho procesorového času pro porovnávání výsledků. V případě našich testů domén to však neplatí – data v DNS se mohou průběžně měnit, v jiných světových lokalitách můžeme získávat různá data (např. pokud DNS server nějaké domény vrací různé IP adresy podle lokality dotazujícího se klienta), některé testy se nám nemusí zdařit z důvodu náhodného výpadku spojení s nějakým

serverem apod. Tudíž i z těchto důvodů zůstaneme v uzavřeném systému a nebudeme se tím zabývat.

5.6.5 Připojování a odpojování uzlů

Dalším tématem je připojování uzlů do sítě, jejich odpojování ze sítě a řešení výpadku uzlu. Jednou z možností je úplná evidence všech uzlů serverem, který by v určitých intervalech zjišťoval dostupnost uzlů a detekoval jejich výpadek. Před odpojením ze sítě by se uzly mohly na serveru odhlásit, po připojení by se mohly přihlásit. Server by si pro každý uzel evidoval údaje o tom, jaké úkoly právě zpracovává. Při výpadku nebo odhlášení by dané úkoly ihned předal jinému uzlu anebo by se uzel sám mohl před svým odpojením postarat o předání svých úkolů nějakému sousednímu uzlu.

My si to však opět zjednodušíme, minimalizujeme komunikační složitost a zavedeme čistý model klient-server. Server nebude uzly nikdy kontaktovat a nebude mít o nich žádný přehled. Navazovat spojení se serverem budou uzly – když nebudou mít co dělat, požádají server o další práci. Když nějakou práci dokončí, zašlou serveru výsledky. To, že nějaký uzel vypadl či se odpojil, pozná server tak, že pro zaslanou práci vyprší timeout. Uzly o sobě vzájemně vědět nebudou. To vše nám velmi zjednoduší přidávání dalších uzlů, prostě jej zapojíme, sdělíme mu adresu serveru a hotovo, nebude nutné nikde jinde (např. na serveru) cokoliv konfigurovat.

Hlavní nevýhodou tohoto řešení, ve kterém skoro nikdo o nikom neví, je situace při havárii uzlu. Uzel během zpracovávání úkolu nikomu nesděluje, jak je daleko s jeho řešením, a nenabízí žádné mezivýsledky, a tak při jeho havárii přijdeme o všechnu jeho rozpracovanou práci. I když bude uzel vědět, že za chvíli bude ukončena jeho činnost, nebude schopen úkol předat k dokončení na jiný uzel.

5.6.6 Rozdělení a granularita úkolů

Z tohoto a dalších důvodu je vhodné se na začátku před použitím systému zamyslet nad tím, jakou granularitu použijeme při rozdělování úkolů. V našem případě, kdy budeme chtít analyzovat domény, se jeden úkol bude sestávat z nějakého počtu domén. Pokud zvolíme příliš malé množství, zvýšíme tím komunikační režii, protože zpracování

takových úkolů bude velmi krátké a uzel se brzy bude dožadovat dalších úkolů. Ale na druhou stranu při poruše uzlu ztratíme málo práce. Jestliže ale naopak zvolíme dávky domén velké, snížíme komunikační režii – uzel bude hodně dlouho pracovat samostatně na úkolu a nebude mít potřebu s nikým komunikovat, což navíc také sníží dopady v případě, že se porouchá server (uzel se bude s menší pravděpodobností nudit, možná si výpadku serveru ani nevšimne). Na druhou stranu však havárie uzlu bude znamenat mnoho ztracené práce a času. Je tedy potřeba najít nějaký kompromis.

Dále je nutné zapřemýšlet nad situací, co se má stát, když uzel sice havaruje, ale po chvíli opět ožije (např. krátkodobý výpadek proudu či úmyslný restart celého uzlu administrátorem). Pokud si uzel průběžně ukládá rozdělanou práci na svůj lokální disk, není celá práce ztracená a může navázat na poslední uložený stav. Není však vhodné, aby si rozdělanou práci zapisoval na disk neustále. Proto je opět potřeba najít nějaký kompromis.

Zde použijeme řešení takové, že jeden úkol se bude celý zpracovávat pouze v paměti a žádné mezivýsledky se zapisovat nebudou. Aby se nestalo, že po naběhnutí nebude mít uzel co dělat (např. nebude mít tou dobou spojení se serverem, ať je příčina jakákoliv), může se úkoly předzásobit. To znamená, že si uzel bude ze serveru stahovat více úkolů než je schopen v jeden okamžik paralelně zpracovávat. Bude mít tedy svou lokální frontu úkolů čekajících na vyřízení, kterou bude vhodným způsobem udržovat, aby v ní bylo tak akorát úkolů. Je potřeba si promyslet, co znamená to „tak akorát“. Když jich bude příliš málo, může se uzel nudit v případě, že si nedokáže rychle frontu úkolů doplňovat ze serveru a nebude mít dostatečnou zásobu práce, kdyby server havaroval. Když jich bude příliš mnoho, může se stát, že uzel nebude stíhat úkoly rychle vyřizovat a server po uplynutí timeoutů pojme podezření, že zpracování nějakého úkolu zhavarovalo, a tak ten samý úkol předá dalšímu uzlu. Tím se stane, že jeden úkol bude zpracován duplicitně a budeme mrhat procesorovým časem.

Z výše uvedeného tedy shrneme, že je potřeba si promyslet:

- velikost úkolů (např. dle průměrné délky zpracování jednoho úkolu)
- horní a dolní mez velikosti lokální fronty na uzlu
- timeout pro zpracování úkolů (na straně serveru)

- počet uzlů a počet vláken na jednom uzlu (záleží na tom, jak rychle chceme mít kompletní výsledky všech úkolů)

Nutno zdůraznit, že uvedený timeout pro zpracování úkolu se neměří od okamžiku, kdy uzel začne úkol skutečně zpracovávat, ale od okamžiku kdy si úkol stáhne ze serveru. Server totiž neřeší, že má uzel nějakou frontu – předá úkol uzlu a začne odpočítávat čas, do kterého musí uzel dodat výsledek. Pokud nastane timeout, server neřeší příčinu. Označí si ve svých záznamech daný úkol zpět jako čekající a při nejbližší příležitosti jej předá dalšímu (ale i klidně tomu samému) uzlu. Samozřejmě se může stát, že dostane výsledek stejného úkolu vícekrát – zde stačí duplicitní výsledek prostě ignorovat.

Dále nás zajímá situace, kdy dojde k havárii serveru nebo ztrátě spojení se serverem a uzel dokončil nějaký úkol. Jeho výsledky se mu nepodaří serveru předat, ale není důvod, aby příslušné vlákno nic nedělalo a čekalo na obnovení spojení se serverem. Proto budou mít uzly také fronty výsledků úkolů, do té budou vlákna ukládat výsledky a ty se pak budou asynchronně odesílat na server. Vlákno nebude čekat na úspěšné doručení výsledků a hned se pustí do dalšího úkolu (pokud ve frontě čekajících úkolů nějaký je).

5.6.7 Komunikace a RPC

Jak již bylo vyřešeno v textu výše, celý systém bude založen na principu klient-server. Server bude u sebe evidovat úkoly k vyřešení a bude pasivně čekat na klienty (uzly), které si budou ze serveru vyzvedávat úkoly, vypracovávat je a pak serveru zašlou výsledky. Server nebude uzly sám od sebe nikdy kontaktovat.

Budeme předpokládat, že všechny počítače, které se systému účastní, jsou již propojeny počítačovou sítí s IP protokolem a samotný přenos dat po síti je spolehlivý. Nebudeme vymýšlet žádné nové komunikační protokoly, využijeme unicastovou komunikaci přes HTTP protokol se SSL šifrováním nad TCP. Teprve nad protokolem HTTP vybudujeme svůj vlastní protokol pro výměnu informací mezi serverem a uzly.

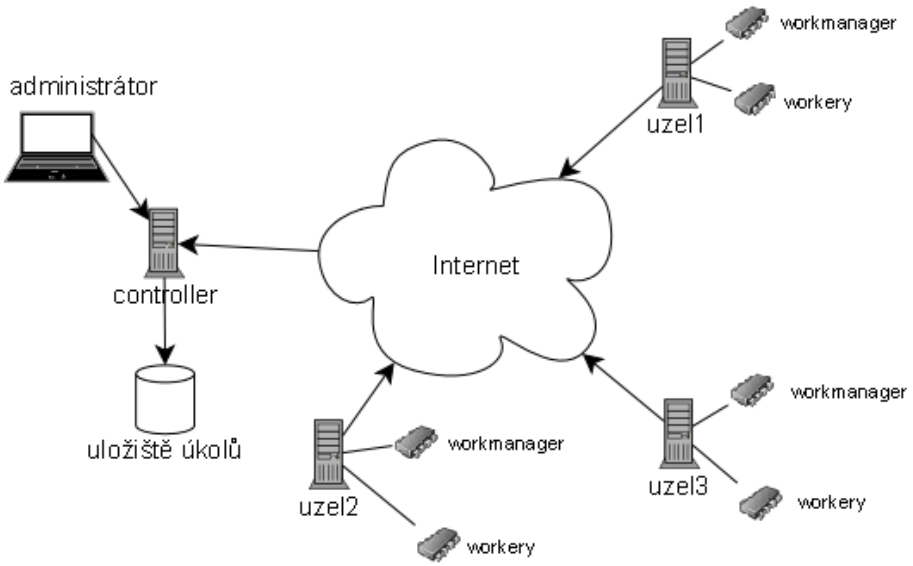
Je potřeba též podrobně vyřešit výměnu dat mezi serverovou a klientskou částí distribuovaného systému. Samozřejmě bude vhodné použít nějakou obdobu vzdáleného

volání procedur (RPC – Remote Procedure Call), které schová celou režii sítové komunikace a programátor dalších částí systému se tím nebude muset zabývat. Pro tyto účely využijeme vlastní RPC řešení – XAPI, popsané v předchozí části.

5.6.8 Implementace architektury

Zde implementovaný univerzální systém distribuovaného zpracování se dělí na následující komponenty:

- **controller** – Knihovna, která je součástí aplikace běžící na serveru. Eviduje úkoly, přijímá od klientů požadavky na předání úkolů a přijímá též od klientů výsledky úkolů, komunikuje s dalšími systémy na serveru (přijímá od nich úkoly a předává jim výsledky).
- **administrace** – Běží na serveru, slouží k ovládání systému administrátorem skrz controller (vytváření sad úkolů, jejich spouštění a správa). Administrace není přímou součástí controlleru, musí si ji vytvořit příslušná aplikace pro své potřeby a napojit se na controller.
- **workmanager** – Samostatný proces běžící v jedné instanci na straně uzlu ve formě asynchronního vlákna, komunikuje s controllerem na serveru, vyměňuje si s ním úkoly a výsledky, spravuje frontu úkolů čekajících na zpracování a výsledků čekajících na odeslání na server.
- **worker** – Samostatný proces běžící v libovolném počtu instancí na uzlu, od workermanageru si přebírá úkoly, zpracovává je a vrací výsledky.



Ilustrace 2: schéma architektury distribuovaného systému

Aplikací máme v tomto textu na mysli celý systému na serveru, který zadává controlleru úkoly, rozumí jejich obsahu, rozumí výsledkům a umí je interpretovat. V našem případě se tedy jedná o systém, který eviduje domény, předává provedení jejich analýzy controlleru a zpracovává výsledky těchto analýz. Tato kapitola pojednává pouze o controlleru a dále o součástech distribuovaného zpracování na uzlech.

Celý řešený problém (v našem případě analýza sady domén) se v controlleru nazývá **sada úkolů (suite)**. Sada úkolů se pak skládá z libovolného množství úkolů, což je rozkouskování celého problému na menší nezávislé části (v našem případě rozdělení všech domén po určitém počtu na menší skupiny). Každá sada úkolů má svůj typ, který říká, o jaký druh úkolů se jedná, resp. která knihovna (resp. část aplikace) rozumí obsahu úkolů. Pro účely této práce používáme typ úkolů „dnst“, což znamená DNS test domén. Podle tohoto typu controller zasílá dané úkoly jen takovým uzlům, které umí daný typ úkoly vypracovat, a výsledky úkolů předává příslušné knihovně. Controller se neomezuje na jeden typ úkolů, může současně řídit zpracování libovolného počtu sad úkolů libovolných typů. U úkolů však není možné stanovovat priority, jsou zpracovávány tak jak přijdou pod ruku controlleru (jak je vybere databázový server ze seznamu čekajících úkolů). Zavedení priorit by však nebyla komplikovaná modifikace. Pokud potřebujeme nějakou sadu úkolů vypracovat přednostně, můžeme ostatní sady pozastavit. Celá sada je vyřešena v případě, že jsou dokončeny všechny úkoly v této sadě.

Každá sada a každý úkol jsou identifikovány svým ID. Tato ID jsou unikátní v rámci celého controlleru.

Sada úkolů se může nacházet v následujících stavech:

- **new** – nově založená sada, může být zatím bez úkolů, administrátor ji teprve připravuje
- **processing** – úkoly dané sady se vykonávají, tj. buď čekají na serveru k odeslání na uzel či jsou zpracovávány na uzlech
- **finished** - všechny úkoly sady jsou dokončené
- **disabled** – vykonávání úkolů sady bylo pozastaveno, může být později opět spuštěno
- **cancelled** – vykonávání úkolů sady bylo zcela zrušeno

Každý z úkolů se může nacházet ve stavu:

- **new** – založen, ale sada ještě nebyla spuštěna
- **queued** – sada je spuštěna, úkol čeká ve frontě na serveru až bude odeslán na nějaký uzel
- **processing** – úkol byl odeslán na některý uzel a čeká se na jeho výsledky (na uzlu může buď čekat ve frontě na zpracování, může být skutečně vykonáván anebo čeká ve frontě výsledků na odeslání)
- **finished** – úkol byl dokončen, výsledky byly doručeny na server, kde byly předány aplikaci a nedošlo k žádnému problému
- **disabled** – úkol ještě nebyl odeslán a sada byla pozastavena
- **cancelled** – úkol ještě nebyl odeslán a sada byla zrušena

5.6.9 Server a controller

Controller je v podstatě knihovna, která organizuje celé distribuované zpracování. Zpřístupňuje zbytku celé aplikace na serveru rozhraní, kterým se předávají sady úkolů a úkoly ke zpracování a přes něj se zpět získávají informace o průběhu zpracování a výsledky úkolů. Controller používá databázi k evidenci těchto sad úkolů a jednotlivých úkolů a jejich stavu. Data úkolů a výsledky úkolů si v databázi ani nikde jinde u sebe neeviduje, dle potřeby je přebírá od aplikace a výsledky aplikaci ihned předává. Jedinou

výjimkou je fronta nových výsledků v souborovém systému, která slouží pouze jako dočasné úložiště mezi přijetím výsledku úkolu od uzlu a jeho předáním aplikaci.

Rozhraní controlleru nabízí aplikaci následující příkazy:

- vytvoření sady
- přidání úkolů do sady
- spuštění vykonávání sady – úkoly sady se přepnou do stavu „queued“ a mohou být odeslány na uzel
- pozastavení vykonávání sady
- znovuspuštění pozastavené sady
- zrušení sady

Naopak aplikace musí controlleru poskytnout rozhraní, kterým controller sděluje aplikaci:

- změnu stavu sady
- žádost o předání dat konkrétního úkolu
- úspěšné odeslání daných úkolů na nějaký uzel
- úspěšné dokončení úkolu (včetně předání dat výsledků)

Pokud vykonávání úkolů pozastavíme, neznamená to, že se jejich zpracování zastaví i na uzlech. Uzly se o tomto pozastavení nedozví. Znamená to pouze to, že server přestane předávat další čekající úkoly dané sady uzlům na jejich žádost, přijaté výsledky však běžným způsobem zpracuje. Podobně úplné zrušení vykonávání sady znamená, že server už nikdy další úkoly této sady nepředá k vyřízení, ale na uzlech dříve odeslané úkoly běžně doběhnou až do konce, avšak jejich výsledky budou controllerem ignorovány.

Na druhé straně controller poskytuje XAPI rozhraní, na kterém přijímá spojení od uzlů (resp. jejich workmanagerů). Rozhraní poskytuje 2 XAPI metody:

- požadavek na zaslání úkolů
- předání výsledků úkolů

Toto rozhraní samozřejmě může být v jeden okamžik paralelně voláno více uzly. Při předávání výsledků nemůže dojít k problémům při souběhu, protože každý úkol má své

unikátní označení a pod jeho názvem se ukládá do souborového systému (fronta responses). Problémem je však souběh při požadavcích na další úkoly – nesmí se stát, aby server předal stejné úkoly více uzlům, které přesně ve stejný okamžik zavolají controller s žádostí o další úkoly. To se řeší pomocí zámků na úrovni databáze.

Controller používá následující adresáře pro ukládání dat:

- **archive** – archiv dat výsledků úkolů, které již byly předány aplikaci, jeho obsah může být smazán
- **responses** – fronta výsledků úkolů, přijatých od uzlů, které čekají na předání aplikaci

Součástí controlleru jsou další podpůrné skripty. První skript zpracovává nově přijaté výsledky. Jak bylo již řečeno, jakmile server obdrží od uzlu nový výsledek úkolu, nezačne jej ihned zpracovávat, ale pouze jej vloží do fronty nových výsledků. Následně se každou minutu spouští skript, který bere nové výsledky z fronty a předává je aplikaci. Druhý skript slouží ke sledování stavu úkolů a je spouštěn každých 10 minut – restartuje úkoly, které byly odeslány ke zpracování na uzel, ale již vypršel timeout pro přijetí výsledků (úkol tak bude opět odeslán na nějaký uzel).

5.6.10 Workmanager

Workmanager je proces, který stále běží v jedné instanci na uzlu. Na jedné straně komunikuje po síti (přes XAPI) s controllerem, přijímá od něj nové úkoly a odesílá výsledky úkolů. Na druhé straně komunikuje s workery, kterým rozděluje úkoly a přijímá od nich výsledky.

Workmanager nepoužívá žádnou databázi, veškerá data se ukládají pouze v souborovém systému. Tato varianta byla zvolena z důvodu menší náročnosti na instalaci a údržbu uzlů. Workmanager workery nijak nevolá, komunikuje s nimi v podstatě pouze skrz souborový systém. Jsou stanoveny adresáře, v nichž workrmanager vytváří frontu úkolů a podobně z jiného adresáře načítá frontu výsledků od workerů.

Používá se následující adresářová struktura:

- **archive** – Archiv zadání úkolů, sem přesunou workery úkoly po svém dokončení. Obsah tohoto adresáře je možné mazat.
- **processing** – Adresář se soubory úkolů, které jsou momentálně zpracovávány. Sem si workery přesouvají soubory z adresáře requests.
- **requests** – Fronta úkolů čekajících na zpracování. Odtud si workery berou další práci.
- **responses** – Fronta výsledků úkolů. Do tohoto adresáře workery zapíší výsledek úkolu a workmanager si je odtud vyzvedává k odeslání na server. Po odeslání je soubor s daty výsledku smazán.

Životní cyklus workmanageru vypadá následujícím způsobem:

- Zkontrolovat množství úkolů ve frontě requests. Pokud množství kleslo pod stanovenou mez, stáhnout ze serveru další úkoly a vložit je do fronty (do adresáře requests).
- Zkontrolovat množství výsledků úkolů ve frontě responses. Pokud je fronta neprázdná, vzít všechny soubory (resp. dle nastaveného maxima v konfiguraci) a odeslat je na server. Odeslané výsledky smazat z fronty.
- Uspat se (na čas stanovený v konfiguraci). V tomto případě byla zvolena 1 minuta.

Úkoly se ze serveru stahují v určitých dávkách, jejichž velikost je opět stanovena v konfiguraci. Bylo by zbytečné stahovat úkoly po jednom, během jednoho spojení se jich stáhne více. Na druhou stranu není vhodné jich najednou stahovat příliš mnoho – počet by se měl odvíjet od schopností uzlu, tedy od jeho výkonu a počtu workerů, které na něm běží. Je také potřeba dát pozor na to, že příliš velikou dávku dat by nemusel proces workmanageru dokázat přijmout pro jeho paměťovou náročnost. Podobná omezení platí o maximálním počtu současně odesílaných výsledků. Při pokusech docházelo k havárii aplikace v okamžiku, kdy probíhala serializace či deserializace příliš velkých datových zpráv XAPI.

V XAPI požadavku na zaslání úkolů se předávají následující informace:

- číslo verze protokolu distribuované vrstvy (pro případnou budoucí koexistenci více verzí protokolů)
- ID workmanageru

- počet úkolů, které jsou požadovány
- capabilities – seznam typů úkolů, které uzel umí zpracovat (podle toho controller volí úkoly, které zašle)

Odpověď obsahuje:

- číslo verze protokolu
- počet zasílaných úkolů (údaj může být nulový, pokud není žádný úkol ke zpracování)
- pole s daty úkolů

V XAPI požadavku s předáním výsledků úkolů se předávají následující informace:

- číslo verze protokolu
- ID workmanageru
- počet úkolů, jejichž výsledky se zasílají
- pole s daty výsledků úkolů – součástí jsou také informace o ID workeru, který úkol vypracoval, čas trvání úkolu aj.

Odpověď obsahuje:

- číslo verze protokolu
- počet přijatých výsledků úkolů (slouží pro kontrolu, zda data dorazila v pořádku)

Aby nedošlo k souběhu při zapisování a čtení souborů, tedy aby worker nepřečetl soubor s úkolem, který ještě nebyl celý zapsán na disk, anebo aby workmanager nepřečetl soubor s výsledkem úkolu, který ještě worker nestihl celý zapsat na disk, používá se atomický zápis. Ten spočívá v tom, že se soubory na disk zapisují pod dočasnými jmény (která jsou v adresářích při čtení ignorována) a po dokončení zápisu se provede přejmenování na konečný název. Předpokládá se totiž, že operace přesunutí (resp. přejmenování) souboru je v rozumných souborových systémech atomická.

Pokud workmanager přijímá nebo odesílá data a během přenosu dojde k poruše (chyba na síti nebo havárie jedné z komunikujících stran), nesmí dojít ke ztrátě dat. Na straně controlleru jsou tedy čekající úkoly označené jako odeslané ke zpracování až po úspěšném potvrzení příjmu uzlem. Podobně na uzlu se soubor s výsledkem úkolu smaže

až poté, co se uzel ujistí, že controller data přijal. Při poruše se při nejbližší možné příležitosti přenos opakuje. Může se stát, že controller obdrží data výsledku vícekrát (aplikace musí být natolik chytrá, aby výsledky nezapočítala dvakrát) či controller předá jeden úkol více uzlům (je to škoda, mrháme tím procesorovým časem uzlů, ale chyba také nenastane).

5.6.11 Worker

Worker je samostatný proces běžící na uzlu. Na jednom uzlu může běžet libovolné množství workerů. Z toho důvodu je potřeba řešit možné souběhy (race condition), aby se např. nestalo, že si z fronty úkolů 2 workery ve stejný okamžik vezmou stejný úkol a oba jej budou vykonávat. Worker tuto situaci ošetřuje tak, že si soubor se zadáním úkolu z fronty (requests) přesune do adresáře běžících úkolů (processing) pod svým unikátním jménem a poté zkontroluje, zda soubor existuje. Jiné místo možné kolize workerů není. To vychází z již uvedeného předpokladu atomicity přesunutí souboru v rámci jednoho souborového systému. Od okamžiku vyzvednutí úkolu workerem mají všechny soubory spojené s tímto úkolem unikátní název odvozený od ID workmanageru a ID workeru.

Životní cyklus workeru vypadá následujícím způsobem:

- Pokud je fronta úkolů ke zpracování (requests) prázdná, uspat se na určitou dobu (stanovena v konfiguraci) a zkusit to znovu.
- Zvolit jeden úkol ve frontě a jeho soubor přesunout pod unikátním názvem do adresáře processing.
- Pokud soubor po přesunu neexistuje (což znamená, že jsme se o něj poprali s jiným workerem a nezískali jsme jej), vrátit se k předchozímu bodu a zkusit převzít další úkol.
- Načíst soubor s úkolem, zjistit z něj typ úkolů a podle toho jej předat příslušné knihovně, která umí daný typ úkolu vypracovat.
- Čekat na dokončení úkolu, převzat výsledek, zapsat jej do fronty responses.

5.6.12 Běh asynchronních vláken workeru na pozadí

V této aplikaci je tedy potřeba pro provedení automatizované hromadné analýzy několik asynchronních vláken, v nichž běží klienti distribuovaného systému a vykonávají úkoly, které dostávají. Zde se PHP skripty samozřejmě nepouští přes WWW server, ale přímo z příkazové řádky (tzv. CLI – command line interpreter). Abychom měli jistotu, že vlákna poběží stále a že v případě ukončení se vlákno zrestartuje, využívá se balíčku daemon-tools od D. J. Bernsteina (http://cr.yp.to/daemontools.html), což je sbírka nejrůznějších UNIXových systémových utilit. Pro nás je podstatná utilita supervise, která umožňuje spustit jakýkoliv program či skript jako služba na pozadí (odpojí se standardní vstup a výstup) a běh programu je monitorován. Jakmile se ukončí, je opět spuštěn.

Po instalaci daemontools (je potřeba je zkompilovat) stačí v adresáři /service vytvořit pro každý spouštěný skript podadresář a do něj vložit spustitelný soubor s názvem „run“, který je spouštěn a sledován. V něm pak můžeme pustit jakýkoliv další program, v našem případě se zde pouští PHP skript asynchronního vlákna. Konkrétně např. do adresáře /service/dns-worker-2-1 vložíme skript „run“ s tímto obsahem:

```
#!/bin/bash
cd /cesta_k_aplikaci/dns-info-node/scripts
exec su apache_shell -c "exec ./dw_worker.sh 2 1"
```

Skriptu dw_worker.sh se zadávají 2 parametry – ID workmanageru (tím se identifikuje uzel) a ID vlákna v rámci tohoto uzlu. Nyní stačí v adresáři /service vytvořit libovolné množství podobných podadresářů a tolik budeme mít vláken workeru. Použitím výše uvedeného software samozřejmě použití celé aplikace omezujeme na UNIXové systémy.

Je nutné myslet na to, že pokud změníme cokoliv ve zdrojových kódech aplikace či změníme konfiguraci, je potřeba všechna vlákna na uzlu restartovat, aby se ukončila a znovu spustila. K tomu slouží skript killall_worker.sh, která na všechny procesy workerů zavolá příkaz kill. Tím se ukončí a supervise je opět spustí. To ovšem také bohužel znamená, že přerušíme případné aktuálně zpracovávané úkoly a budou muset být od začátku provedeny znovu.

5.6.13 Závěr distribuovaného zpracování

Od tohoto systému samozřejmě nemůžeme chtít to, aby bylo možné mu zadat nějaký úkol a okamžitě se dozvědět výsledek. Na to nebude připraven – většina věcí je řešena asynchronně přes několik front požadavků a odpovědí. Když však budeme trpěliví a nebudeme na výsledky spěchat, dokáže nám systém zpracovat obrovské množství dat. Systém je také vhodný jen pro některé druhy úkolů – takové, které lze vzít včetně všech potřebných vstupních dat a samostatně dojít k celkovému výsledku. Navíc nesmí záležet na pořadí, v jakém se úkoly vykonají. Je třeba myslet na to, že úkol, který vložíme do systému, se nám již nemusí podařit vzít zpět a zrušit jeho zpracování. Administrátor může přímo ovládat pouze serverovou část, klienti běží samostatně a nelze jim zaslat příkaz k ukončení nějaké operace.

5.7 Implementace testů

Pro komunikaci s DNS servery se opět využívá balíček Net_DNS z knihovny PEAR. Problém je v tom, že tento balíček je již poměrně zastaralý a v současné době se nevyvíjí a nikdo se o něj nestará. Na jeho základní funkčnost to vliv nemá, jen nerozumí novým typům DNS záznamů. Knihovna musela být proto trochu upravena, musela jí být přidána podpora SPF záznamů, čehož se dosáhlo tím, že se SPF záznamy nastavily jako alias pro TXT záznamy, protože jejich formát je shodný. Bohužel se však nepodařilo rozchodit správné čtení DNSKEY záznamů, resp. Net_DNS má zřejmě obecně problémy se čtením velkých záznamů, což DNSKEY záznamy jsou. Naštěstí ani nebylo v plánu DNSKEY záznamy analyzovat či kontrolovat, a tak se lze spokojit s tím, že knihovna Net_DNS dokáže zjistit, zda doména nějaký DNSKEY záznam má či nikoliv. Nad Net_DNS je postavena nadstavba, která v souborovém cachuje výsledky některých DNS dotazů.

Třídy testů z bakalářské práce byly výrazně přepracovány a vylepšeny. Systém testů je nyní mnohem modulárnější. Některé testy byly také zobecněny – např. kontrola shody TTL určité skupiny záznamů, takto se kontrolují nyní NS, A, AAAA, MX i SRV záznamy, testy se liší pouze daty. Podobně byla zobecněna kontrola reverzních záznamů a kontrola duplicity záznamů.

V rámci tohoto projektu byly také vyvinuty knihovny HTTP klienta (net_http) a SMTP (net_smtp) klienta. Je sice pravda, že v PHP na to knihovny existují, avšak žádná z nich nenabízí možnost získávat podrobnosti o celé komunikaci a analyzovat ji (zkoumat přijímaná a odesílaná data, zjišťovat návratové kódy všech odpovědí apod.).

Přehled důležitých tříd:

- dns_parser – parser výstupu programu dig při AXFR stahování celých zón a též parser seznamu SK domén, který nabízí ke stažení SK-NIC
- dns_resolver – sbírka metod pro získávání záznamů určitých typů
- dns_resolver_auth – třída specializovaná na zjišťování autoritativních DNS serverů domén postupným dotazováním od kořene
- net_http – klient HTTP protokolu
- net_smtp – klient SMTP protokolu
- whois – klient pro komunikaci s WHOIS servery
- whois_net – zjišťování informací o autonomních systémech a podsítích podle IP adresy
- dw_controller – obecný controller distribuované vrstvy
- dw_workman – workmanager distribuované vrstvy
- dw_worker – worker distribuované vrstvy

Souborová cache se používá pro ukládání mnoha výsledků:

- odpovědi od WHOIS serverů (surové odpovědi)
- informace zjištěné o autonomních systémech a podsítích
- seznamy autoritativních DNS serverů k doménám – DNS servery domény se přece jen nemění tak často, navíc jsou domény, na které se ptáme neustále dokola (na DNS servery TLD)
- výsledek testu, zda je mailserver open relay

Bylo by možné tyto informace cachovat v paměti, ale přednost byla dána souborovému systému, protože se jedná o data delší platnosti (minuty, hodiny až týdny) a speciálně při hromadné analýze je těchto dat mnoho a zbytečně by zabíraly paměť, přestože není jisté, že výsledky znovu využijeme. Práce se souborovou cache přitom není pomalá, operační systém se postará o to, že soubory se také částečně cachují, a tak většina

výsledků z cache ve skutečnosti pochází ze souborové mezipaměti operačního systému a na disk se nepřistupuje. Navíc obsah cache přežije restart serveru. Abychom souborový systém příliš nezatížily, protože těchto souborů může vzniknout mnoho (stovky tisíc až miliony na uzlu pro hromadnou analýzu), jsou soubory rozdělovány do podadresářů podle počátečních znaků či podle částí IP adres (někde až do 3 úrovní podadresářů).

5.8 Databázové struktury

V implementaci se používají 2 na sobě zcela nezávislé databáze. První, zde pojmenovaná „dist_work“, slouží pro controller distribuovaného zpracování a evidují se v ní sady úkolů a úkoly.

Struktura tabulky dist_work.**dw_suites** (sady úkolů)	
ID	ID sady úkolů
name	popis sady úkolů
tasks_type	typ úloh – typ workeru, který úkolů rozumí a umí jej provést, v našem případě „dnst“
tasks_cnt	počet úkolů v této sadě
tasks_timeout	počet minut, do kdy musí být úkol uzlem vypracován a odeslán zpět, jinak bude považován za nevyřízený a předán ke zpracování znovu
tasks_queued	počet úkolů této sady, které čekají na odeslání na uzel
tasks_processing	počet úkolů této sady, které byly odeslány na nějaký uzel a očekává se jejich vykonání a zaslání výsledku
tasks_finished	počet úkolů této sady, které byly dokončeny
status	stav sady úkolů
start_date	datum a čas spuštění sady
disabled_date	datum a čas posledního pozastavení sady
cancelled_date	datum a čas zrušení sady
finished_date	datum a čas dokončení sady úkolů (přijetí posledního výsledku)

Struktura tabulky dist_work.**dw_tasks** (úkoly)	
ID	ID úkolu
suite_id	ID sady úkolů
type	typ úkolu (stejný jako u sady)
workman_id	ID workmanageru, který úkol vyřídil
worker_id	ID workeru, který úkol vyřídil
task_time	trvání zpracování úkolu (sekundy)
queued_date	datum a čas vložení úkoly do fronty k odeslání na uzel
start_date	datum a čas odeslání úkolu na uzel
disabled_date	datum a čas pozastavení úkolu (odpovídá pozastavení sady úkolů)
cancelled_date	datum a čas zrušení úkolu (odpovídá zrušení sady úkolů)
finished_date	datum a čas přijetí výsledku úkolu
starts_cnt	počet spuštění úkolu (inkrementuje se při jeho restartu, pokud controller neobdrží do timeoutu výsledek úkolu)
status	stav úkolu

Opět je třeba zdůraznit, že controller si neeviduje data úkolů, protože nezná jejich strukturu a obsah. K přepočítávání všech čítačů dochází při jakékoliv změně stavu sady úkolů či nějakého úkolu.

Druhá databáze, zde pojmenovaná „dnsinfo_dns“, eviduje testované domény a všechny zjištěné informace, které jsou následně používány k sestavování statistických výsledků. Obsahuje následující tabulky:

- batch_suites – sady testů domén, odpovídá sadě úkolů v distribuovaném zpracování
- batch_dom – seznam domén s podrobnostmi
- batch_a – zjištěné A záznamy u jednotlivých domén
- batch_aaaa – zjištěné AAAA záznamy u jednotlivých domén
- batch_as – zjištěné autonomní u DNS serverů jednotlivých domén
- batch_dom_ns – zjištěné autoritativní DNS servery u jednotlivých domén
- batch_mx – zjištěné MX záznamy u jednotlivých domén
- batch_net – zjištěné podsítě u DNS serverů jednotlivých domén
- batch_srv_sip – zjištěné SRV SIP záznamy u jednotlivých domén
- batch_codes – kódy výsledků testů u jednotlivých domén

Dále následují tabulky se sumárními daty. Ty se liší tím, že neobsahují sloupec odkazující na doménu, ale obsahují navíc sloupec pro uložení počtu.

- batch_a_sum
- batch_aaaa_sum
- batch_as_sum
- batch_mx_sum
- batch_net_sum
- batch_srv_sip_sum
- batch_codes_sum

Jak bylo již uvedeno, na uzlech se žádná databáze nevyužívá a vše se na nich odehrává pouze v paměti.

5.9 Zpracování výsledků

První skupina databázových tabulek s doménami obsahuje podrobné informace pro každou doménu. Zapisuje se do nich při přijetí výsledku úkolu. Tato podrobná data o každé doméně však poté potřebovat nebudeme, protože nás zajímají „sumární“ čísla. Tato sumární data si po dokončení testů všech domén v sadě sestavíme a uložíme do dalších tabulek. Po sestavení sumárních dat můžeme původní podrobná data zaarchivovat a smazat, statistiky již máme spočítané a navíc podrobná data zabírají mnoho prostoru.

Jakmile controller zjistí, že u nějaké sady úkolů byl předán poslední výsledek, oznámí aplikaci konec celé sady (aplikace si tento okamžik nemusí hlídat sama). Aplikace na základě toho provede sestavení sumárních výsledků.

Následující grafy a tabulky jsou na veřejných stránkách zveřejněny ke každé analýze:

- DNS servery
  - graf počtu DNS serverů u domény (dle nadřazeného)
  - žebříček názvů DNS serverů u domén
  - žebříček IP adres DNS serverů u domén
  - žebříček počtu názvů DNS serveru na IP adresu
  - výsledky testů DNS

- Autonomní systémy a podsítě
  - graf v kolika autonomních systémech mají domény své DNS
  - graf v kolika podsítích mají domény své DNS
  - žebříček autonomních systémů u DNS serverů
  - žebříček podsítí u DNS serverů
  - výsledky autonomních systémů a podsítí
- Údaje ze SOA záznamů
  - žebříček hodnot z položky MNAME
  - žebříček hodnot z položky RNAME
  - výsledky testů údajů ze SOA
- A a AAAA záznamy
  - graf počtu A záznamů bez „www“
  - graf počtu A záznamů s „www“
  - graf počtu AAAA záznamů bez „www“
  - graf počtu AAAA záznamů s „www“
  - žebříček A záznamů (s reverzními záznamy)
  - žebříček AAAA záznamů
  - výsledky testů A a AAAA záznamů
- MX záznamy a mailservery
  - graf počtu MX záznamů u domén
  - žebříček názvů z MX záznamů
  - žebříček IP adres názvů MX záznamů
  - výsledky testů MX záznamů a mailserverů
- SRV SIP záznamy
  - výsledky testů SRV SIP záznamů
- DNSSEC
  - výsledky testů DNSSEC záznamů
- souhrnné výsledky a zajímavosti

5.10 Automatizace testů

Celý systém byl, stejně jako v případě bakalářské práce, vyzkoušen na všech doménách druhé úrovně v zóně CZ. Pro tyto účely byl připraven skript, který se postará o stažení aktuálního obsahu zóny (pomocí AXFR z jednoho z autoritativních DNS serverů

domény CZ). Je však dát pozor na to, že tento soubor je poměrně objemný (několik stovek MB). Od analýzy z roku 2007 výrazně narostl, ale nikoliv proto, že narostl počet zaregistrovaných a delegovaných domén, ale zejména kvůli přítomnosti DNSKEY záznamů a elektronických podpisů každé sady NS záznamů a každé sady DNSKEY záznamů. Z výstupu AXFR se získá seznam všech domén, které obsahuje. Soubor se seznamem získaných domén se poté nahraje přes administraci k příslušné sadě a sada se může následně spustit. To jsou všechny úkony potřebné ke spuštění nových testů na aktuálním seznamu domén.

Poté, co se všechny úkoly dokončí, tj. otestují se všechny domény a zapíší se všechny výsledky do databáze na serveru, spustí se automaticky vytvoření sumárních výsledků.

Při pouštění testů ve velkém množství vláken na serveru je nutno myslet na jeden možný problém. Jelikož dochází k velmi intenzivní komunikaci, na uzlu se otevírá značné množství TCP a UDP spojení (při testování jedné domény je vzneseno několik desítek DNS dotazů). Pokud je přímo na uzlu anebo na jeho bráně do Internetu provozován stavový firewall, musí si tento firewall evidovat všechna odchozí spojení, aby korektně zpět propouštěl pakety s odpověďmi. Avšak i tyto stavové firewally mají své limity a dokáží v jednom okamžiku evidovat jen určitý počet spojení. Problém u UDP komunikace je v tom, že neobsahuje navazování a ukončení spojení, tudíž firewall nemůže vědět, zda UDP komunikace mezi dvojicí komunikujících stran již skončila anebo zda ještě mají přijít nějaká další data. U TCP komunikace to problém není, firewall rozezná pakety oznamující konec spojení, následně může být záznam ze seznamu spojení ihned smazán. U záznamu o UDP komunikaci je nutné počkat na vypršení timeoutu. Může se tedy snadno stát, že se stavová tabulka přeplní a firewall znemožní vytvářet další nová spojení.

V Linuxu je možné velikost stavová tabulky zjistit ze souboru:
/proc/sys/net/ipv4/netfilter/ip_conntrack_max

A aktuální počet záznamů ve stavové tabulce zjistíme ze souboru (tento údaj je vhodné během testování sledovat):
/proc/sys/net/ipv4/netfilter/ip_conntrack_count

5.11 Veřejné webové rozhraní

Veřejné rozhraní pro provádění testů jednotlivých domén vypadá stejně jako v bakalářské práci, jen přibyly výsledky nově vytvořených testů. Formulář, do něhož se vyplňuje název domény a zaškrtávají se testy, které se mají provést, je opět chráněn pomocí captcha obrázků. Je to sice trochu otravná metoda ochrany proti zneužití rozhraní, avšak použití rozhraní chceme co nejvíce omezit. Tím, že narostl počet prováděných testů a zejména přibyla komunikace s WWW a SMTP servery, narostla doba testování. Test může probíhat několik sekund až minut, uživatelé s tím musí počítat.

Celé veřejné rozhraní je přizpůsobeno pro více jazyků. Veškeré texty se nenachází přímo ve zdrojových kódech, ale v datových souborech, lze přidat libovolné množství jazyků. Izolováno bylo celkem více než 500 jazykových textů (všechny možné nápisy, názvy, popisky a krátké odstavce textu) Je použito kódování UTF-8, problémem tedy nejsou i některé exotičtější jazyky. Rozhraní na provádění testů jednotlivých domén a stránky s výsledky analýz byly přeloženy do angličtiny. Zbytek stránek, tj. obecné texty o DNS a komentáře k výsledkům analýz, přeložen nebyl.

Následují screenshoty veřejného rozhraní:

formulář pro zadání názvu domény a parametrů testů

Výsledky testu - dns-info.cz

Autoritativní DNS servery domény							
název serveru	TTL		IPv4 adresa	IPv4 glue	IPv6 adresa	sériové č.	reakce
ns.forpsi.net (pri)	1800	30m	81.2.194.130			2009090402	165 ms
ns.forpsi.cz	1800	30m	81.2.209.185	81.2.209.185		2009090402	24 ms
ns.forpsi.it	1800	30m	62.149.230.87			2009090402	64 ms

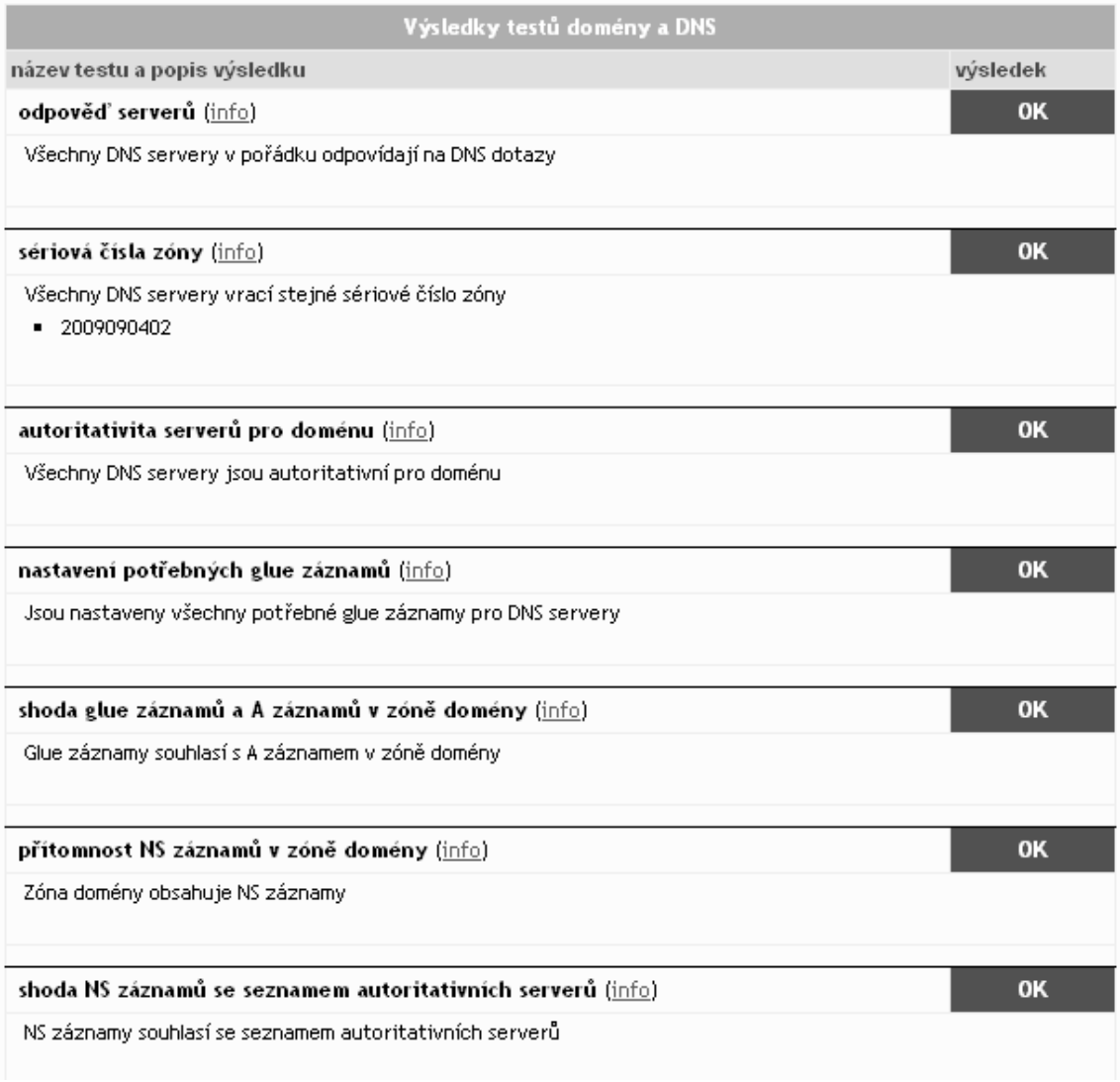
Podsítě a autonomní systémy DNS serverů			
název	IPv4 adresa	podsíť	ASN
ns.forpsi.net	81.2.194.130	81.2.194.0/23	24806
ns.forpsi.cz	81.2.209.185	81.2.208.0/22	24806
ns.forpsi.it	62.149.230.87	62.149.224.0/19	31034

SOA záznam (ns.forpsi.net)			
položka	hodnota		popis
serial	**2009090402**		sériové číslo zóny domény
mname	**ns.forpsi.net**		název primárního DNS serveru zóny
rname	**admin.forpsi.com**		kontakt na administrátora zóny
refresh	**3600**	1h	interval pro zjišťování aktualizací pro sekundární DNS servery (sekundy)
retry	**1800**	30m	interval pro opakovaní pokusu o zjištění změn v případě neúspěchu (sekundy)
expire	**2592000**	30d	doba, po jejímž uplynutí se stává zóna neplatnou, jestliže se ji nedaří aktualizovat z primárního DNS serveru (sekundy)
minimum	**3600**	1h	TTL pro negativní cachování

NS záznamy ze zóny (ns.forpsi.net)			
doména	TTL		hodnota
dns-info.cz	3600	1h	**ns.forpsi.net**
dns-info.cz	3600	1h	**ns.forpsi.cz**
dns-info.cz	3600	1h	**ns.forpsi.it**

MX záznamy ze zóny (ns.forpsi.net)					
doména	TTL		hodnota	IPv4 adresa	IPv6 adresa
dns-info.cz	1800	30m	**10 mx.wedos.com**	82.117.140.240	

základní informace o testované doméně (část)



Výsledky testů domény a DNS	
název testu a popis výsledku	**výsledek**
odpověď serverů (info) Všechny DNS servery v pořádku odpovídají na DNS dotazy	OK
sériová čísla zóny (info) Všechny DNS servery vrací stejné sériové číslo zóny ▪ 2009090402	OK
autoritativita serverů pro doménu (info) Všechny DNS servery jsou autoritativní pro doménu	OK
nastavení potřebných glue záznamů (info) Jsou nastaveny všechny potřebné glue záznamy pro DNS servery	OK
shoda glue záznamů a A záznamů v zóně domény (info) Glue záznamy souhlasí s A záznamem v zóně domény	OK
přítomnost NS záznamů v zóně domény (info) Zóna domény obsahuje NS záznamy	OK
shoda NS záznamů se seznamem autoritativních serverů (info) NS záznamy souhlasí se seznamem autoritativních serverů	OK

výpis výsledků jednotlivých testů domény (část)

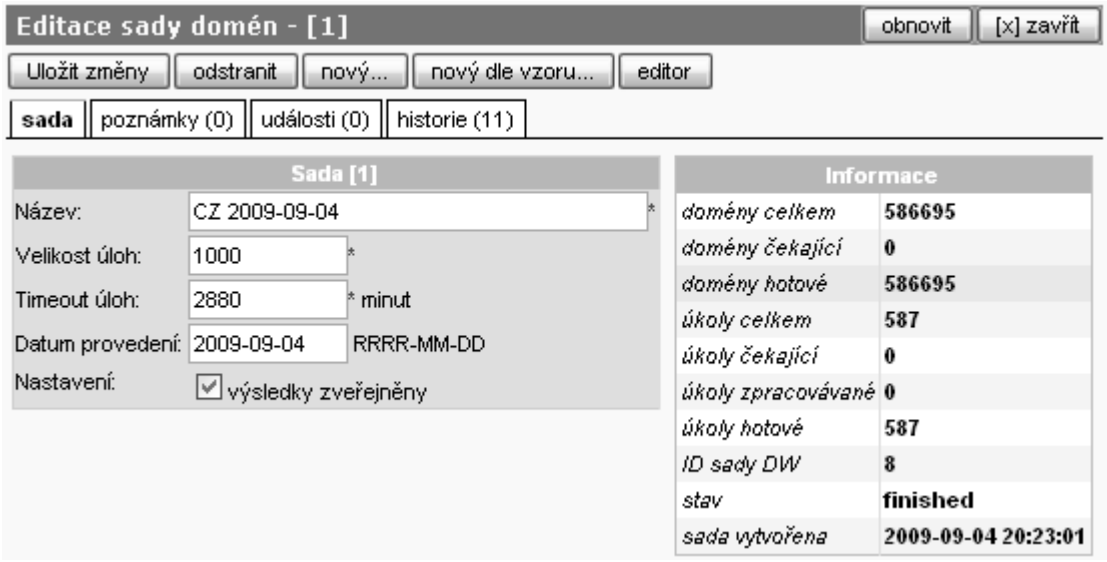
5.12 Administrační webové rozhraní

Administrační rozhraní nabízí jednoduchou správu sad domén. Obsahuje v podstatě 2 stránky – seznam sad a detail sady. Umožňuje vytvořit novou sadu, změnit její nastavení, nahrát seznam domén a ovládat zpracování v distribuovaném systému (spouštět, pozastavovat, zrušit). Seznam domén se k sadě nahrává uploadem textového souboru (jeden název domény na řádek). Tento soubor nám buď vygeneruje parser výstupu AXFR, parser seznamu SK domén anebo jej můžeme získat libovolným jiným způsobem.

Bylo by možné systém udělat tak, aby se analýzy iniciovaly zcela automatizovaně, např. aby se konkrétní den v každém měsíci spustil skript, který nějakým způsobem získá aktuální seznam domén, vytvoří novou sadu, nahraje seznam domén a analýzu spustí. Bylo to však raději uděláno tak, aby se toho musel účastnit administrátor a seznam domén nahrát přes administraci. Analýzy nemá smysl provádět velmi často, stačí jednou za měsíc. A je lepší, když to dělá administrátor a vše zkontroluje – aby se např. omylem nenahrál chybný seznam domén apod.

V detailu sady lze provádět několik málo nastavení a sledovat aktuální stav provádění analýzy. Tyto stavové informace se u sad aktualizují každých 10 minut.

Administrace nepodporuje více jazyků, je určena jen pro česky hovořícího správce. Následuje screenshot obrazovky s detailem sady:



náhled detailu sady domén v administraci

5.13 Požadavky na provoz aplikace

Webová aplikace potřebuje ke svému provozu:

- PHP ve verzi >= 5.2 s povolenou podporou soketů
- knihovnou GD s podporou freetype (pro generování captcha obrázků na veřejném rozhraní)

- balíček Net_DNS z http://pear.php.net/ s potřebnými modifikacemi (jsou součástí frameworku u aplikace)
- databázi MySQL ve verzi => 5.0 s podporou InnoDB
- webový server Apache s povoleným mod_rewrite (pokud je třeba provozovat aplikace ve více jazycích)

Pro distribuované analýzy navíc potřebujeme:

- balíček daemon-tools od D. J. Bernsteina pro běh skriptů PHP na pozadí jako asynchronní vlákna

Aplikace byla otestována na operačním systému Linux s webovým serverem Apache 2, PHP verze 5.2.9 a MySQL verze 5.0.75. Zda aplikace v pořádku běží na operačním systému Windows nebylo vyzkoušeno, určitě na ní však nebude fungovat distribuované prostředí (potřebovalo by nějaký mechanismus pro podporu běhu asynchronních vláken PHP skriptů).

Pro provádění hromadných analýz potřebujeme jeden nebo více serverů trvale připojených k Internetu. Uzly, provádějící samotné testy domén, nesmí být připojeny přes domácí připojení (jejich IP by byly blokovány na DNSBL serverech). Počet uzlů není nijak omezen.

Kapitola 6 – Analýza CZ a SK domén

6.1 Způsob provedení analýzy

Po dokončení vývoje této aplikace byl proveden test všech CZ domén druhé úrovně. Z něj vyplynuly nejen zajímavé informace získané díky novým testům, ale zejména bylo možné provést srovnání relativní četnosti zjištěných skutečností oproti testu, který byl vykonán v roce 2007 v rámci bakalářské práce. Nakonec byla neplánovaně provedena i analýza všech SK domén, protože to bylo vzhledem k úspěšné implementaci plně automatizované analýzy velmi jednoduché.

Při těchto nových analýzách však nebyly prováděny testy WWW serverů a mailserverů, a to z několika důvodů. Za prvé při pokusných hromadných testech bylo zjištěno, že komunikace s těmito servery zabírá velké množství času a vše trvalo nekonečně dlouho a není jisté, zda bychom se při testování všech 587 tisíci CZ domén dočkali v rozumném čase celkového výsledku. Druhý a závažnější problém nastal se SMTP servery. Sice se během testů ve skutečnosti neodešle jediný e-mail, protože veškerá SMTP komunikace skončí nejpozději příkazem RCPT TO a k příkazu DATA se nikdy nedostaneme, ale přesto SMTP servery monitorují všechnu komunikaci a pokusy o doručování e-mailů a po chvíli začaly zavádět opatření. Některé servery omezovaly maximální počet spojení za minutu, a tak testy u mnoha domén začaly selhávat. Pokud by to šlo takto dál, pravděpodobně by se IP adresy serverů, které byly použity k testům, objevily v některých blacklistech jako zdroj velkého počtu SMTP spojení a nebylo by snadné je následně z blacklistů odstranit. Navíc by byly výsledky hodně zkreslené. Tudíž celkový přehled stavu WWW a mailserverů českých domén se nyní nedozvíme. Musely nám tedy stačit informace získané z A, AAAA a MX záznamů.

Jako uzly byly použity 2 servery připojené k páteři Internetu, na nich testy běžely v celkem 50 vláknech. Servery by zvládly mnohem více vláken (násobky použitého počtu), ale protože nebyly vyhrazeny pouze pro potřeby těchto analýz, raději se vlákny šetřilo a jejich počet byl stanoven tak, aby jedna analýza běžela nejdéle 24 hodin (přibližná doba trvání analýzy byla stanovena provedením testů na zkušebním vzorku náhodně vybraných domén).

6.1.1 Domény CZ

Analýza byla provedena na všech CZ doménách, jejichž seznam byl získán ze zónového souboru domény CZ dne 4.9.2009. Domén bylo celkem 586695. Domény byly rozděleny do balíku po 1000, sada úkolů v distribuovaném zpracování tedy měla 587 úkolů. Oproti minulé analýze nebyla zjišťována dostupnost DNS serverů z více lokalit a nebyl zjišťován software DNS serverů.

Testování trvalo celkem 26 hodin a 20 minut. Součet časů práce všech vláken je 3151153 sekund, tj. 875 hodin. To znamená, že jedna doména se analyzovala v průměru 5,37 sekundy. Jeden úkol (1000 domén) na uzlech se zpracovával v průměru 89 minut. Mezi controllerem a uzly bylo přeneseno 53 MB dat úkolů (tj. seznamy domén) a 888 MB dat výsledků. To vše bylo samozřejmě komprimováno XAPI rozhraním až na 5% původní velikosti. Výkon serverů nehrál příliš roli, protože se většinu času analýzy domény čekalo. Běžný značkový server by zvládal stovky vláken.

6.1.2 Domény SK

Analýza byla následně provedena na všech SK doménách, jejich úplný seznam je volně k dispozici je stažení na stránkách správce SK-NIC (http://www.sk-nic.sk/). Podle publikovaných informací se jedná pouze o domény, které jsou generovány do zóny. Ostatní domény, které jsou sice zaregistrovány, ale jsou deaktivovány, by nás stejně nezajímaly. Seznam pocházel ze dne 12.9.2009, test byl proveden stejného dne. Domén bylo celkem 186611. Pro provedení jejich analýzy byl použit stejný postup se stejnými prostředky jako u CZ domén. Mimochodem je zajímavé, že někteří správci TLD seznam domén přísně stráží a jiní jej dají veřejně k dispozici na titulní stránce svých WWW včetně kódů majitelů.

Testování trvalo celkem 6 hodin a 10 minut. Součet časů práce všech vláken je 957251 sekund, tj. 266 hodin. To znamená, že jedna doména se analyzovala v průměru 5,13 sekundy. Jeden úkol (1000 domén) na uzlech se zpracovával v průměru 85 minut. Průměrné časy testů byly tedy velmi podobné s CZ doménami.

6.2 Výsledky analýzy CZ domén

6.2.1 Souhrn a zajímavosti

počet	*počet*	*%*
Celkem domén k otestování	586695	
Celkem otestovaných domén	585291	99,76
Nebylo možno otestovat (nelze získat žádné údaje o doméně)	1404	0,24
Počet unikátních DNS serverů dle názvu	17413	
Počet unikátních DNS serverů dle IP adresy	13398	
Průměrný počet domén na DNS serveru (dle IP adresy)	44	
Počet DNS serverů s IPv6	267	1,53
Počet domén na DNS serverech dostupných přes IPv6	34894	5,96
Počet DNS serverů (dle názvu), kterým nesouhlasí glue	204	1,17
Počet autonomních systémů	2000	
Počet podsítí	7784	
Suma NS záznamů na nadřazeném serveru	1353213	
Suma NS záznamů v zónách domén	1333460	
Suma MX záznamů v zónách domén	751558	
Suma A záznamů bez WWW v zónách domén	509468	
Suma A záznamů s WWW v zónách domén	541666	
Suma AAAA záznamů bez WWW v zónách domén	4307	
Suma AAAA záznamů s WWW v zónách domén	4415	
Suma SRV SIP záznamů v zónách domén	245	
Počet všech chyb	254575	
Počet všech varování	959721	
Počet všech upozornění	729040	
Počet domén s chybami	206710	35,32
Počet domén s varováním (a bez chyb)	320238	54,71
Počet domén s upozorněním (a bez chyb a varování)	22984	3,93
Počet domén bez jakéhokoliv problému	36763	6,28

Náročnými kritérii metodiky tedy prošlo pouze 6,28% domén. Je však zarážející, že třetina domén obsahuje nějaké chyby a více než polovina má nějaký problém ve formě varování.

DNS servery u domén podle názvu (TOP 10)			
DNS server	***IP adresa***	***domén***	***%***
ns.forpsi.net	81.2.194.130	91379	15,58
ns.forpsi.cz	81.2.209.185	90347	15,4
ns.forpsi.it	62.149.230.87	84494	14,4
alfa.ns.active24.cz	81.95.96.2	63499	10,82
beta.ns.active24.cz	81.31.37.213	63498	10,82
ns2.ignum.cz	213.235.133.138	47267	8,06
ns1.ignum.com	217.31.49.46	46236	7,88
ns1.regzone.cz	217.198.113.10	26320	4,49
ns1.regzone.info	77.93.209.246	26320	4,49
ns2.pipni.cz	81.0.235.34	15249	2,6

A záznamy u domén (TOP 10)			
IP	***reverzní záznam***	***domén***	***%***
81.2.194.128	128.194.forpsi.net	24813	4,23
81.95.96.29	default.domeny.cz	24449	4,17
62.168.63.249	?.czechia.cz	13202	2,25
217.31.49.238	mufflon.core.ignum.cz	11340	1,93
81.91.86.11	php5.web4u.cz	6300	1,07
193.86.238.12	www2.pipni.cz	6282	1,07
217.31.49.20	baghdad.core.ignum.cz	5088	0,87
216.8.179.24	ptr-216-8-179-24.ptr.nextdimensioninc.com	4873	0,83
81.2.194.176	c176wp.forpsi.com	3197	0,54
193.86.238.13	www3.pipni.cz	2871	0,49

IP adresy z prvních dvou řádků předchozí tabulky se používají pro „zaparkované“ domény bez vlastního obsahu. To znamená, že minimálně desetina zaregistrovaných a funkčních CZ domén je jen zaparkované a nejsou nijak využívány.

MX záznamy u domén (TOP 10)			
MX záznam	***IP***	***domén***	***%***
mail2.ignum.cz	82.117.159.67	28809	4,91
mxavas.forpsi.com	81.2.195.200	26717	4,55
mx.smtp.cz	81.31.37.214	22955	3,91
in.smtp.cz	81.95.97.102	22386	3,82
10mx.zoner.com	217.198.112.246	12414	2,12
15mx.zoner.com	82.208.28.74	12346	2,1
mail8.ignum.cz	217.31.49.69	12111	2,06
relay.zoner.com	217.198.112.217	11872	2,02
mx1b20.thinline.cz	82.208.47.3	9654	1,65
mx1d10.thinline.cz	88.146.119.14	9637	1,64

6.2.2 Komentář k výsledkům a porovnání s předchozí analýzou

Je překvapivé, u kolika domén se nepodařilo navázat spojení s žádným z DNS serverů (15792 – 2,69%). Možná jsme se trefili do doby, kdy měla nějaká skupina DNS serverů s velkým počtem domén poruchu, možná je to chyba našeho testu, ale vypadá to, že skutečně takové množství CZ domén má nefunkční DNS servery, protože podobný výsledek jsme získali i v analýze provedené před 2 roky a u SK domén se takový problém nevyskytl. Vypadá to tedy skutečně na problém u CZ domén. Ale raději nebudeme tento výsledek brát příliš vážně.

U některých údajů jsou vidět výrazné změny mezi současnou analýzou a analýzou provedenou v roce 2007. Překvapením je nárůst možnosti stažení celé zóny domén (AXFR) z 18% na 34,12%. Tudíž momentálně není problém stáhnout si všechny záznamy u více než třetiny CZ domén. Naopak výrazně klesl počet domén, jejichž DNS server lze využít k rekurzivním dotazům – z 31,15% na 14,06%. Žebříček používaných DNS serverů se v podstatě nezměnil. Ke zlepšení došlo u rozložení DNS serverů domén do více autonomních systémů. Původně mělo více než 1 AS 59,08% domén, nyní je to už 74,48%. Pořád je ale příliš mnoho domén (10,18%), které jsou závislé jen na jedné podsíti a oproti minulé analýze se to zlepšilo jen minimálně.

Z výsledků zkoumání A záznamů vyplývá, že většina domén (92%) je využívána k WWW službám. Ale AAAA záznamy jich má jen zanedbatelné množství (0,6%). E-

mailové služby poskytuje 76% domén. SPF informaci má u sebe 3,23% domén, ale všechny až na pár výjimek k tomu používají záznam typu TXT, které by se pro tyto účely již používat neměly, ale je otázka, zda tento typ podporují DNS servery (je to poměrně nová věc). Žádný z mailserverů není na žádném námi používaném DNSBL, žádný z mailserverů tudíž zřejmě není provozován přes žádné ADSL či jiné domácí připojení, které by bylo implicitně v alespoň jednom DNSBL blokováno. SRV SIP záznamy má 0,04% domén, většinou nad UDP protokolem anebo nad oběma (TCP i UDP). Žádná doména nepoužívá jen TCP.

DNSSEC podle této analýzy využívá 155 domén. Avšak podle tiskové zprávy CZ.NICu byla dne 4.8.2009 zavedena 1000. doména, kterou chrání technologie DNSSEC. Není příliš pravděpodobné, že bychom chybně získávali DNSKEY záznamy, a tak by byl náš počet nesprávný. Zřejmě to znamená, že mnoho domén má sice u registrátora domény aktivovánu podporu DNSSEC, ale ve skutečnosti ji používá jen 15% z nich. Třeba je příčina v tom, že majitelé domén chtěli tuto technologii použít, ale pak třeba zjistili, že jimi využívaný DNS server ji vůbec nepodporuje anebo zatím neměli čas či technické znalosti k nasazení této technologie v zóně své domény.

6.3 Výsledky analýzy SK domén

6.3.1 Souhrn a zajímavosti

počet	*počet*	*%*
Celkem domén k otestování	186611	
Celkem otestovaných domén	186311	99,84
Nebylo možno otestovat (nelze získat žádné údaje o doméně)	300	0,16
Počet unikátních DNS serverů dle názvu	11901	
Počet unikátních DNS serverů dle IP adresy	8516	
Průměrný počet domén na DNS serveru (dle IP adresy)	22	
Počet DNS serverů s IPv6	137	1,15
Počet domén na DNS serverech dostupných přes IPv6	1396	0,75
Počet DNS serverů (dle názvu), kterým nesouhlasí glue	111	0,93
Počet autonomních systémů	1358	
Počet podsítí	4650	
Suma NS záznamů na nadřazeném serveru	464757	
Suma NS záznamů v zónách domén	436103	
Suma MX záznamů v zónách domén	344226	
Suma A záznamů bez WWW v zónách domén	145710	
Suma A záznamů s WWW v zónách domén	166482	
Suma AAAA záznamů bez WWW v zónách domén	9872	
Suma AAAA záznamů s WWW v zónách domén	9878	
Suma SRV SIP záznamů v zónách domén	177	
Počet všech chyb	122662	
Počet všech varování	319356	
Počet všech upozornění	288387	
Počet domén s chybami	83363	44,74
Počet domén s varováním (a bez chyb)	71794	38,53
Počet domén s upozorněním (a bez chyb a varování)	26525	14,24
Počet domén bez jakéhokoliv problému	4929	2,65

Zde všemi kritérii prošlo pouhých 2,65% domén a chyb je relativně také více oproti CZ doménám.

DNS servery u domén podle názvu (TOP 10)			
DNS server	***IP adresa***	***domén***	***%***
sns.domains.sk	195.47.67.218	18750	10,05
dns.domains.sk	82.208.46.200	18749	10,05
ns1.nameserver.sk	212.89.225.134	14497	7,77
ns2.nameserver.sk	217.67.30.16	14483	7,76
ns1.dnsbackup.net	212.89.225.164	14395	7,71
ns2.dnsbackup.net	86.110.226.47	14361	7,7
ns.exohosting.sk	82.119.226.117	8372	4,49
ns1.exohosting.sk	92.240.234.73	8366	4,48
ns3.exohosting.sk	93.184.71.66	8187	4,39
ns2.telecom.sk	195.146.132.59	7595	4,07

A záznamy u domén (TOP 10)			
IP	***reverzní záznam***	***domén***	***%***
81.95.96.29	default.domeny.cz	4039	2,16
213.81.152.54	web.t-com.sk	3740	2
82.208.46.117	www.domains.sk	3233	1,73
82.119.226.53	max.websupport.sk	2011	1,08
217.67.22.85	mort.webglobe.sk	1794	0,96
217.67.30.192	web1.freeserver.sk	1784	0,96
82.119.226.57	colossus.websupport.sk	1708	0,92
195.28.64.101	web.slovanet.net	1592	0,85
81.89.56.37	ws2.ltc.sk	1531	0,82
81.89.56.32	ws1.ltc.sk	1484	0,8

MX záznamy u domén (TOP 10)			
MX záznam	***IP***	***domén***	***%***
sns.domains.sk	195.47.67.218	17507	9,38
mailin.mx-hub.sk	217.67.30.112	13514	7,24
mailin.mx-hub.net	217.67.30.112	13503	7,24
mailin.mx-hub.cz	217.67.30.112	13456	7,21
mailin.mx-hub.eu	217.67.30.112	13454	7,21
as.mx-hub.sk	217.67.30.96	12932	6,93
as.mx-hub.net	217.67.30.97	12928	6,93
relay.exohosting.sk	93.184.71.66	8174	4,38
relay1.exohosting.sk	92.240.234.67	8174	4,38
relay2.dnsserver.eu	82.119.226.101	8107	4,34

Nenechte se zmást na první pohled stejnou IP adresou mailserverů mailin.mx-hub.sk, mailin.mx-hub.net, mailin.mx-hub.cz, mailin.mx-hub.eu. Všechny tyto názvy mají 3 různé IP adresy z 2 různých autonomních systémů.

6.3.2 Komentář k výsledkům

Zdá se, že správce SK domény (SK-NIC) umožňuje u domén uvést pouze jeden DNS server, a více než 2% SK domén toho využívá. Je to docela překvapivé a těžko říct, proč je něco takového vůbec možné. Je také zajímavé, že u SK domény není možné mít více než 4 DNS servery, zatímco u CZ domén mají některé i 9 serverů. Použití jednoho DNS serveru (za předpokladu, že se za jeho názvem neschovává více A záznamů) samozřejmě implikuje závislost domény na funkčnosti jednoho autonomního systému a jedné podsítě. Nutnost použít alespoň 2 DNS servery u CZ domén alespoň lidi motivuje k tomu, aby jich skutečně použili více.

Je zvláštní, že u 5,5% domén (10379) není ani jeden z DNS serverů autoritativní a že to všem majitelům těchto domén nevadí. Anebo je takové velké množství domén jen tak zaregistrováno pro nějaké budoucí použití či ke spekulacím? Relativně mnoho domén je závislých na jedné podsíti (15,84 %), více autonomních systémů má 64,93 %.

SK domény mají relativně více SPF záznamů (7,23 %) než CZ domény (3,23 %) a dokonce mnohem více používají skutečné záznamy typu SPF (a nikoliv TXT), týká se to 548 SK domén. Opět nebyl žádný z mailserverů nalezen v DNSBL.

Překvapením je, že 57 domén má ve své zóně DNSKEY záznamy. Doména SK však DNSSEC zatím vůbec nepodporuje, a tak jsou k ničemu. Zřejmě se jedná jen o jakési experimenty jejich správců.

6.4 Závěr

Lze nalézt pár zajímavých relativních rozdílů mezi CZ a SK doménami. Výrazně více CZ domén má své DNS servery dostupné přes IPv6, ale přitom SK domény mají ve svých zónách mnohem více AAAA záznamů. CZ domény však obecně obsahují méně chyb než SK domény. A také výrazně více CZ domén (25,4%) nabízí získání všech záznamů domény (AXFR) oproti SK doménám (9,16%).

Domén SK je celkově výrazně méně, to je však dáno politikou SK-NICu – doménu si může zaregistrovat pouze osoba či firma, která žije či podniká na Slovensku (dnes již trochu zastaralé myšlení). Naopak CZ domény žádná podobná omezení nemají.

Kapitola 7 – Závěr

7.1 Splnění cíle

Zadání projektu bylo úspěšně splněno. Vznikl modernizovaný nástroj s novými testy a podporou automatizovaných distribuovaných analýz velkého souboru domén. Systém je velmi dobře škálovatelný a neměl by byt problém provést analýzu milionů domén.

Veřejné webové rozhraní nabízí velmi podrobné informace o jednotlivých doménách a je použitelný i pro laiky. Stránky této aplikace splňují normy XHTML 1.0 Transitional a CSS 2 a byly vyzkoušeny v množství internetových prohlížečů (Internet Explorer, Firefox, Opera aj.). Součástí je také administrační rozhraní pro ovládání hromadných testů. Závěrem byla provedena analýza všech CZ i SK domén.

Během vývoje této aplikace také vzniklo několik znovupoužitelných komponent:

- knihovny whois a whois_net pro komunikaci s WHOIS servery a zjišťování informací o IP adresách, podsítích a autonomních systémech
- kompletní framework pro distribuované zpracování úloh pro PHP, který byl již od počátku vymýšlen a implementován tak, aby byl zcela nezávislý na typu zpracovávaných úloh; tento framework se velmi osvědčil a je použitelný pro velkou škálu úloh, které mají podobné požadavky na distribuované zpracování
- knihovny klienta HTTP a SMTP protokolů

K diplomové práci patří CD, které obsahuje:

- zdrojové kódy a dalších součásti, potřebné ke zprovoznění testovacího jádra a webové aplikace pro testování jednotlivých domén včetně všech dodatečných skriptů (avšak bez klíče pro přístup k obsahu zóny CZ, který byl poskytnut organizací CZ.NIC a nelze jej dále distribuovat)
- návod k instalaci webové aplikace
- návod k zprovoznění distribuovaného prostředí a jeho administrace
- dokumentace zdrojových kódů, vygenerovaná nástrojem phpDocumentor
- tato diplomová práce ve formátu PDF

Funkční webová aplikace je k dispozici na stránkách, na kterých jsou také k nalezení všechny podrobné výsledky provedených analýz:
http://www.dns-info.cz/

7.2 Praktické zkušenosti s aplikací

Kdybychom chtěli testovat mnohem více domén, museli bychom přidat další vlákna a případně další uzly. Pokud by nás například napadlo analyzovat všechny domény pod TLD EU, která nyní má téměř 3 miliony domén, trvalo by nám to při stejném počtu vláken přibližně 90 hodin. Toto číslo je však skutečně pouze orientační, protože nelze odhadnout, jak na tom EU domény a jejich DNS servery jsou (jak rychle by odpovídaly, u které komunikace by nám vypršely timeouty apod.). Lze předpokládat, že při běhu analýzy na serverech v ČR by běžná komunikace s DNS servery EU domén byla o něco pomalejší, protože servery by byly v průměru „vzdálenější".

Při vyšším počtu domén a vláken by bylo potřeba zvýšit výkon serveru, na kterém běží controller, protože ten má hodně práce s evidováním všech domén a všech výsledků a to docela slušně vytěžuje databázi. Zdá se, že právě controller je potenciálně nejužším místem celého systému. Je tedy potřeba dát si pozor na to, aby stíhal zpracovávat výsledky (zapisovat je do databáze) a nehromadily se na něm ve frontě. Musí tedy držet tempo s uzly. Ale přesto to není tolik kritické. Předávání úkolů uzlům a stahování a ukládání výsledků do fronty na controlleru je totiž zcela nenáročné. Pokud tedy nebude controller stíhat, nebude mít problémy s obsluhováním uzlů, ale bude mít problém stíhat výsledky zapisovat do databáze. To nás ale zase tolik netrápí. Znamená to, že všechny testy doběhnou v obvyklém čase, ale pak se několik hodin nebo dní bude dokončovat jejich zpracování.

Původně se předpokládalo, že problémem bude objem komunikace mezi controllerem a uzly. Nakonec se ukázalo, že, jak bylo vypozorováno při analýze CZ domén, komprimace přenášených dat vrstvou XAPI sníží objem dat na 5%.

Problémem je však doba trvání testu jedné domény, pokud chceme o ní zjistit úplně všechno, tedy včetně komunikace s WWW servery a mailservery. To pak může zabrat mnoho sekundy, desítek sekundy a minut a v případě hromadné analýzy tudy cesta

nevede. Jedině že bychom měli opravdu velký počet uzlů a hodně času. Jenže analýza by neměla probíhat příliš dlouho, protože domény se stále mění nelze dobře zachytit jejich stav v jednom okamžiku.

7.3 Možnosti dalšího vývoje

Jedním z hlavních možných rozšíření, které se nepovedlo v této diplomové práci realizovat, je zkouška spojení na všechny zúčastněné servery (DNS, WWW i mailservery) přes IPv6 protokol. Přítomnost AAAA záznamu totiž ještě neznamená, že se nám spojení skutečně zdaří. Bohužel však nebyl k dispozici počítač, připojený do Internetu přes IPv6, na kterém by bylo možno provádět experimenty v rámci tohoto projektu. Dříve nebo později však musí tento systém podporu IPv6 zavést, aby byl použitelný do budoucna.

Dalším zajímavým rozšířením, o kterém se uvažovalo, je zkouška spojení na WWW a SMTP servery přes SSL, a pokud by se spojení zdařilo, provést analýzu certifikátu serveru. Šlo by např. o zjištění vydavatele certifikátu, parametry zabezpečení, detekce self-signed certifikátů, kontrola doby platnosti aj. Bohužel se nepodařilo najít způsob, jak v PHP za pomoci knihovny OpenSSL získat certifikát ze serveru z otevřeného TCP spojení (přitom samotné OpenSSL to pomocí několika příkazů podporuje). Komunikace přes SSL by však přinesla mnoho další komunikační režie a spotřeby procesorového času. A možná také mnoho čekání na timeout u serverů, které SSL nepodporují a nemají ani otevřené příslušné TCP porty.

Dále je možné provádět analýzy obsahu WWW stránek domén. Jedná se např. o detekování různých výchozích stránek nenakonfigurovaných WWW serverů, detekci škodlivého kódu ve WWW stránce (podobně jako to činí Google a hledá ve stránkách vzorky známých podvržených JavaScriptů). Tam je však otázka, jak takové vzorky k hledání získat. U WWW serverů by šla podle dalších hlaviček a obsahu rozšířit detekce použitého softwaru (např. PHP při použití své knihovny session vkládá do hlavičky cookie s klíčem PHPSESSID, podobně se dá detekovat ASP.NET).

Co se týká mailserverů, podobným způsobem by bylo možné zjišťovat i jejich software. Servery, které přímo neprozradí své jméno a verzi, často mívají typické texty, které

doprovází kódy odpovědí. A dále by bylo dobré z MX záznamů zjišťovat sítě a autonomní systémy, v nichž se mailservery nachází, a doporučovat, aby doména měla více mailserverů ve více různých sítích, nebo ještě lépe v různých AS.

Distribuované zpracování by si také zasloužilo další optimalizace. Současný problém spočívá v tom, že controller nemá žádnou průběžnou zpětnou vazbu od uzlů, takže nemá tušení, co se po odeslání úkolu na uzel děje. Neví to controller a ani člověk, který daný systém ovládá (jedinou možností je sledovat přímo logy na uzlech). Dobré by bylo zpětnou vazbu implementovat, aby uzel v nějakých krátkých intervalech informoval controller o tom, jaké úkoly právě zpracovává, z jaké části jsou hotové a kdy se předpokládá jejich dokončení, které úkoly selhaly a musely být na uzlu restartovány apod. Podobně je problém v situaci, kdy se rozhodneme celou již zpracovávanou sadu úkolů zrušit. Nyní neexistuje mechanismus jak uzlům říci, že mají úkoly z této sady přestat zpracovávat a zahodit. Uzel by se tedy mohl controlleru průběžně dotazovat, zda k nějaké takové situaci nedošlo.

Vyvinuté řešení distribuovaného zpracování počítá s tím, že všechny zúčastněné uzly jsou hodné a že je má pod správou stejná osoba či organizace. Pokud by se měly do systému zapojit další servery třetích stran, bylo by nutné se zabývat důvěryhodností výsledků.

Zajímavé by bylo rozšířit test DNSKEY záznamů o jejich kontrolu, tj. načíst jejich obsah a příslušný podpis v nadřazené zóně a tento podpis zkontrolovat. Bude se totiž zřejmě běžně stávat, že klíče u domény budou neplatné, protože správci domén musí klíče pravidelně obměňovat (nejsou platné do nekonečna) a třeba na to zapomenou nebo se jim to z nějakého důvodu nezdaří. Výměna klíče totiž znamená vykomunikovat tuto změnu se správcem nadřazené zóny. Neobnovení klíčů by ale znamenalo, že kontrola DNSSEC záznamů selhala a DNS servery by mohly prohlásit všechny záznamy za neplatné.

Literatura a další zdroje

[1] Mockapetris, P. (1987): *Domain Names - Concepts and Facilities*, STD 13, RFC 1034, WWW http://rfc.net/rfc1034.html

[2] František Gajdůšek: Webhosting s podporou IPv6, WWW http://gajdusek.net/2008/07/webhosting-s-podporou-ipv6/

[3] Crocker, D. (1982): *Standard for the Format of ARPA Internet Text Messages*, RFC 822, WWW http://www.faqs.org/rfcs/rfc822.html

[4] Klensin, J. (2008): *Simple Mail Transfer Protocol*, RFC 5321, WWW http://www.faqs.org/rfcs/rfc5321.html

[5] Wikipedia: *Session Initiation Protocol*, WWW http://cs.wikipedia.org/wiki/Session_Initiation_Protocol

[6] Rosenberg, J. a kolektiv (2002): *SIP: Session Initiation Protocol*, RFC 3261, WWW http://tools.ietf.org/html/rfc3261

[7] Gulbrandsen, A., Vixie, P., Esibov, L. (2000): *A DNS RR for specifying the location of services (DNS SRV)*, RFC 2782, WWW http://www.ietf.org/rfc/rfc2782.txt

[8] Wikipedia: *Comparison of DNS blacklists*, WWW http://en.wikipedia.org/wiki/Comparison_of_DNS_blacklists

[9] Wong, M., Schlitt, W. (2006): *Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1*, RFC 4408, WWW http://tools.ietf.org/html/rfc4408

[10] Organizace IANA (2002): *Special-Use IPv4 Addresses*, RFC 3330, WWW http://tools.ietf.org/html/rfc3330

[11] Elz, R., Bush, R. (1997): *Clarifications to the DNS Specification*, RFC 2181, WWW http://tools.ietf.org/html/rfc2181

[12] Wikipedia: *Honeypot (computing)*, WWW http://en.wikipedia.org/wiki/Honeypot_(computing)

[13] Fielding, F. a kolektiv (1999): *Hypertext Transfer Protocol – HTTP/1.1*, RFC 2616, WWW http://www.ietf.org/rfc/rfc2616.txt

[14] Postel, Jonathan B. (1982): *Simple Mail Transfer Protocol*, RFC 821, WWW http://www.ietf.org/rfc/rfc821.txt

[15] Klensin, J. a kolektiv (1994): *SMTP Service Extension for 8bit-MIMEtransport*, RFC 1652, WWW http://www.ietf.org/rfc/rfc1652.txt

[16] Klensin, J. a kolektiv (1995): *SMTP Service Extension for Message Size Declaration*, RFC 1870, WWW http://www.ietf.org/rfc/rfc1870.txt

[17] Klensin, J. a kolektiv (1995): *SMTP Service Extensions*, RFC 1869, WWW http://www.ietf.org/rfc/rfc1869.txt

[18] Vaudreuil, G. (2000): *SMTP Service Extensions for Transmission of Large and Binary MIME Messages*, RFC 3030, WWW http://www.ietf.org/rfc/rfc3030.txt

[19] Atkins, D., Austein, R. (2004): *Threat Analysis of the Domain Name System (DNS)*, RFC 3833, WWW http://www.ietf.org/rfc/rfc3833.txt

[20] Crockford, D. (2006): *The application/json Media Type for JavaScript Object Notation (JSON)*, RFC 4627, WWW http://www.ietf.org/rfc/rfc4627.txt

[21] Arends, D., Austein, R., Larson, M., Massey, D., Rose, S. (2005): *DNS Security Introduction and Requirements*, RFC 4033, WWW http://www.ietf.org/rfc/rfc4033.txt

[22] Arends, D., Austein, R., Larson, M., Massey, D., Rose, S. (2005): *Resource Records for the DNS Security Extensions*, RFC 4034, WWW http://www.ietf.org/rfc/rfc4034.txt

[23] Arends, D., Austein, R., Larson, M., Massey, D., Rose, S. (2005): *Protocol Modifications for the DNS Security Extensions*, RFC 4035, WWW http://www.ietf.org/rfc/rfc4035.txt

[24] Partridge C. (1986): *Mail Rounting and the Domain System*, RFC 974, WWW http://www.ietf.org/rfc/rfc974.txt