České vysoké učení technické v Praze
Fakulta elektrotechnická

**Katedra telekomunikační techniky**
**Obor: Komunikační sítě a internet**

# DNS – bezpečné řešení pro Cloud

# Safe DNS System for Cloud

DIPLOMOVÁ PRÁCE

Vypracoval: Bc. Jan Zlevor
Vedoucí práce: Ing. Ján Kučerák
Rok: 2023

# ZADÁNÍ DIPLOMOVÉ PRÁCE

## I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Zlevor** Jméno: **Jan** Osobní číslo: **483696**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra telekomunikační techniky**

Studijní program: **Elektronika a komunikace**

Specializace: **Komunikační sítě a internet**

## II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

**DNS – bezpečné řešení pro Cloud**

Název diplomové práce anglicky:

**Safe DNS System for Cloud**

Pokyny pro vypracování:

Navrhněte a realizujte bezpečné řešení DNS (Domain Name Systém) v prostředí datového centra a jeho cloudových systémů. Popište problematiku fungování DNS, současné bezpečnostní hrozby, ochranu a nepoužívanější techniky včetně technologií specializovaných výrobců. Proveďte analýzu současných nástrojů a postupů pro detekci a prevenci bezpečnostních incidentů v DNS. Připravte vlastní řešení použitím vybraného nástroje (Bind, Knox, Microsoft). Implementujte bezpečnostní opatření jako DNSSEC, alternativní resolvery a ochranu na síťovém perimetru. Řešení navrhněte s ohledem na možnost automatizace změn konfigurace (mysql, api, jenkins). Otestujte řešení a navržená opatření na vybraném vzorku DNS serverů a zhodnoťte efektivitu.

Seznam doporučené literatury:

[1] ROSE, S.; CHANDRAMOULI, P.: Secure Domain Name System (DNS) Deployment Guide, 2013.
[2] BIND 9 Administrator Reference Manual. Dostupné na https://bind9.readthedocs.io/en/latest/ [on-line]
[3] HOFFMANN, P.: DNS Security Extensions (DNSSEC)

Jméno a pracoviště vedoucí(ho) diplomové práce:

**Ing. Ján Kučerák katedra telekomunikační techniky FEL**

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **20.02.2023** Termín odevzdání diplomové práce: **09.01.2024**

Platnost zadání diplomové práce: **16.02.2025**

| Ing. Ján Kučerák<br>podpis vedoucí(ho) práce | podpis vedoucí(ho) ústavu/katedry | prof. Mgr. Petr Páta, Ph.D.<br>podpis děkana(ky) |
|---|---|---|

## III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. Datum převzetí zadání

Podpis studenta

**Prohlášení**

S čistým svědomím prohlašuji, že svou diplomovou práci jsem vypracoval samostatně pouze s podklady, uvedenými v seznamu zdrojů.

V Praze dne .....  .....

Bc. Jan Zlevor

**Poděkování**

Děkuji za vedení této práce a projevenou trpělivost vedoucímu Ing. Jánu Kučerákovi.

Bc. Jan Zlevor

*Název práce:*
**DNS – bezpečné řešení pro Cloud**

*Autor:* Bc. Jan Zlevor

*Studijní program:* Elektronika a komunikace
*Obor:* Komunikační sítě a internet
*Druh práce:* Diplomová práce

*Vedoucí práce:* Ing. Ján Kučerák
Katedra telekomunikační techniky
*Konzultant:* Ing. David Pech
Algotech

*Abstrakt:* Tato práce se zaměřuje na popis klíčových principů DNS (Domain Name System) a jeho bezpečnostních rozšíření DNSSEC (DNS Security Extensions). Předmětem jsou také současná řešení DNS dostupná buď jako open-source, nebo komerční produkty s bezpečnostní nadstavbou. Jádrem práce je návrh topologie pro autoritativní část DNS a návrh nástroje pro správu DNS zón ve spolupráci se společností Algotech. Velký prostor je dán implementačním záležitostem nástroje, jako například složení databáze, vytvoření rozhraní v jazyce Go a dalsí. Přednostmi nástroje je především implementace DNSSEC a snadná správa zón prostřednictvím buď REST API, nebo webového prostředí. Vyvíjený nástroj je v práci také testován a v závěru zhodnocena jeho funkčnost.
*Klíčová slova:* DNS, DNSSEC, Bezpečnost, Cloud, BIND

*Title:*
**Safe DNS System for Cloud**

*Author:* Bc. Jan Zlevor

*Abstract:* This thesis focuses on describing the key principles of Domain Name System (DNS) and its security extensions (DNS Security Extensions). The subject also covers current DNS solutions available as either open-source or commercial products with security extensions. The core of the work is the design of a topology for the authoritative part of DNS and the design of a DNS zone management tool in collaboration with Algotech. A large amount of space is given to implementation issues of the tool, such as database composition, creation of a Go interface and others. The main advantages of the tool are the implementation of DNSSEC and the ease of zone management through either the REST API or the web environment. The developed tool is also tested in this thesis and finally its functionality is evaluated.
*Key words:* DNS, DNSSEC, Security, Cloud, BIND

# Obsah

# Seznam použitých zkratek

| Zkratka | význam v anglickém jazyce |
|---|---|
| DNS | Domain Name System |
| DNSSEC | Domain Name System Security Extensions |
| DN | Domain Name |
| NS | Name Server |
| FQDN | Fully Qualified Domain Name |
| TLD | Top Level Domain |
| ccTLD | Country Code Top Level Domain |
| gTLD | generic Top Level Domain |
| RFC | Request for Comments |
| DHCP | Dynamic Host Configuration Protocol |
| DoH | DNS over HTTPS |
| DoT | DNS over TLS |
| HTTP | HyperText Transfer Protocol |
| IP | Internet Protocol |
| TCP | Transmission Control Protocol |
| API | Application Programming Interface |
| REST | Representational State Transfer |
| SQL | Structured Query Language |
| URL | Uniform Resource Locator |

# Seznam obrázků

# Úvod

Běžný uživatel, přistupující přes prohlížeč na webovou stránku, si možná nepřipouští, že zadáním adresy webu v prohlížeči spouští obsáhlý řetězec událostí, který vyústí v načtení správné stránky (v tom lepším případě). Důležitou částí nejen tohoto řetězce je překlad části URL – doménového jména na adresu IP, přes kterou je možné s daným serverem navázat spojení. Tento překlad zajišťuje systém DNS (Domain Name System), který tvoří rozsáhlá, robustní distribuovaná celosvětová infrastruktura. DNS představuje kritickou součást dnešního Internetu a s trochou nadsázky lze konstatovat, že náhlým výpadkem DNS by došlo k Internetovému „blackoutu“. Ač je toto konstatování nadsazené, podobné případy, kdy došlo ve větší či menší míře k omezení funkce DNS a tím pádem k omezení dostupnosti některých služeb, minulost zaznamenala. Svojí důležitostí je tak DNS atraktivním cílem kybernetických útoků ([1], [2], [3]).

DNS, jako distribuovaný systém, je tvořeno velkým počtem serverů, které na sebe odkazují ve stromové hierarchii. Svůj DNS server si pro vlastní doménu může zařídit prakticky kdokoli a tak už je vlastní DNS často součástí firemních sítí, ať už je jejich podnikání jakékoli. S každým systémem spadajícím do vlastní správy se však váží především bezpečnostní problémy a nejinak je tomu i u vlastního DNS. Je-li předmětem podnikání poskytování IT služeb (hosting webových stránek, cloud), tak bezpečnostní nároky na DNS rostou, ježto společnost má přímou odpovědnost nejenom za DNS záznamy svojí domény, ale i za údaje, které poskytuje (např. o hostovaných doménách).

Tento text si klade za cíl, představit princip DNS na základě současně platných (2023) RFC s bezpečnostními aspekty s tím spojenými, popsat architekturu praktických implementací DNS, analyzovat nástroje užitečné k implementaci DNS infrastruktury a v neposlední řadě vytvořit vlastní řešení založené na provedené analýze. Výsledné řešení pak otestovat v prostředí cloudového poskytovatele, v tomto případě, ve společnosti Algotech a. s..

# Kapitola 1

# DNS

Tato kapitola se zabývá popisem principů DNS. Popsány jsou základy doménových jmen, jmenných serverů a DNS protokolu. Dále je také popsáno nejčastější uskupení serverů v reálné implementaci DNS. Všechny informace jsou široce dostupné, ucelený přehled lze, i přes své stáří, najít také v [4], [5].

## 1.1 Stručná historie DNS

K pochopení toho, jak dnešní DNS vypadá, je účelné se ohlédnout do minulosti a podívat se, jakým vývojem tento systém prošel.

Už v dobách ARPAnetu, předchůdce dnešního Internetu, kdy počet do něj připojených počítačů byl v řádu desítek [6], se jevilo jako vhodné, aby se uživatelé odkazovali na uzly v síti pomocí jmen, která budou lidsky čitelná, nikoli pomocí číselných identifikátorů (adres), pomocí kterých uzly fakticky komunikují. V návaznosti na výskyt této potřeby vydává Peggy Karp roku 1971 RFC 226 *„STANDARDIZATION OF HOST MNEUMONICS“* [7], ve kterém představuje soupis dvaceti jmen pro počítače v ARPAnetu. Následně specifikuje způsob pojmenování uzlů v RFC 247 *„Proferred Set of Standard Host Names“* [8]. Příští rok (1972) je pak poprvé publikován soubor *hosts.txt* na veřejném FTP serveru institutu SRI (*Stanford Research Institute*). Tento jednoduchý textový soubor uchovával jméno počítače a jeho adresu řádek po řádku. Klienti, kteří chtěli komunikovat pomocí jmen, si tak museli opatřit kopii souboru *hosts.txt*. Ke komunikaci s některým ze síťových uzlů, pokud byl uveden v *hosts.txt*, se jeho adresa získala prostým nahlédnutím do souboru (v rámci programu). Jednoduchost tohoto systému sebou přináší zřejmé nevýhody. Systém není dobře škálovatelný. Chtěl-li někdo zaregistrovat nové jméno, musel prostřednictvím emailu kontaktovat se svojí žádostí způsobilou osobu v SRI. Když bylo jeho žádosti vyhověno, tak si museli všichni účastníci stáhnout celý soubor *hosts.txt* znovu. Dále nikterak nebyly řešeny jakéko-

liv bezpečnostní otázky, jež nabyly na důležitosti s pozdějším masovým nasazením Internetu. Zkrátka bylo zřejmé, že takový systém se v potenciálně globální síti nemohl udržet a bylo nutné hledat komplexnější řešení, které by vyvstalé problémy řešilo. Odkaz minulosti lze dnes spatřit v existenci souboru */etc/hosts* v různých UNIXových distribucích.

V průběhu času, jak síť rostla, začaly být problémy s údržbou a distribucí souboru *hosts.txt* markantnější a hledaly se cesty, jak vytvořit systém s lepšími vlastnostmi. V roce 1981 (přes 400 jmen a „přezdívek“ v *hosts.txt*) představuje David Mills v RFC 799 *„Internet Name Domains“* [9] myšlenku doménových jmen jako hierarchické struktury. V síti se pak nacházeli jistí prostředníci (forwarders), kteří přeposílali zprávy v síti podle doménových jmen v nich obsažených. Průlomem, dá-li se to tak říct, byla publikace RFC 882 *„DOMAIN NAMES - CONCEPTS and FACILITIES“* [10] a RFC 883 *„DOMAIN NAMES - IMPLEMENTATION and SPECIFICATION“* [11] (v roce 1987 nahrazeny RFC 1034 [12] a RFC 1035 [13] se shodnými názvy) od Paula Mockapetrise. Zde se již hovoří o soustavě jmenných serverů (name servers), kde každý nese zónové soubory (zonefiles). Zónové soubory jsou takové „miniatury“ souboru *hosts.txt* v tom smyslu, že nesou záznamy o jménech a adresách jenom z určité podmnožiny prostoru doménových jmen (*hosts.txt* pojímal celý jmenný prostor). V těchto dokumentech Mockapetris předkládá tvar a hierarchii doménových jmen a dva stěžejní pojmy, na kterých DNS mimo jiné stojí: autorita a delegace (viz 1.2.2).

Z dalšího vývoje stojí za připomínku mimo jiné zavedení bezpečnostních opatření v podobě DNSSEC (specifikace z roku 2005), přinášející zásadní bezpečnostní opatření, zabraňující útokům v podobě podvržení DNS záznamů.

Detailnější soupis historie DNS lze nalézt ve spisku *One History of DNS* [14].

## 1.2 Princip DNS

**Upozornění**

Domény uvedené v následujícím textu mají pouze ilustrační charakter. Buď vůbec k datu 1.1.2024 neexistovaly, nebo mají s obsahem práce přímou souvislost, nebo se nejedná o domény společností s komerčnímy záměry. V žádném případě se nejedná o skrytou propagaci služeb s danými doménami potenciálně spojenými.

Dle RFC 1034 [12] stojí systém doménových jmen na třech hlavních částech:

- Prostor doménových jmen
- Jmenné servery

- Resolvery

První zmíněná část – prostor doménových jmen je logický prostor, utvářející stromovou strukturu, ve které má každý uzel své jméno. Každý uzel reprezentuje buď nějaký koncový bod v síti (třeba webový server) nebo skupinu takových bodů.

Druhou částí jsou jmenné servery. Ty primárně slouží k držení databáze doménových jmen a jim přidružených adres, dále pak odpovídají na dotazy, které jsou na ně kladeny.

Poslední částí jsou resolvery což jsou programy sloužící k dotazování jmenných serverů, primárně pak k překladu doménových jmen.[1]

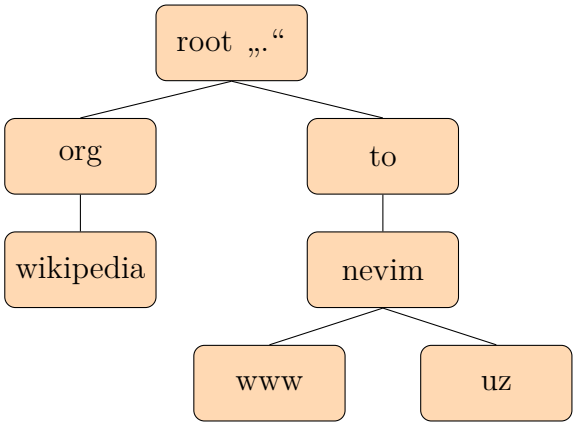
### 1.2.1 Prostor doménových jmen

Prostor doménových jmen, jak již bylo zmíněno, je logická struktura, tvořící stromovou hierarchii. Každý uzel je charakterizován jménem (*label*) složeným z písmen a-z, na jejichž velikosti nezáleží (case insensitive), z číslic 0-9 a z pomlčky -. Každý uzel je zároveň kořenem podstromu – domény. Pokud je dána nějaká doména s kořenem A a jiná s kořenem B, kde uzel B je potomkem A, pak B se nazývá subdoménou či poddoménou domény A. Důležitou doménou je kořenová doména (root domain), kde všechny další domény jsou její subdomény. Stěžejní pojem – doménové jméno, je jiné označení uzlu, které je napojením jmen jednotlivých uzlů pomocí teček. Uzel reprezentující kořen celkového jmenného stromu, má jméno tvořené prázdým řetězcem.

Obr. 1.1 zobrazuje část jmenného stromu včetně doménového jména webového serveru *www.nevim.to.*. Je zřejmé, že doménové jméno vzniká napojováním jmen uzlů ve směru od kořene zprava doleva. Jelikož je kořen reprezentován prázdným řetězcem, tak je na konci doménového jména jen napojovací tečka, kterou bývá zpravidla označena kořenová doména taktéž. Doménové jméno, obsahující i kořenovou doménu, se anglicky nazývá *Fully Qualified Domain Name* (FQDN), které musí být pro každý uzel jedinečné. Doménové jméno které není FQDN, není kompletní, je pouze relativní a pro přístup na uzel s tímto názvem je třeba znát i FQDN kořene uvažovaného podstromu. Obdobnou strukturou je souborový systém na UNIXových systémech, na kterém lze ukázat analogii s DNS, má také jeden kořen, lze se odkazovat relativními cestami, soubor může sdružovat další soubor (domény a poddomény) a soubor je vždy určen jednoznačnou cestou (FQDN).

Doména, jakožto podstrom, je způsob sdružení uzlů z obecně různých sítí (nejsou zde myšleny uzly jmenného stromu) do společné skupiny.

[1] Zde může nastat zmatek s označením právě popsaného programu (resolveru) a označení pro specifický typ jmenného serveru, provádějící rekurzivní překlad (rekurzivní resolver). Jedná se o rozdílné komponenty DNS a resolver bude značit vždy v zásadě jednoduchý program k vytvoření dotazu a přijetí odpovědi. Rekurzivní resolver bude někdy zaměňován za cache server.




**Obrázek 1.1:** Ukázka struktury doménového stromu

Existují význačnější domény, než jiné. Čím blíž je doména kořenové doméně, tím nabývá na „důležitosti“. Z toho důvodu mají i speciální názvy. Domény hned pod kořenovou doménou se nazývají *Top Level Domains – Domény prvního řádu* (TLD). Mezi nejznámější TLD patří národní domény *Country Code Top Level Domain* (ccTLD), které se vyznačují použitím kódů označujících zemi či teritorium podle normy ISO 3166 (např. *.cz, .mz, .to, ...*). Druhou skupinou TLD jsou generické domény prvního řádu – *Generic Top Level Domain* (gTLD). Zde se vyskytují domény typu *.com, .org, .mil*.

Kromě zmíněných nejčastěji užívaných TLD, existují ještě rezervované domény definované v RFC-2606 [15] (*.example*, *.invalid*, *.localhost*, *.test*) a doména infrastruktury (*.arpa*), dnes sloužící převážně pro reverzní záznamy IPv4 a IPv6 (*.in-addr.arpa*, *ip6.arpa*), více je v RFC-3172 [16].

### 1.2.2 Jmenné servery

Primární funkcí jmenných serverů (name server, DNS server) je poskytnout odpověď na požadavek o překlad doménového jména. Způsob, jakým je tato operace vykonána, dělí servery na různé typy. Jmenný server dokáže zpravidla relevantně odpovědět jen na dotazy spadající do tzv. *zón* tomuto serveru přidělené. Pro záznamy ze zóny je daný jmenný server autoritou.

V RFC-1034 [12] jsou pojmy autorita a zóna popsány následovně: *„Name servers know the parts of the domain tree for which they have complete information; a name server is said to be an AUTHORITY for these parts of the name space. Authoritative information is organized into units called ZONEs, and these zones can be automatically distributed to the name servers which provide redundant service for the data in a zone.“*

Server je autoritou pro určitou část jmenného prostoru, ke kterému má úplné

informace. Tím se rozumí schopnost serveru dát konečnou odpověď na dotaz (server vrátí odpověď s návratovým kódem NOERROR nebo NXDOMAIN, viz sekci 1.2.4 ). Pokud je server dotázán s požadavkem (třeba o překlad domény) a dokáže na základě dat ve své databázi odpovědět překladem, pak je pro tuto doménu autoritou. Data, pro které je server autoritou – autoritativní data má uložené v zónových souborech (zonefiles).

Jmenný server může část svých pravomocí odkázat jinému jmennému serveru. Místo toho, aby nadřazený jmenný server pro zónu *nevim.to*, ať už jsou důvody jakékoli, překládal záznamy i pro subdoménu *uz.nevim.to*, tak předá autoritu nad touto subdoménou podřízenému jmennému serveru. Příjde-li požadavek na nadřízený server o překlad doménového jména z domény *uz.nevim.to*, pošle odpověď, že překlad zajistit nezvládne, ale součástí odpovědi je zároveň odkaz na adresu podřízeného serveru. Tento odkaz je v databázi uložen pod typem NS a mnohdy je označován pojmem *glue record*, tj. záznam, který spojuje různé jmenné servery a díky němu je možné v distribuované databázi DNS vyhledávat. Odkazování jednoho serveru na druhý se také nazývá delegace autority. Limitem počtu zřetězených serverů je pouze možná délka doménového jména a tím pádem i počet subdomén.

Delegující server má pro delegovanou doménu ve své, nadřazené zóně pouze záznamy typu NS a A, které se vztahují k podřazenému jmennému serveru na nějž byla delegována autorita. Na tom je vidět zřejmý rozdíl mezi doménou a zónou. Doména reprezentuje vždy celý podstrom celkového doménového stromu. Údaje v zóně mohou být vztaženy k celému podstromu (doméně), ale také může vynechat údaje pro určité větve doménového stromu. Zóna obecně obsahuje záznamy, které jsou podmnožinou domény.

Procesem delegace autority je dosaženo markantního zlepšení kýžené škálovatelnosti. Jediné, co se mění s přidáním nového jmenného serveru pro novou doménu je to, že do databáze na nadřezeném jmenném serveru se přidá záznam NS pro delegaci (a k tomu záznamy pro DNSSEC, má-li být nová doména podepisována), o zbytek se postará nový jmenný server.

Jmenné servery se podle několika kritérií dělí do skupin. Je-li jmenný server pro nějakou část jmenného prostoru autoritou, pak se takový server nazývá *autoritativní*. Server, který není autoritativní, ale poskytuje odpovědi na dotazy dotazováním se ostatních autoritativních serverů, se nazývá *rekurzivním resolverem*, někdy, možná nepřesně, *cache* serverem.

### Autoritativní jmenné servery

Autoritativní jmenné servery drží informace o doménových jménech skrze zónové soubory a podle způsobu jejich držení se rozdělují na primární (primary, master) a na sekundární (secondary, slave). Primární server má v držení zónové soubory, které jsou uloženy na disku. Pokud je nakonfigurován k zónovému přenosu, tak

od něj mohou získat kopii zónového souboru sekundární servery, které si zónové soubory drží pouze v dočasné paměti (délka intervalu, po jehož uplynutí se sekundární server zeptá na zónový soubor, je jedním z parametrů záznamu SOA). To je hlavní rozdíl mezi primárním serverem a sekundárním. Účelem tohoto dělení je snadnější propagace změn v zóně, kdy stačí změnit jen zónový soubor na primárním serveru a o rozšíření změny se již systém postará sám. Server, který je pro jednu zónu primární, může být pro druhou sekundární a naopak. Označení *primární* se tedy vztahuje výhradně k určité zóně, nikoli ke spravovaným zónám jako celku.

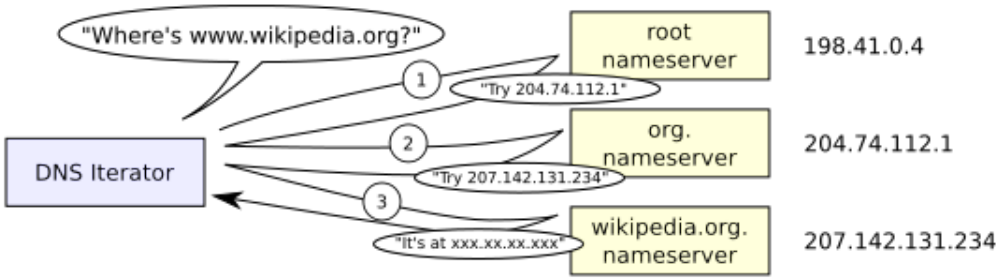
#### Cache servery

Cache servery představují odlišný model DNS serveru, který neslouží k poskytování informací jen z několika daných zón, jako je tomou u autoritativních serverů, ale zpravidla je určen k vykonání celkového procesu DNS překladu (taktéž je nazýván *rekurzivním resolverem*).

Když přijde na cache server dotaz, tak na základě domény, započne proces (obr. 1.2) iterativního doptávání autoritativních serverů. Nejprve se cache server zeptá stejným dotazem jmenných serverů pro kořenovou zónu ([a-m].root-servers.net). Na dotaz o překlad jména, např. algotech.cz, kořenové jmenné servery neodpoví překladem, ale poskytnou adresy jmenných serverů, na které je nutné se dále obrátit (TLD .cz). Po přesměrovaní na autoritativní server pro .cz TLD z dotazu, se cache server zeptá opět stejným dotazem a je mu, v tomto případě, vrácen seznam jmenných serverů, které už jsou autoritou pro doptávanou doménu. Z důvodu optimalizace velikosti datových toků a doby odezvy, obsahují cache servery paměť (cache, po které získal také označení), do které se výsledky z popsaného řetězce dotazování zapisují. Když přijde dotaz, tak se cache server nejdříve podívá do paměti, zda-li uz se stejný dotaz někdy vyřizoval, a pokud ano, tak jako odpověď vrátí výsledek z paměti a nezapočíná celý iterativní proces. Výsledky v paměti nezůstávají libovolně dlouhou dobu, ale řídí se hodnotou TTL záznamu. Když je výsledek v paměti déle než po dobu TTL, tak se výsledek z paměti zahodí a v případě dotazu na stejnou doménu, se znovu musí provést iterativní cyklus.

### 1.2.3 Resolvery

Resolverem se v rozdělení, které je tu popisováno, myslí program, jehož funkcí je zajistit DNS dotazování a sbírání odpovědí na uživatelské úrovni. V praxi to vypadá tak, že se jedná o nějakou knihovní funkci pro tvorbu paketu s DNS parametry, jeho odeslání na DNS server a zpracování odpodvědi. Jednotlivé uživatelské procesy mohou mít DNS dotazování řešeno takříkajíc po vlastní ose, nebo se spoléhat na některou službu operačního systému (např. systemd-resolved.service




**Obrázek 1.2:** Příklad iterativního doptávání cache serveru, zdroj [17]

v distribucích Linuxu se systemd init systémem). Většinou tyto resolvery nedělají rekurzivní překlad, ale s dotazem se obrátí na nějaký cache server, který případně rekurzivní překlad vykoná za něj.

### 1.2.4 DNS protokol

Způsob dotazování v DNS již byl z části popsán. K celkovému popisu je však třeba vidět dovnitř DNS dotazu (datové struktury, nalézající se v aplikačních vrstvách jak ISO/OSI modelu, tak TCP/IP modelu).

#### DNS zpráva

DNS jako protokol aplikační vrstvy, je na síťové vrstvě přenášen pomocí protokolu IP (IPv4 a IPv6) a na transportní vrstvě je zpravidla použit protokol UDP na portu 53, ač přenos přes TCP je také možný, od služby jako je DNS se očekává co nejvyšší rychlost, tím pádem režie se sestavováním TCP spojení je poněkud nežádoucí.

Obsahem datagramu je pak DNS zpráva ve formátu znázorněném v tabulkách z RFC-1035 1.1, 1.2. Každá zpráva obsahuje hlavičku, sekci otázky, sekci odpovědi, sekci autority a dodatek. Hlavička je složena z ID transakce (16 bitů) které slouží ke spárování otázky s odpovědí pro případ, že klient vyšle více dotazů v krátkém časovém intervalu a odpovědi mu chodí v jiném pořadí, než v jakém pokládal otázky. Následuje oblast několikabitových polí:

- **QR** (1 bit) – říká, jedná-li se o otázku (0) či odpověď (1).
- **Opcode** (4 bity) – značí typ dotazu, standardní dotaz (0), inverzní dotaz (1), dotaz na stav serveru (2), notifikace a změně zóny (4), aktualizace zóny (5), zbytek je rezervován (6-15).

**Tabulka 1.1:** Formát zprávy v DNS, RFC-1035 [13]

| Header |
|---|
| Question |
| Answer |
| Authority |
| Additional |

- **AA** (1 bit) – pouze pro odpovědi, říká, jde-li o autoritativní odpověď (0), či nikoli (1).
- **TC** (1 bit) – značí, že odpověď byla zkrácena z důvodu přesahující délky odpovědi povoleného rámce.
- **RD** (1 bit) – tímto bitem autor dotazu říká, že přednostně preferuje rekurzivní překlad (na serveru, kterého se táže), tento bit se kopíruje i do odpovědi.
- **RA** (1 bit) – odpověď na požadovanou rekurzi, říká, zda-li je podporována rekurze na serveru, či nikoli.
- **Z** (3 bity) – rezervováno pro budoucí použití, mělo by obsahovat samé nuly.
- **RCODE** (4 bity) – kód odpovědi, v RFC-1035 je definováno základních pět navratových kódů: NOERROR (0), FORMERR (1), SERVFAIL (2), NXDOMAIN (3), NOTIMP (4), REFUSED (5); RFC-2136 [18] definuje další kódy pro reprezentaci konfliktů při dynamických aktualizacích zón: YXDOMAIN (6), YXRRSET (7), NXRRSET (8); RFC-2671 [19] přidává hodnotu BADVERS (16); Zatím poslední přídavek udělalo RFC-6895, které přidalo hodnoty: NOTAUTH a NOTZONE;

Zbytek hlavičky je tvořen šestnáctibitovými poli QDCOUNT, ANCOUNT, NSCOUNT a ARCOUNT. Každé obsahuje počet záznamů v jednotlivých sekcích – QDCOUNT pro sekci Question, ANCOUNT pro Answer, NSCOUNT pro sekci Authority a ARCOUNT pro Addition.

Jednotlivé sekce následující po hlavičce, jsou buď prázdné nebo obsahují záznamy *resource records* (RR). RR je v podstatě řádek ze zónového souboru. Jeho formát ukazuje tabulka 1.3. Záznam je tvořen několika poli.

První údaj, je název vlastníka, tedy doménové jméno uzlu v doménovém stromě, ke kterému je tento záznam vztažen. Délka doménového jména je proměnná a je určena částečnými délkami uvedenými v záznamu na začátku každého jména uzlu (místa oddělená tečkami v doménovém jméně).

**Tabulka 1.2:** Formát hlavičky v DNS zprávě, RFC-1035 [13]

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ID | | | | | | | | | | | | | | | |
| QR | Opcode | | | | AA | TC | RD | RA | Z | | | RCODE | | | |
| QDCOUNT | | | | | | | | | | | | | | | |
| ANCOUNT | | | | | | | | | | | | | | | |
| NSCOUNT | | | | | | | | | | | | | | | |
| ARCOUNT | | | | | | | | | | | | | | | |

**Tabulka 1.3:** Formát záznamu RR, RFC-1035 [13]

| Name | Type | Class | TTL | RDLENGTH | RDATA |
|---|---|---|---|---|---|

Dalším polem je šestnáctibitový typ záznamu, který separuje různé záznamy pro stejného vlastníka a dává informaci o tom, co se má očekávat za data v sekci RDATA. Základní přehled nejčastějších typů záznamů dává tabulka 1.4. Pro ucelenější seznam nechť ctěný čtenář navštíví [20].

Následující pole je šestnáctibitová třída. Každá zóna se nachází v třídě, což je jakýsi jmený prostor nad doménovými jmény, tedy dvě stejná doménová jména v různých třídách mají různý význam. Možné třídy jsou čtyři: IN – třída pro užití v Internetu; CH – třída Chaos, původně určená pro testování; HS – třída Hesiod, víceméně pro experimentální účely, CS – třída pro CSNet, již v době RFC-1035 byla označena jako *obsolete*. Ze všech uvedených tříd, došlok praktickému využití, až na pár vyjímek s třídou CH, pouze u třídy IN.

Každý záznam je opatřen 32 bitovou hodnotou TTL (Time to Live), která v sekundách vyjadřuje dobu platnosti záznamu uloženého v cache paměti resolverů.

Poslední dvě pole jsou délka dat a samotná data. Délku tvoří šestnáctibitové číslo, a udává délku dat v bytech. Forma dat v datovém poli pak odpovídá typu záznamu, popsanému výše.

### Komunikace s DNS serverem

Dosud nebyl zcela popsán řetězec událostí při vybavování dotazu. Když se naskytne potřeba přeložit doménové jméno či poslat dotaz jiného typu, v drtivé většině případů se tázající nejdříve zeptá (skrze funkci lokálního resolveru) svého DNS serveru (jehož adresu dostane zpravidla přidělenou DHCP serverem). Tento DNS server je rekurzivní resolver/cache server. Jakmile na něj padne dotaz, pokud nemá odpověď v paměti, začne ho vyřizovat iterativním procesem popsaným v 1.2.2. Bylo řečeno, že autoritativních jmenných serverů může být vícero pro jednu zónu. To, který se vybere, záleží na odpovědi ze serveru pro nadřízenou

**Tabulka 1.4:** stručný výčet některých typů DNS záznamů

| Typ | Popis |
|---|---|
| A | IPv4 adresa |
| AAAA | IPv6 adresa |
| PTR | Doménové jméno pro reverzní překlad |
| MX | Seznam poštovních serverů |
| CNAME | Alias pro doménové jméno |
| NS | Jméno jmenného serveru pro danou zónu |
| SOA | Informace o zóně |
| TXT | Libovolný text |
| RRSIG | Podpis záznamu |
| DNSKEY | Šifrovací klíč |

zónu, který v odpovědích servery pro delegovanou zónu střídá, čímž se dosahuje přibližně vyrovnaného zatížení daných serverů. Výsledek iterativního doptávání si rekurzivní resolver uloží do paměti a pošle tazateli. Komunikace ze strany klienta je tak většinou dvoubodová ve smyslu, že se s DNS dotazem obrací pouze na jeden sever, který za něj všechnu práci s překladem odvede.

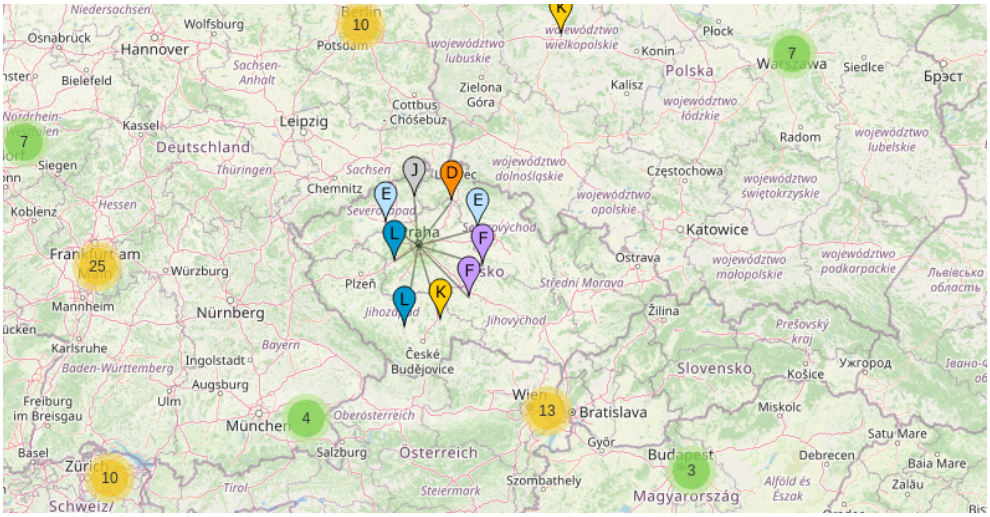
## 1.3 Architektura DNS

Na architekturou se dá pohlížet dvěmi způsoby. První může být pohled z *blízka*, kdy se hledí na uskupení serverů v praktické implemenetaci systému v síti nějaké společnosti (Kolik serverů, kolik primárních a sekundárních, atd.). Druhý pohled může z celosvětového měřítka zkoumat, jak je síť DNS postavená, které servery spravují které domény, jak jsou delegována práva na *nižší* subjekty a podobně.

### 1.3.1 DNS v lokální síti

Možná nejčastější implementace DNS v LAN je za pomoci Microsoft Active Directory, jehož pevnou součástí práve je Active Directory Domain Services (AD DS). DNS z AD DS však představuje řešení pro komunikaci uvnitř sítě.

Tvorba autoritativního DNS pro vlastní domény by se měla řídit především bezpečnostními kritérii. DNS servery jsou oblíbeným cílem DoS útoků, tudíž je zapotřebí také uvažovat o robustnosti řešení. Oblíbenou topologií autoritativního DNS je tzv. *hidden master*. Toto schéma spočívá v existenci jednoho či více primárních jmenných serverů, držících originály zónových souborů pro spravované zóny. Primární servery jsou dostupné pouze z vnitřní sítě. Na primární server/y je napojeno $N$ sekundárních serverů, které mají jedno síťové rozhraní pro vnitřní síť,




**Obrázek 1.3:** Servery pro kořenovou zónu v Česku a okolí

druhé pro připojení se do veřejné sítě. Záznamy na sekundárních serverech jsou spravovány pouze primárním serverem, který je nedosažitelný z vnější sítě. Tento přístup dosti ztěžuje útok na zdroj originálních dat. Je-li veden útok z vnějšku, tak pouze na sekundární servery, jejihž funkce nijak neohrožuje originální zónové soubory. Sekundární servery jsou takříkajíc nahraditelné. Toto schéma bude také použito v praktické části této práce (kap. 4).

### 1.3.2 DNS z globálního pohledu

DNS jako globální distribuovaná síť hraje důležitou roli, a tak jsou její prvky, minimálně ty blíže kořenu zabezpečeny a dimenzovány natolik, aby v případě výpadku některých elementů byl systém stále funkční.

Jmené servery pro kořenovou zónu jsou na 13 doménách ([a-m].root-servers.net) a zároveň na 13 IP adresách. Neznamená to, že by bylo celosvětově jenom 13 jmenných serverů pro kořenovou zónu, ale IP adresy pro kořenové servery jsou anycastové a tak se za jednou adresou skrývá více fyzických serverů, klient tak může komunikovat s geograficky bližším severem ale se stejnou logickou (síťovou) adresou [21]. Na stránce [22] je seznam kořenových serverů a společností, které je spravují, jak vidno, nestojí za nimi pouze jedna. Stránka [23] zobrazuje geografické rozložení serverů pro kořenovou zónu, obr. 1.3 ukazuje jejich počty nebo distribuci v Česku a okolí.

Kořenová zóna deleguje autoritu dále do TLD zón, které spadají pod různé komerční i nekomerční subjekty, nicméně z hlediska architektury se mnohdy neliší od kořenové. Také využívají anycastové směrování a distribuují své jmenné servery

po celém světě. Delegované zóny hlouběji pod TLD už jsou většinou ve správě menších subjektů a robustnost, vlastní kořenové, či některým TLD zónám, už není na takové úrovni, avšak není ani vyžadována.

Ohledně hospodaření s TLD jsou zajímavé příjmy z provozu atraktivních ccTLD. Ostrov Anguilla, jehož ccTLD je *.ai*, zaznamenal s celospolečenským zpopularizováním umělé inteligence v roce 2023 příjmy z registrací domén pod *.ai* 30 milionů amerických dolarů (USD), což je oproti roku 2022 nárůst o cca 316 % [24]. Obdobně přitažlivá doména je třeba *.tv*, která spadá pod stát Tuvalu.

# Kapitola 2

# Bezpečnost v DNS

Jedním z hlavních nedostatků původního DNS (RFC-1034, RFC-1035) byla absence jakýchkoli bezpečnostních praktik. Celá komunikace mezi klientem a serverem probíhá protokolem UDP na portu 53 v otevřené podobě, tudíž prakticky kdokoli na trase putování zprávy dokáže zprávu odchytit a přečíst si, na co se klient ptá. To by ještě, kromě zřejmého narušení soukromí, nebyl takový problém, ale na odchycený dotaz lze ve vhodnou dobu podvrhnout odpověď, a to už je značný problém.

V následujících podkapitolách budou popsány zranitelnosti DNS, typy útoků a techniky, kterými se zranitelnostem bránit. Největší důraz bude kladen na DNSSEC, ježto je jedním z hlavních témat zadání.

## 2.1 Zranitelnosti DNS

Tato podkapitola se nezabývá zranitelnostmi jednotlivých implementací DNS serverů, ale zranitelnostmi samotného protokolu. V roce 2004 vyšlo RFC-3833 s názvem *„Threat Analysis of the Domain Name System (DNS)“* [25], které zkoumalo pro útok využitelné vlastnosti DNS protokolu. V roce 2013 také vyšla příručka pro nasazení zabezpečené DNS infrastruktury vzešlé z instituce NIST (National Institute of Standards and Technology) [26]. Uvedené zdroje jsou dobrým souhrnem rizik i prevence, z nichž bude následující popis čerpat mimo jiné.

Hlavní vlastnosti DNS, vedoucí k různým typům útoků, jsou v jádru dvě: první je absence nutnosti se nějakým způsobem autentikovat při dotazování se na server, druhou je pak otevřenost komunikace. První umožňuje komukoliv se dotazovat a dostávat odpovědi, což je pro globální systém vlastnost chtěná, nicméně také dobře zneužitelná k DoS útokům. Druhá vlastnost – otevřenost komunikace dělá komunikaci transparentní a se znalostí toho, na co se klient ptá, lze podvrhnout odpověď.

## 2.2 Typy útoků a zneužití

Popsané techniky v zásadě využívají dvě výše popsané vlastnosti protokolu DNS.

### 2.2.1 DoS a DDoS

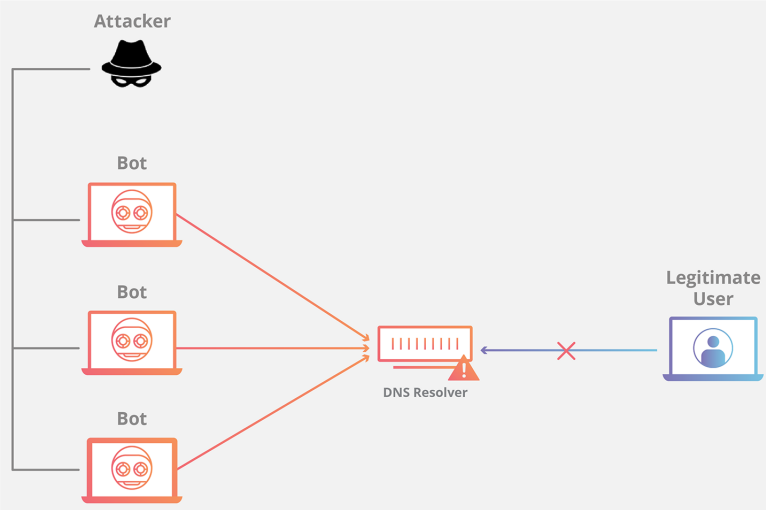
Útok spočívající ve vyřazení určité služby se nazývá *Denial of Service* (DoS). DNS DoS je útok, jež se skrze enormní datový tok tvořený DNS dotazy snaží naplno vytížit DNS server tak, že není schopen dále odpovídat na ostatní dotazy. Další charakteristikou DoS je směřování útoku z úzkého okruhu útočících klientů (stroje ze stejné sítě, apod.). Tato skutečnost dovoluje vcelku efektivní zamezení tomuto typu útoku.

Útok typu DoS vedený z velikého množství klientů z různých sítí se nazývá *Distributed Denial of Service* (DDoS). DDoS je v principu stejný jako u DoS (zamezení chodu služby na základě zahlcení serveru požadavky, akorát z více míst oproti DoS), ale naprosto zásadně se liší se schopností být zamezen. Jednoduchým pravidlem firewallu se takový útok nedá mitigovat, protože povedený útok se jeví, že přichází odevšad. Pokud by se útok skládal z podezřelých DNS dotazů (řada dotazů za sebou se stejným Query ID), dá se útok mírnit, nicméně obnáší to značně více práce než u jednoduchého DoS. DDoS může být vyvolaný z botnet sítě napadených zařízení, čímž se docílí ta distribuovanost z názvu útoku.

Útoky směřující na zamezení chodu služby jsou ve světě DNS s čím dál více nasazovaným DNSSEC nejčastějším útokem. Infrastrukturu pro DNS, jako kritickou službu, je třeba dimenzovat na několikanásobek běžného provozu – takové opatření zůstává jako nejsilnější ochrana proti DDoS, protože spíše k DoS nedojde.

### 2.2.2 DNS flood

Cílem tohoto útoku je taktéž vyřazení služby, avšak DNS servery zde hrají roli prostředníka, nikoli bod k vyřazení. Schéma je na obr. 2.1. Princip spočívá ve vysílání DNS dotazů útočníkem ovládaným botnetem směrem k DNS serveru stejně jako v minulém případě. Rozdíl je ve zdrojové IP adrese paketu, která se změní na adresu oběti (spoofing). Tímto způsobem pak DNS server nepošle odpověď na dotaz zpět útočníkovi, ale nasměruje ji k oběti. Pří úspěšném útoku tak dojde k vyřazení síťového rozhraní oběti s útočníkem použitou adresou. Obranou takovému útoku může být blokace na straně DNS, nicméně tak dojde k zamezení přístupu oběti na daný DNS server.




**Obrázek 2.1:** Příklad útoku typu flood, zdroj: [27]

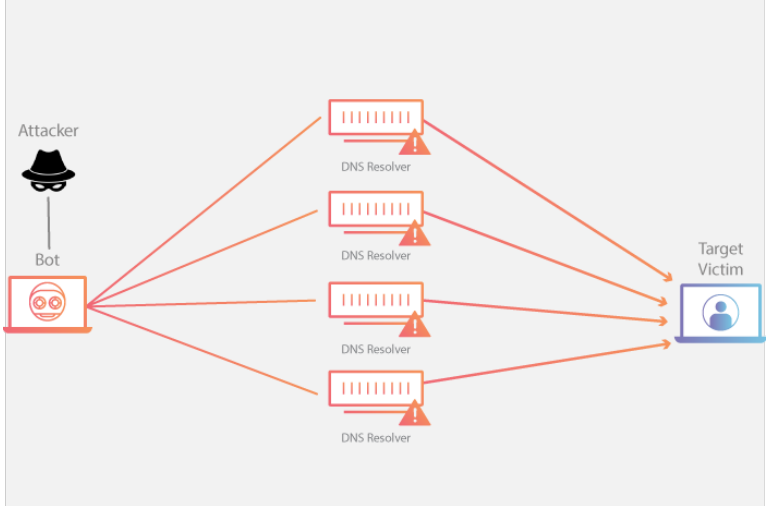
### 2.2.3 DNS amplification

Nápadně podobný útok útoku předešlému. Schéma ukazuje obr. 2.2. Rozdíl spočívá v tom, že nároky na útočníka nekladou podmínku vlastnictví botnetu, ale stačí mu podstatně méně strojů (na obrázku jeden). Útočník opět podvrhne zdrojovou adresu dotazu na adresu oběti a vzniklé dotazy posílá na DNS servery, které odpovědi pošlou směrem k oběti. Útočník vybírá takové dotazy na takové záznamy, aby poměr velikosti odpovědi vůči velikosti dotazu byl co největší (TXT, RRSIG, apod.).

### 2.2.4 Otrávení cache (Cache poisoning)

Otrávení cache je útok, jehož cílem je podvrhnout pravý záznam na cache serveru. Útočník pošle dotaz na záznam, který chce podvrhnout, na cache server. Předpokladem je, že server záznam nemá v paměti, a tak se začne doptávat autoritativních serverů na položený dotaz. Jakmile cache pošle dotaz a očekává odpověď, do hry opět vstoupí útočník, který pošle odpověď s podvrženým záznamem. Cache server si odpověď na dotaz uloží do paměti, a po dobu TTL bude na tento dotaz odpovídat podvrženou odpovědí. Pro úspěšný útok vyvstávají pro útočníka dva problémy:

- Načasování – útočník musí načasovat vyslání podvržené odpovědi tak, aby stihla na server přijít před pravou odpovědí.
- Autenticita – útočníkova odpověď musí splňovat parametry dotazu, který vznesl server. Přinejmenším se musí strefit do stejného ID dotazu.




**Obrázek 2.2:** Příklad útoku typu amplification, zdroj: [28]

Takové problémy nejsou nikterak triviální, a z otrávení cache se stává celkem obtížný typ útoku k realizování. Obrana proti tomuto útoku spočívá v zapnuté validaci pomocí DNSSEC. Více se lze dočíst z [29].

### 2.2.5 Únos DNS (DNS hijacking)

Jádrem této techniky je, stejně jako u otrávení cache, podvržení záznamů kupříkladu za cílem zobrazení webové stránky útočníků, ne nepodobné té, na kterou měla oběť úmysl vstoupit. Pokud nezaregistruje, že se jedná o podvrh, tak může vyzradit své údaje, přijít o majetek a podobně. Způsob, jakým k podvrhu záznamu dojde, je buďto pomocí nějaké slabiny získat přístup na jmenný server a na něm měnit, záznamy, nebo stejným způsobem získat práva na jmenný server pro nadřazenou zónu, ve které se upravý záznam typu NS, který odkazuje na útočníkův jmenný server. Mimo získání práv na jmenných serverech je možné pomocí otrávení cache podvrhnout NS záznam. Více o útoku v článku [30].

### 2.2.6 DNS tunelování (DNS Tunneling)

Zde se nejedná ani tak o útok, ale o techniku ukrytí komunikace do DNS zpráv, která je však zneužitelná různým malwarem. Tato technika vytváří komunikační kanál, který má vyšší šanci obejít firewall. Pokud chce útočník dostat informace ze zařízení oběti a podaří se mu na zařízení dostat nástroj, který tato data sbírá, může je posílat skrze DNS kanál. Ten vytvoří tak, že zařízení data kóduje nějakým způsobem tak, aby byly použity jen platné znaky pro doménová jména. Takto zakódovaná data jsou připojena k doménovému jménu útočníkova jmenného serveru.

Poté zařízení vznese požadavek o překlad, zdánlivě nesmyslného jména vytvořeného v předešlém kroku, na útočníkův server. Tímto způsobem se k útočníkovi dostanou zakódovaná data, se kterými může dál pracovat. Komunikace může být obousměrná, druhá strana posílá stejným způsobem data v odpovědi. Tím, že se komunikuje pomocí DNS zpráv, které mohou být velké 512 bytů, je hlavním omezením možná délka doménového jména. Na posílání větších datových toků je to technika nepoužitelná, nicméně zprávy typu email se posílat dají. Zmíněné obejití firewallu spočívá v tom, že většinou je na firewallech DNS služba povolena (53/udp) a není taková komunikace vyhodnocena jako hrozba.

## 2.3 DNSSEC

Zranitelnost DNS vůči útokům zneužívajícím neověřování pravosti záznamů (zmíněné útoky otrávení cache, únos) vedla k vytvoření mechanismu, který se snaží tyto vlastnosti opravit a potlačit možnost úspěšného útoku. Tento mechanismus vstoupil do světa v podobě DNSSEC, představené v RFC-4033 *„DNS Security Introduction and Requirements“* [31], RFC-4034 *„Resource Records for the DNS Security Extensions“* [32] a v RFC-4035 *„Protocol Modifications for the DNS Security Extensions“* [33]. Pěkný souhrn dokumentů RFC pojednávajících o DNSSEC je v RFC-9364 *„DNS Security Extensions (DNSSEC)“* [34].

### 2.3.1 Princip

DNSSEC k zabezpečení záznamů využívá digitálních podpisů. Každý záznam v podepsané zóně má přidružený záznam typu RRSIG, který obsahuje podpis daného záznamu. K podpisu je zapotřebí klíč, jehož veřejná část je pro zónu uložena v záznamu DNSKEY, soukromou část si server střeží pro sebe. Proces podepisování probíhá v krocích:

1. Vypočítá se hash daného záznamu podle zvolené hashovací funkce.
2. Hash se zašifruje pomocí soukromého klíče (pro danou zónu) zvoleným algoritmem.
3. Výsledek se uloží do nového záznamu typu RRSIG, který má stejného vlastníka (doménové jméno).

Záznam RRSIG obsahuje mimo samotného podpisu také typ záznamu, ke kterému se podpis vztahuje, algoritmus podpisu, TTL podepisovaného záznamu, termín vypršení podpisu, termín začátku platnosti podpisu, tag klíče, jméno toho, jež podepisoval (volí se prosté jméno zóny). Příklad záznamu typu RRSIG pro záznam typu SOA je ve výňatku zónového souboru 2.1.

```
nevim.to.    3600    RRSIG   SOA 8 2 3600 (
                     20240109135140 20231210135140 61159 nevim.to.
                     Yeu1HK0DD8zNuEcz/+yfz1g8FnfF63mn3Pd/
                     sOgAHohgfRylPdSDIXzlApGxN0U7EMEWVZTG
                     +sDhdgjaaT+b/h8sS9L9WLuVD8JrDG5F5x+j
                     5MUP2TkEL2J/7e9AC5hNPPveDkIAJ8vfZ/+I
                     q0SwoQC4tAIAZHG+16f90xhFwYk= )
```

**Zónový soubor 2.1:** Příklad záznamu typu RRSIG (záznam nelze vyhledat, ilustrační příklad)

Na první pohled je trochu kuriózní fakt, že se v zóně nepodepisují jen záznamy v ní obsažené, ale také bloky, které říkají, že žádný další záznam mezi sousedními záznamy není. Zkrátka, pokud chce server odpovědět s RCODE NXDOMAIN, musí být odpověď ověřeníschopná pomocí DNSSEC validace. Toho se dosáhne tak, že při podepisování zónového souboru jsou záznamy nejdříve lexikograficky uspořádány. Ke každému ze vzniklých shluků záznamů vztažených ke stejnému doménovému jménu je přidán další záznam typu NSEC. Ten obsahuje doménové jméno předešlého shluku (pro první shluk je to jméno posledního) a výčet typů záznamů, které se v daném shluku vyskytují. Při dotazu na typ, který pro dané doménové jméno neexistuje, se vrátí záznam NSEC, který obsahuje jen platné typy pro dané jméno. Při dotazu na doménové jméno, které není v zóně obsaženo, server vrátí záznam NSEC, který pokrývá interval, do kterého se lexikograficky umístilo doptávané jméno.

Jelikož lze pomocí záznamů typu NSEC zjistit strukturu zóny čistě validním doptáváním se na záznamy. Technika využívající tento postup se nazývá *zone walking*, obecněji spadá do techniky *zone enumeration*. Na potlačení takového konání existuje mimo NSEC záznamu záznam typu NSEC3.

Rozdíl NSEC3 oproti NSEC je v použití hashe předcházejícího jména namísto jména v otevřené podobě. Tím je dosaženo ukrytí struktury zóny, ale se stále funkčním důkazem neexistence nějakého záznamu.

### 2.3.2 Řetězec důvěry

Doposud byly představeny jen záznamy s podpisy a záznam, uchovávající veřejný klíč. Pro ověření spolehlivosti záznamů je však potřeba uchování informace o podepisovacím klíči na spolehlivém místě (vygenerovat a podepsat zónu může útočník taktéž). Co se týče podepisovacích klíčů, ve skutečnosti jsou použity dva: *Zone Signing Key* (ZSK) a *Key Signing Key* (KSK). Klíč ZSK slouží k podepisování záznamů v zóně jak bylo ukázáno. Klíč KSK pak slouží k podepsání pouze záznamů obsahující veřejné klíče (DNSKEY). Na serveru, který spravuje nadřazenou zónu, je pak uložen záznam typu DS, který obsahuje hash klíče KSK. Ten je v nadřazené zóně podepsán pomocí ZSK nadřazené zóny, ten je podpsán KSK nadřazené zóny a

jeho hash je uložen v nadřazené zóně nadřazené zóny. Tímto způsobem se záznamy s klíči poutají a tvoří řetězec. Na konci řetězce je vždy kořenový jmenný server, který také má vlastní ZSK a KSK, avšak nemá žádnou nadřazenou zónu, kam by uložil záznam DS. Klíče na kořenových serverech jsou tak považovány za platné vždy.

### 2.3.3 DNSSEC validace

Postup, kterým klient ověří platnost přijatého záznamu vnáší do vyřízení dotazu dosti velké navýšení počtu zpráv v systému. Je to tak neblahá daň za autenticitu a integritu poskytnutých záznamů.

Když klient dostane odpověď na dotaz a chce ověřit jeho pravost, udělá následující kroky:

1. Opatří si podpis záznamu v záznamu typu RRSIG.
2. Opatří si veřejný klíč ZSK v záznamu typu DNSKEY.
3. Veřejným klíčem ZSK dešifruje podpis a porovná ho s hashem ověřovaného záznamu.
4. Opatří si podpis klíče ZSK.
5. Opatří si klíč KSK.
6. Klíčem KSK dešifruje podpis klíče ZSK a výsledek porovná s hashem klíče ZSK.
7. Z nadřazené zóny si opatří záznam DS pro danou zónu.
8. Porovná záznam DS a hash klíče KSK.
9. Opakuje předchozí kroky pro ověření klíčů v nadřazených zónách, pokud jakékoli porovnání skončí neúspěchem (porovnávaný hash záznamu a dešifrovaný podpis nejsou stejné), pak je narušena autenticita ověřovaného záznamu a dotaz bude považován za nezodpovězený.

## 2.4 Opatření proti útokům

Podepisování zón pro ověřování záznamů DNSSEC validací je dobrá praktika, která ochrání tázající klienty. Nicméně neřeší obranu proti útokům DoS. V kapitole 3 jsou představena různá řešení pro DNS, řešící i ochranu takovýchto hrozeb.

DNSSEC sice dokáže odhalit podvržený záznam, což může zabránit například infiltraci malwaru na uživatelovo zařízení, ale DNS zprávy pořád putují sítí v otevřené podobě, takže kdokoli po cestě vidí, jaká adresa se ptá na jakou doménu či na jaký záznam. Je to spíše otázka soukromí, nicméně útočník by mohl podle DNS dotazů vysledovat chování oběti (např. jaké webové stránky asi navštěvuje) a to pak využít při plánování útoku.

Otázku soukromí a bezpečnosti v DNS provozu řeší řada protokolů, jako jsou DoH (DNS over HTTPS), DoT (DNS over TLS), DNSCrypt nebo DNS over Tor. Všechny tyto techniky přidávají do DNS dotazování režii a z původně zamýšleného konceptu „jeden dotaz – jedna odpověď“, který cílil na rychlost, se stává vcelku komplexní systém s nutnostní sestavení spojení atd.

# Kapitola 3

# Současná řešení DNS

V současné době je nabízena řada řešení pro nasazení DNS, jak pro autoritativní servery, tak pro cache servery. Akcentovaná jsou ta řešení, která cílí na bezpečnost, nejen aplikováním DNSSEC, ale ještě pomocí nadstavbových systémů, přinášející další bezpečnostní vrstvu.

Možností, jak implementovat autoritativní jmenný server nebo rekurzivní resolver (pomocí jakého softwaru), je poměrně vysoké množství, leč z Wikipedie, pěkné porovnání je v článku [35]. Ve zdejším výběru jsou poznamenány pouze některé, které povětšinou implementují všechny funkce, které jsou očekávány: od autoritativního serveru podpora primárních a sekundárních jmenných serverů, podpora DNSSEC – podepisování zón a podobně; od resolverů se očekává podpora validace DNSSEC a případně další bezpečnostní praktiky.

Nutno podotknout, že bezpečnostní otázka týkající se DoS útoků je obtížná. Většina implementací umožňuje nastavení tzv. *rate limitu* na serveru, omezující počet dotazů za časovou jednotku od jednoho klienta. Sofistikovanější DoS jsou však problém a k úspěšné mitigaci je třeba specializovaná infrastruktura mimo DNS.

## 3.1 DNS řešení

Následuje výběr několika řešení jak pro autoritativní servery tak pro rekurzivní resolvery.

### 3.1.1 BIND9

DNS server BIND (Berkeley Internet Name Domain) [36] je takovým nepsaným standardem v oblasti DNS serverů. Vyvíjen je pod záštitou organizace Internet Systems Consortium (ISC) [37]. BIND podporuje takřka všechny funkce DNS ser-

veru: mimo autoritativního serveru lze nakofigurovat i rekurzivní resolver/cache; má plně implementován DNSSEC; podpora pro primární a sekundární autoritativní jmenné servery (podpora přenosu zón, notifikace). Jeho funkce a fakt, že BIND je uvolněn jako open-source z něj dělají jedním z nejpoužívanějších DNS serverů.

BIND je server, konfigurovatelný úpravou textových konfiguračních souborů, postrádá grafické uživatelské rozhraní. Konfigurační soubory se však dají upravovat různými nástroji pro zpracování textu (awk, sed) a konfiguraci tak vytvářet automatizovaně dle zadaných parametrů.

Dále má BIND také bohaté možnosti logování, od zachytávání chyb, které se při provozu serveru naskytly, přes přístupové logy s dotazy, kladenými na server až po celkové statistiky, které mohou být vystaveny přes soubor nebo interním HTTP serverem.

### 3.1.2 Knot

Knot je projekt správce české domény CZ.NIC a skládá se ze dvou částí. Autoritativní server je implementován v části projektu s názvem Knot DNS [38], drouhou část je Knot Resolver [39], který implementuje, jak název napovídá, DNS resolver. Knot je uvolněn jako open-source [40], [41] a prodělává stálý vývoj. Knot DNS si klade za cíl implementaci vysokovýkonového moderního autoritativního serveru. V zásadě má všechny základní funkce jako výše zmíněný BIND (v autoritativní části).

Druhá část projektu DNS Resolver ma za cíl implementovat malý výkoný DNS rekurzivní resolver, který by byl postaven na modulární architektuře, která umožňuje přidávat aktualizacemi funkce. Kromě validace DNSSEC podporuje řadu dalších bezpečnostních aspektů, jako je DoH, DoT. Umožňuje funkci DNS64 [42], které umožňuje komunikaci IPv6 klienta s IPv4 servery pomocí NAT64.

### 3.1.3 PowerDNS

Projekt PowerDNS je, tak jako předešlý projekt Knot, rozdělen na část implementující autoritativní server – PowerDNS Authoritative Server [43] a na část implementující rekurzivní resolver – PowerDNS Recursor [44]. Oba moduly, PowerDNS Authoritative Server a PowerDNS Recursor, jsou navrženy s důrazem na vysoký výkon a flexibilitu. Projekt je uvolněn také jako open-source dostupný z [45].

Vlatností autoritativní části je možnost využití různorodých backendů jakožto databázi DNS záznamů. Umožňuje mít data o zónách uložena v SQL databázi (MySQL, PostgreSQL, SQLite) nebo v zónových souborech jako to má server BIND, či v řadě dalších. Dokáže se tak dobře přizpůsobit potřebám uživatele (administrátora). V závislosti na použitém backendu PowerDNS Authoritative Server podporuje různé funcke jako je DNSSEC, primární a sekundární server a podobně.

Část implementující rekurzivní resolver – PowerDNS Recursor cílí na vysoký výkon. Umožňuje bezpečnostní funkce jako DoT a DoH. PowerDNS Recursor podporuje skriptování v jazyce Lua. To umožňuje uživatelům vytvářet vlastní skripty, které modifikují chování resolveru podle specifických požadavků. Recursor obsahuje opatření pro ochranu proti amplifikačním útokům, které by mohly zneužít DNS servery k posílání velkého objemu odpovědí na relativně malé dotazy.

### 3.1.4 Autoritativní DNS v cloudu

Zajímavou alternativou k provozování DNS na vlastních serverech je přesun hostingu do cloudu a odpadá tak zodpovědnost za chod strojů, problémy s fyzickým umístěním a podobně, na druhé straně se samozřejmě jedná o placenou službu typu IaaS nebo SaaS (Infrastructure as a Service, Software as a Service).

Nějaký typ cloudového DNS nabízejí velké softwarové společnosti jako Google (Google Cloud DNS), Microsoft (Azure DNS), Amazon (Amazon Route 53), IBM (IBM Cloud DNS), Cloudflare a další. Všechny se pyšní bezpečným a robustním systémem.

### 3.1.5 Microsoft DNS

Windows DNS Server je implementací DNS serveru poskytovanéhojako součást operačního systému Windows Server. Hlavním účelem Windows DNS Serveru je poskytovat DNS služby ve Windows prostředí. Windows DNS Server může fungovat buď jako autoritativní server, nebo jako rekursivní server. Windows DNS Server je často integrován s Active Directory, což usnadňuje správu DNS záznamů ve spojení s informacemi v Active Directory. Tato integrace umožňuje automatickou aktualizaci DNS záznamů pro počítače, které jsou přidány do domény.

## 3.2 Resolvery/cache s bezpečnostní nadstavbou

Projekty zmíněné výše (BIND, Knot, ...) zpravidla řeší bezpečnost podle RFC (typu DNSSEC). Na trhu však existují specializované produkty, které přináší nadstavbovou část nad běžným rekurzivním resolverem, snažící se zamezit bezpečnostním incidentům. Některá z těchto bezpečnostních řešení umožňují filtrovat provoz na základě reputační databáze domén, vytvářet whitelisty a blacklisty povolených či naopak nežádoucích domén, provádět analýzu provozu v reálném čase pomocí metod strojového učení (Whalebone). Většinou se jedná o produkty, které jsou výsledkem vývoje komerčních firem a nejedná se tak o řešení typu open-source. Následující úzký výběr představuje jen malou část možných rešení.

### 3.2.1 Akamai CacheServe

Spolecnost Akamai vyvíjí DNS resolver CacheServe [46] (před akvizicí v roce 2017 ze strany Akamai se jednalo o produkt firmy Nominum). Hlavními prezentovanými vlastnotmi jsou robustnost řešení z hlediska výkonu a obranyschopnost proti útokům typu cache-poisoning. Mimo DNS resolveru má společnost Akamai v nabízeném portfolilu také různé síťové a bezpečnostní služby, včetně ochrany vůči útokům DDoS.

### 3.2.2 Whalebone

Whalebone Peacemaker [47] je komerční produkt implementující rekurzivní resolver s dodatečnou bezpečnostní vrstvou. Kromě podpory DNSSEC, IPv6 nebo DoH nabízí řešení s přidanou hodnotou v podobě analýzy provozu.DNS dotazy dopadající na resolver jsou podrobeny prozkoumání za pomoci reputační databáze. Jako DNS server je v řešení Whalebone použit Knot Resolver.

### 3.2.3 Allot

Společnost Allot taktéž nabízí DNS resolver [48] s bezpečnostními opatřeními v podobě aktivní ochrany s reputační databází. Více se lze dočíst z [49]. Řešení Allot DNS Secure je jedním z řady síťových produktů této společnosti, která mimo jiné nabízí řešení proti útokům DDoS. Allot DNS Secure nabízí ochranu proti phishingovým útokům, filtrování provozu (také jako rodičovskou kontrolu).

## 3.3 Veřejná DNS

Jako rekurzivní resolver lze využívat řadu známých DNS serverů, vystavených veřejnosti. Mezi nejznámější patří servery společnosti Google (IPv4 adresy 8.8.8.8 a 8.8.4.4) nebo Cloudflare (IPv4 adresy 1.1.1.1 a 1.0.0.1). O veřejné DNS servery se dá opřít v nouzi, pakliže vlastní resolver není dostupný, došlo k výpadku a podobně, ovšem s potenciálním rizikem úniku části svého soukromí. Výhodou může být podpora DoT (DNS over TLS) nebo DoH (DNS over HTTPS) některých poskytovatelů, která zvýší míru soukromí po cestě mezi klientem a DNS serverem.

# Kapitola 4

# Návrh a implementace vlastního řešení

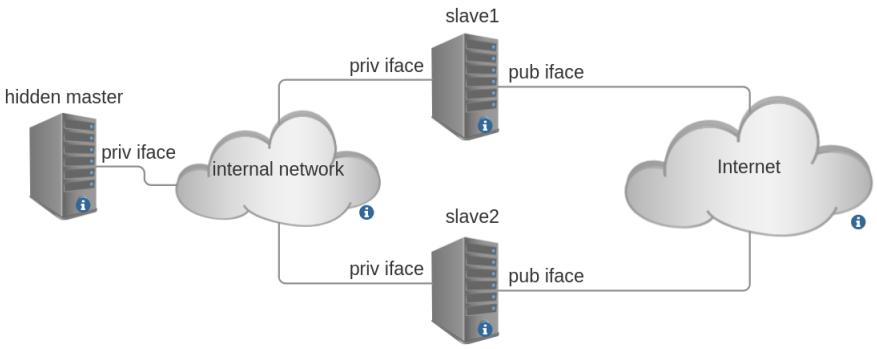
Jádrem této práce je návrh infrastruktury DNS v prostředí cloudového poskytovatele, od výběru a instalace autoritativních serverů, přes zajištění jejich bezpečnosti až k tvorbě systému pro správu záznamů, podepisování zón a kontrolu platnosti podpisů. Obstarat všechny tyto požadavky si žádá pečlivý návrh řešení, ježto potenciálních problémů, naskytnuvše se při implementaci neprověřeného modelu, je dost. Následující sekce se zabývají návrhem všech částí systému a jeho implementací.

## 4.1 Výchozí stav DNS ve firmě Algotech

DNS infrastruktura v Algotechu za doby psaní této práce je rozdělena do dvou částí. První část je DNS pro vnitřní síť, sloužící pro správu serverů a interní komunikaci. DNS pro vnitřní síť je implementováno na základě Microsoft Active Directory. Druhou částí je veřejné DNS, které slouží k veřejnému přístupu klientů na hostované aplikace. Kromě autoritativnícho serveru je zde také cache server, který je poskytován zákazníkům jako IaaS. Na autoritativním serveru není implementováno DNSSEC, jehož implementace (na testovacím stroji) je jedním z hlavních cílů této práce.

## 4.2 Návrh uspořádání serverů

Prvek, který by bylo vhodné ze současného řešení udržet, je separace DNS pro vnitřní síť a veřejného autoritativního serveru. Pokud by došlo například k útoku na veřejné servery, nedojde k výpadku služby DNS ve vnitřní síti a nebude tak omezena možnost interní komunikace.




**Obrázek 4.1:** Návrh topologie s jedním primárním serverem a dvěma sekundárními servery

Stavebím kamenem pro další návh je představa o počtu a složení serverů a taktéž o jejich uspořádání. Požadavky, které by měl návrh dodržet čítají především:

- Redundanci
- Co největší míru nezávislosti funkce DNS serverů
- Nemožnost podvržení záznamů v databázi DNS záznamů

Topologie (obr. 4.1) vychází ze schématu *hidden master*, tj. skrytý primární server. Pojmem skrytý je zde míněna skutečnost, že primární autoritativní jmenný server není dosažitelný z Internetu, existuje pouze v interní síti. Skrytý primární server si drží zónové soubory, které jsou považovány za originály a o pravosti záznamů v nich obsažených není pochyb. K tomuto serveru je připojeno prostřednictvím interní sítě obecně $N$ sekundárních autoritativních jmenných serverů, které si zkrze přenos zón (*zone transfer*) vytváří vlastní kopie originálních zónových souborů ze skrytého primárního serveru. Sekundární servery jsou připojeny jak do interní sítě, tak do Internetu, tudíž jsou vystaveny vnějším hrozbám a musí tak být brána na zřetel bezpečnost kupříkladu pomocí firewallových pravidel (komunikace na veřejném síťovém rozhraní povolena pouze na portu 53). Návrh řeší pouze autoritativní část DNS, návrhem s rekurzivním resolverem se nezabývá.

## 4.3 Návrh rozhraní pro správu DNS záznamů

Na skrytém primárním serveru poběží vedle služby pro DNS server také HTTP server s REST API, které umožní přidat či odebrat zóny, nebo upravovat jejich obsah. Tento HTTP server pak na základě vstupu od uživatele ukládá informace o zónách a jejich obsazích do SQL databáze. Z obsahu databáze jsou následně

vygenerovány zónové soubory a nasazeny na DNS server. Důležitou součástí je podepisování zón, které, ač jisté implementace DNS serverů to umožňují interně (BIND9), bude prováděno odděleně od služby DNS serveru. Toto oddělení umožní případně změnit software pro DNS server za jiný a zóny se přesto budou podepisovat stále. Dále je nutné kontrolovat platnost podpisů, či podepisovacích klíčů a případně vynutit jejich rotaci. Nad REST API bude webové grafické rozhraní které umožní uživatelovi pohodlnější správu systému.

## 4.4 Dohled

Servery je nutné monitorovat a v případě potřeby vyvodit důsledky. Pro dohled DNS serverů se budou sledovat parametry typu počet dotazů za sekundu, jejich rozdělení podle verze protokolu IP, počet zodpovězených dotazů za sekundu, počet dotazů se statusem SERVFAIL za sekundu a podobně. Tyto parametry dokáží dát základní přehled o funkci služby DNS. Z hlediska funkce samotného serveru je dobré sledovat vytížení procesoru a paměti nebo stav disků.

Dále je účelné mít službu, který v definovaných intervalech posílá dotazy na veřejné rozhraní DNS serveru (simuluje klienty) a měří dobu odezvy a návratový kód odpovědi. V případě neočekávané či žádné odpovědi může přes monitorovací systém vyburcovat alarm a dát vědět administrátorovi.

## 4.5 Použité technologie pro implementaci

V této sekci je popsán využitý software. Vesměs jde o software s otevřeným zdrojovým kódem.

### 4.5.1 Operační systém

Celý vývoj i nasazení probíhá na Red Hat distribucích GNU/Linux. Zkušební servery běží na systému Rocky Linux 9, což je v podstatě klon Red Hat Enterprise Linux 9, akorát volně přístupný. Alternativou je třeba CentOS, doposud s oblibou nasazovaný v serverové oblasti. Ač je vývoj omezený na Red Hat distribuce, s malým úsilím není problém vyvíjenou aplikaci nasadit i na jiných distribucích středního proudu, tj. distribuce na založené především na Debianu, na kterých všechny potřebné balíky jsou taktéž k nalezení. Výběr distribuce tak nehraje kritickou roli v tom smyslu, že by vybraná distribuce oplývala konkrétní funkcemi, které by ostatní distribuce postrádaly.

### 4.5.2 DNS server

Pro zajištění DNS služby je vybrán DNS server BIND9 [36], dostupný ve všech větších linuxových distribucích a také v podobě uvolněného zdrojového kódu. BIND zajišťuje řadu funkcí, v závislosti na obsahu konfiguračního souboru *named.conf*, lze implementovat autoritativní jmenný server (jak primární tak sekundární, což lze určit pro každou jednotlivou zónu zvlášť, zdali je pro ni server primární či sekundární) nebo cache server. BIND9 má plnou podporu DNSSEC a zvládá automatické podepisování zón, což je ale vlastnost, jejíž použití v plánu není. BIND se nastavuje klíčovými konfiguračními soubory *named.conf* a *named_zones.conf*, obvykle k nalezení v adresáři */etc*. Soubor *named.conf* obsahuje obecnou konfiguraci jako komu je povoleno se dotazovat, pomocí kterého protokolu má probíhat komunikace (TCP nebo UDP), zda-li je povolen rekurzivní překlad, a spousta dalších nastavení. V souboru *named_zones.conf* jsou pak definovány jednotlivé zóny a jejich vztah k serveru (jestli je pro ně primární či sekundární, v případě sekundárního serveru pak ještě přibude adresa primárního serveru, ze kterého se obsah zóny zkopíruje). Samotný obsah zón je uložen v zónových souborech (v Red Hat distribucích je výchozí umístění */var/named*) a má standardní formát zónových souborů.

### 4.5.3 Systém pro správu DNS

Na vývoj aplikace budou užity různé technologie jak pro stranu serveru, tak pro stranu klienta.

#### Backend

Pro vývoj funkcí na straně serveru (backend) bude použit programovací jazyk Go [50], z dílny Google, uvolněný jako open source. Za vývojem jazyka stojí mimo jiné legenda Ken Thompson, tvůrce jazyka B, předchůdce jazyka C. Go je malý jazyk, který nachází uplatnění v oblasti webových aplikací a v úlohách vhodných pro paralelizaci. Pro paralelní běh funkce stačí v Go napsat klíčové slovo *go* před volání dané funkce a volaná funkce se vykonává nezávisle na funkci, ze které byla vyvolána. V základu má Go ve standardní knihovně širokou škálu funkcionalit, z nichž pro účely vyvíjené aplikace jsou hlavní: HTTP server, práce se soubory a práce s SQL databází.

Go je zástupcem kompilovaných jazyků. Typický příklad programu v Go je následující kód:

```
1 package main
2
3 import (
4     "fmt"
```

```
)

func main() {
    fmt.Printf("qui bibit, sanctum est\n")
}
```

**Zdrojový kód 4.1:** Příklad programu v jazyce Go

Jazyk Go je velmi silně typový, bez explicitního přetypování nelze vynásobit ani proměnou typu float s proměnnou typu int. To sice může působit poněkud otravně při rychlém prototypování, avšak ještě před překladem se takto dá vyvarovat řadě chyb. Go obsahuje automatickou správu paměti, tudíž není nutné žádat o alokaci paměti, jako je tomu například u programování v C. Za poznámku stojí ještě velikost přeloženého programu, která u příkladu výše činí 1,8 MB. Jen pro porovnání, totožný program v jazyce C má po překladu (přelozeno v gcc verze 13.2.1 bez dalších voleb) 20 kB.

#### Frontend

Pro možnost prohlížení záznamů a jejich úpravy z prostředí webového prohlížeče bude použit framework Angular. Angular je open-source framework vyvinutý Googlem a slouží pro vývoj frontendu webových aplikací. Aplikace v Angularu se skládá z tzv. komponent, kde každá komponenta má svou strukturu danou HTML šablonou, CSS šablonou a její chování je popsáno metodami v třídě dané komponenty v Typescriptu. Tento přístup je pohodlný z hlediska tvorby uživatelského rozhraní, protože HTML šablona umožňuje pomocí rozšíření Angularu zanést rozhodovací logiku a vytvářet tak dynamické stránky.

### 4.5.4 Dohledový systém

Integrace do monitoringu bude prostřednictvím instalace programu *zabbix-agent*, což je malý program, který posílá vybrané metriky na Zabbix server, kde jsou vyhodnocovány. Zabbix umí jak kreslit časové grafy, tak sledovat stav systému a reagovat na něj skrze třeba emailové upozornění. Data dokáže agent posílat pomocí SNMP, nebo vlastním protokolem. Zabbix je zvolen mimo jiné proto, že je již v Algotechu nasazen, takže dohled nových serverů bude spočívat pouze v instalaci Zabbix agentů a v nastavení sledovaných parametrů na straně Zabbix serveru.

## 4.6 Instalace a konfigurace BIND9

Instalace serveru BIND9 v distribucích založených na Red Hat [51] probíha stylem:

```
# dnf install bind bind-utils
```

Balíček *bind* obsahuje službu jmenného serveru *named*, balíček *bind-utils* pak přináší různé další nástroje zajišťující funkce jako diagnózu a administraci serveru, validaci konfiguračních a zónových souborů.

Konfigurace probíhá skrze úpravu textového souboru */etc/named.conf* a liší se v závislosti na tom, zda-li je záměrem nastavit autoritativní nebo cache server.

### 4.6.1 Autoritativní servery

Obsah konfiguračního souboru je v zásadě všeříkající. Takřka vše je v části *options*. Klíčová jsou nastavení, na kterém portu naslouchá kterým adresám IP a také specifikace, na které IP adresy bude posílat upozornění o změnách v zóně (*notify; also-notify*) a pro které adresy je povolen přenos zón (*allow-transfer*). Rekurzivní překlad je pro autoritativní servery zakázán. Ve zbytku jsou specifikovány spíš provozní vlastnosti samotného serveru, jako je logování a statistiky, které budou vhodné pro dohled.

```
options {
    minimal-responses no;
    directory "/var/named";
    listen-on port 53 {
        any;
    };
    listen-on-v6 port 53 {
        any;
    };
    querylog no;
    recursion no;
    transfer-format one-answer;
    max-transfer-time-in 60;
    notify yes;
    also-notify { 10.10.10.11; };
    allow-transfer { 10.10.10.11; };
};

logging {
    channel sys {
        file "/var/log/named/sys.log" versions 5 size 100m;
        severity info;
        print-time yes;
    };

    channel audit_log {
        file "/var/log/named/audit.log" versions 5 size 100m;
        severity debug;
        print-time yes;
    };

```

```
32      channel query_log {
33          file "/var/log/named/query.log" versions 5 size 100m;
34          severity debug 3;
35      };
36
37      category default {
38          sys;
39      };
40
41      category general {
42          sys;
43      };
44
45      category queries {
46          query_log;
47      };
48
49      category xfer-out {
50          audit_log;
51      };
52  };
53
54  include "/etc/named_zones.conf";
```

**Konfigurace 4.2:** Soubor /etc/named.conf pro primární jmenný server, uvedené IP adresy mají ilustrační charakter

Obsah souboru *named.conf* pro sekundární servery je prakticky totožný až na to, že jsou z konfigurace vyškrtnuty direktivy *notify*, *also-notify* a *allow-transfer*. Primární a sekundární servery se liší především v definicích zón, které jsou v souboru */etc/named_zones.conf*.

Soubor */etc/named_zones.conf* se liší v závislosti na tom, jestli je pro primární nebo pro sekundární server následovně (pouze příklad pro zónu *nevim.to.*):

```
1  zone "nevim.to" {
2      type master;
3      file "master/nevim.to.zone.signed";
4  };
```

**Konfigurace 4.3:** Soubor /etc/named_zones.conf pro primární server

```
1  zone "nevim.to." {
2      type slave;
3      masters { 10.10.10.10; };
4      allow-notify { 10.10.10.10; };
5      file "slave/nevim.to.zone.signed";
6  };
```

**Konfigurace 4.4:** Soubor /etc/named_zones.conf pro sekundární server

V definici zóny pro primární server je pouze stanoveno, že pro danou zónu je primárním serverem a dále je uvedena cesta k zónovému souboru. V definici pro sekundární server je především uvedena IP adresa primárního serveru, odkud se má zóna přenést. Ukázka zónového souboru pro zónu *nevim.to.* je zde:

```
1  nevim.to.                10800    IN  SOA ns.nevim.to. admin.nevim.
       to. (
2                                           2023190900
3                                           10800
4                                           3600
5                                           2592000
6                                           10800
7                                           )
8
9  nevim.to.                10800    IN NS   ns.nevim.to.
10
11 ns.nevim.to.             10800    IN A    10.0.0.11
12
13 test.nevim.to.           10800    IN A    127.0.0.1
```

**Konfigurace 4.5:** Soubor /var/named/master/nevim.to.zone

### 4.6.2 Cache servery

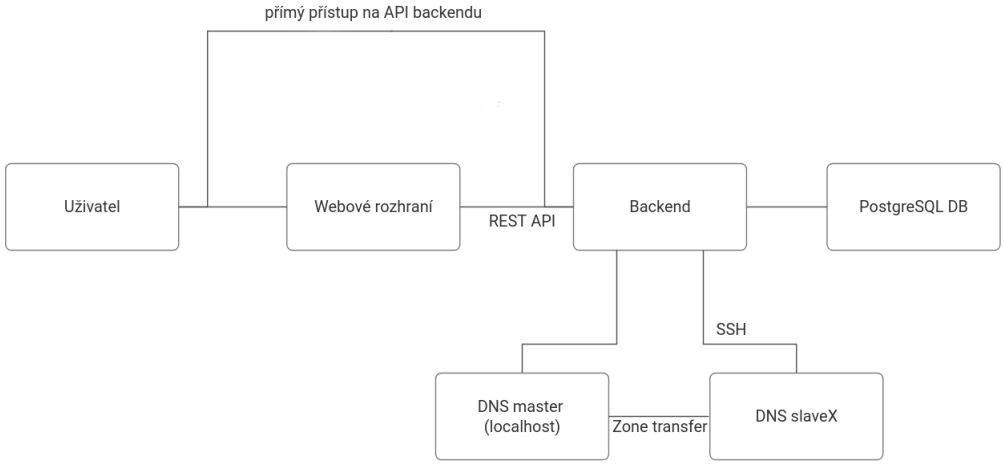
Navrhovaná topologie doposud nezmiňovala cache servery, sloužící k vybavení dotazů od klientů. Měl-li být přítomen i cache server, pak je konfigurace nejsnazší, protože není třeba uvažovat zónové soubory a v zásadě stačí zapnout rekurzivní překlad a DNSSEC validaci. Na vyvíjenou aplikaci to nebude mít žádný vliv, protože ta cílí na správu záznamů na autoritativních serverech, o kterých bude řeč především.

Tímto výčtem konfiguračních souborů je prakticky dokončena nutná práce se serverem BIND9. O vše další, včetně možné překonfigurace, se bude starat nástroj Algoname.

## 4.7 Implementace nástroje Algoname

Prostředník mezi administrátorem a jmennými servery bude aplikace, umožňující upravovat DNS záznamy, vytvořit zónové soubory a propagovat je na server, podepisovat zóny a hlídat platnosti podpisů dle stanovené expirace. Nástroj sestává z několika částí, které budou zevrubně popsány.

Logické schéma aplikace je znázorněno na obr. 4.2. Jádro aplikace je tvořené programem v jazyce Go, jež implementuje potřebné úkony pro ovládání DNS serverů. Rozhraní pro ovládání programu je skrze REST API. Jádro je napojeno




**Obrázek 4.2:** Blokové schéma aplikace

na PostgreSQL databázi, která ukládá spravované zóny a jejich obsah. Pokud je jádru vyslán požadavek o propagaci změn v zóně na jmenný server, pak ze záznamů v databázi pro danou zónu vytvoří zónový soubor, ověří jeho syntaktickou správnost, podepíše a nahraje na primární server. Dále by měl hlídat platnosti podpisů a rotaci podepisovacích klíčů.

Pro uživatelskou přívětivost je na REST API jádra aplikace napojeno webové rozhraní, mající za cíl implementovat intuitivní a snadné úpravy zón.

### 4.7.1 Jádro aplikace (Backend)

S nahlížením na komponenty aplikace jako na černé skříňky s jasně definovanými vstupy a výstupy se po backendu požadují funkce pro:

- zobrazení spravovaných zón,
- přidání zóny,
- odebrání zóny,
- zobrazení všech záznamů v zóně,
- přidání záznamu do zóny,
- odebrání záznamu ze zóny,
- úprava stávajícího záznamu v zóně,
- tvorba zónového souboru pro danou zónu,

- nahrání souborů na server (jmenný, BIND),
- přihlášení se do aplikace.

Pro uvedené operace jsou definovány unikátní metody v URL rozhraní. Se zachovaným pořadím předešlého výčtu jsou to metody (současně uvedené s HTTP metodou dotazu, na kterou daná metoda reaguje):

- /api/dns/zones (GET),
- /api/dns/zones/add-zone (POST),
- /api/dns/zones/del-zone (POST),
- /api/dns/rrs (POST),
- /api/dns/add-rr (POST),
- /api/dns/del-rr (POST),
- /api/dns/mod-rr (POST),
- /api/dns/create-configs (POST),
- /api/dns/propagate (POST),
- /api/login (POST).

Program je rozdělen do několika souborů, pro snazší orientaci a modularitu, pro případný další vývoj. Pohled na zjednodušený souborový systém zdrojových kódů dává následující strom.

```
algoname/
├── db/
│   └── schmema_*.sql
├── dnsdb/
│   ├── db.go
│   ├── models.go
│   └── queries_*.sql.go
├── queries/
│   └── queries_*.sql
├── handlers/
├── middleware/
├── scripts/
├── utils/
└── dist/
```

```
├── main.go
├── zones.go
└── sqlc.yaml
```

V souboru *main.go*, který mimo jiné sdružuje ostatní soubory dohromady a tvoří celek, je pomocí frameworku Gin definováno REST rozhraní s výše popsanými funkcemi. Při definování rozhraní je nutné ke každému indentifikátoru přidat obslužnou funkci na straně serveru, která vykoná požadované operace. Tyto obslužné funkce (*handlers*) jsou definovány v souborech z adresáře *handlers/*.

Pro přístup do PostgreSQL databáze je v aplikaci použit překladač, který z *.sql* souborů, obsahující definice tabulek (*db/schema.sql*) a SQL dotazů (*queries/queries.sql*) vytvoří *.go* soubory s funkcemi podle definovaných dotazů. Tyto funkce se následně importují do hlavního programu a umožňují pohodlnější práci s databází na bázi volání funkcí s parametry a obdržení odpovědí. V celém programu se pak programátor nemusí s čistými SQL dotazy dále zabývat. Veškerá konfigurace sqlc je obsažena v souboru *sqlc.yaml* a jsou v ní uvedeny informace o tom, ke které databázi se má připojovat s jakými přihlašovacími údaji, o jaký typ databáze se jedná (na výběr jsou PostgreSQL, MySQL a SQLite) a kde jsou další soubory, obsahující definice tabulek a dotazů. Překladač sqlc ([52], [53]) se nainstaluje např. pomocí vestavěného příkazu překladače pro jazyk Go:

```
~ go install github.com/sqlc-dev/sqlc/cmd/sqlc@latest
```

Příklad funkce sqlc je v následujících úryvcích. V souboru *db/schema_zones.sql* popis tabulky *dns_zones* (o skladbě databáze bude řeč dále):

```
CREATE TABLE dns_zones (
    z_id        SERIAL NOT NULL PRIMARY KEY,
    z_name      TEXT NOT NULL,
    z_dnssec    BOOLEAN,
    z_cipher    TEXT,
    z_keys      TEXT,
    z_fwd       BOOLEAN,
    z_cre_t     TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

**Konfigurace 4.6:** Úryvek souboru *db/schema_zones.sql*

Dotazy vztažené k této tabulce jsou v souboru *queries/queries_zones.sql*. Dotaz pro získání všech zón z databáze je:

```
-- name: GetZones :many
SELECT z_id,z_name,z_dnssec,z_cipher,z_fwd,z_cre_t FROM dns_zones;
```

**Konfigurace 4.7:** Úryvek souboru *queries/queries_zones.sql*

Řádek nad samotným SQL dotazem je pomocná informace pro sqlc, jak se má funkce zajišťující tento dotaz jmenovat, kolik má výstupů. To, kolik má mít funkce vstupů (v tomto případě žádný), si překladač zjistí sám ze struktury dotazu.

Přetavení těchto souborů do použitelného balíku v Go stačí zavolat v adresáři se souborem *sqlc.yaml* (a mít definovanou proměnnou prostředí GOPATH):

```
$ $GOPATH/bin/sqlc generate
```

Výsledkem jsou soubory v adresáři *dnsdb/*: *db.go*, *models.go* a *queries_*.sql.go*. První jmenovaný je prostředníkem mezi knihovnou sql a tímto balíkem. Druhý soubor obsahuje definice pomocných datavých typů a funckí. Třetí skupina souborů pak obsahuje funkce vykonávající SQL dotazy, výsledek pro funci dotazující se na všechny zóny je:

```
func (q *Queries) GetZones(ctx context.Context) ([]GetZonesRow,
    error) {
    rows, err := q.db.QueryContext(ctx, getZones)
    if err != nil {
        return nil, err
    }
    defer rows.Close()
    var items []GetZonesRow
    for rows.Next() {
        var i GetZonesRow
        if err := rows.Scan(
            &i.ZID,
            &i.ZName,
            &i.ZDnssec,
            &i.ZCipher,
            &i.ZFwd,
            &i.ZCreT,
        ); err != nil {
            return nil, err
        }
        items = append(items, i)
    }
    if err := rows.Close(); err != nil {
        return nil, err
    }
    if err := rows.Err(); err != nil {
        return nil, err
    }
    return items, nil
}
```

**Konfigurace 4.8:** Úryvek ze souboru *dnsdb/queries.sql.go*

Tuto a další funkce definované v tomto souboru lze po importování do zdrojového kódu hned volat. Tento přístup v mnohém eliminuje jinak monotónní a potenciálně chyby zanechávající práci.

Obsluha REST rozhraní je implementována pomocí frameworku Gin [54]. Gin umožňuje snadnou implementaci webového rozhraní pomocí definice cest (tzv. *routes*), které představují metody v URL, o kterých byla řeč dříve. Příklad využití frameworku Gin pro definované rozhraní je:

```
func SetupRouter(db *sql.DB, dbctx context.Context, dbqr *dnsdb.Queries) *gin.Engine {
    r := gin.Default()
    r.Use(cors.Default())
    server := handlers.NewServer(db, dbctx, dbqr)

    r.Use(static.Serve("/", static.LocalFile("dist/algoname-front/browser/", true)))

    router := r.Group("/api")
    router.POST("/login", server.LoginHandler)

    authorized := r.Group("/api/dns")
    authorized.Use(middleware.JwtAuthMiddleware())

    authorized.GET("/zones", server.ZonesHandler)
    authorized.POST("/zones/add-zone", server.AddZoneHandler)
    authorized.POST("/zones/del-zone", server.DelZoneHandler)

    authorized.POST("/rrs", server.RrsHandler)
    authorized.POST("/rrs/add-rr", server.AddRrHandler)
    authorized.POST("/rrs/del-rr", server.DelRrHandler)
    authorized.POST("/rrs/mod-rr", server.ModRrHandler)

    authorized.POST("/create-configs", server.CreateConfigsHandler)
    authorized.POST("/propagate", server.PropagateHandler)

    return r
}
```

**Zdrojový kód 4.9:** Úryvek ze souboru *main.go* reprezentující práci s frameworkem Gin

Kód je hoden komentáře v obecné rovině: frameworku se předloží soubory pro vykreslení stránky (*.*.html, *.css, *.js* ) a dále je jsou jen definovány operace, které se vykonají po příchodu HTTP dotazu dané metody.

Jádro pro komunikaci se servery využívá shellové skripty, popsané v 4.7.5.

### 4.7.2 Databáze

Ukládání informací o spravovaných zónách a jejich obsahu je zajištěno databází PostgreSQL. Databáze je tvořena, z hlediska záznamů týkajících se DNS, dvěmi tabulkami a jednou tabulkou pro ukládání uživatelů oprávněných do aplikace vstou-

pit.

První tabulka *dns_zones* obsahuje názvy spravovaných zón, jejich ID (pro interní odkazování se), datum podepsání zóny a datum expirace podpisu.

```
dns=> \d dns_zones
                                        Table "public.dns_zones"
  Column  |            Type             | Collation | Nullable |

----------+-----------------------------+-----------+----------+
-
 z_id     | integer                     |           | not null |
 z_name   | text                        |           | not null |
 z_dnssec | boolean                     |           |          |
 z_cipher | text                        |           |          |
 z_keys   | text                        |           |          |
 z_fwd    | boolean                     |           |          |
 z_cre_t  | timestamp without time zone |           |          |
Indexes:
    "dns_zones_pkey" PRIMARY KEY, btree (z_id)
```

**SQL tabulka 4.10:** dns_zones (výpis zkrácen o sloupec *Default*)

Druhá tabulka je tvořena záznamy ze všech zón.

```
dns=> \d dns_rrs
                                        Table "public.dns_rrs"
 Column  |            Type             | Collation | Nullable |
---------+-----------------------------+-----------+----------+
 r_id    | integer                     |           | not null |
 r_z_id  | integer                     |           |          |
 r_name  | text                        |           | not null |
 r_ttl   | integer                     |           |          |
 r_class | classenum                   |           |          |
 r_type  | typeenum                    |           |          |
 r_data  | text                        |           |          |
 r_cre_t | timestamp without time zone |           |          |
Indexes:
    "dns_rrs_pkey" PRIMARY KEY, btree (r_id)
```

**SQL tabulka 4.11:** dns_rrs (výpis zkrácen o sloupec *Default*)

Poslední tabulka shromažďuje uživatele oprávněné vstoupit do aplikace.

```
dns=> \d dns_users
   Column    |            Type             |
-------------+-----------------------------+
 u_id        | integer                     |
 u_name      | text                        |
 u_pass_hash | text                        |
 z_cre_t     | timestamp without time zone |
Indexes:
```

```
9     "dns_users_pkey" PRIMARY KEY, btree (u_id)
```

**SQL tabulka 4.12:** dns_users (výpis omezen jen na sloupce *Column* a *Type*)

Obsluha databáze je přes zmíněnou vrstvu funkcí vytvořenovu nástrojem *sqlc* na základě dotazů v souboru *queries/queries.sql*.

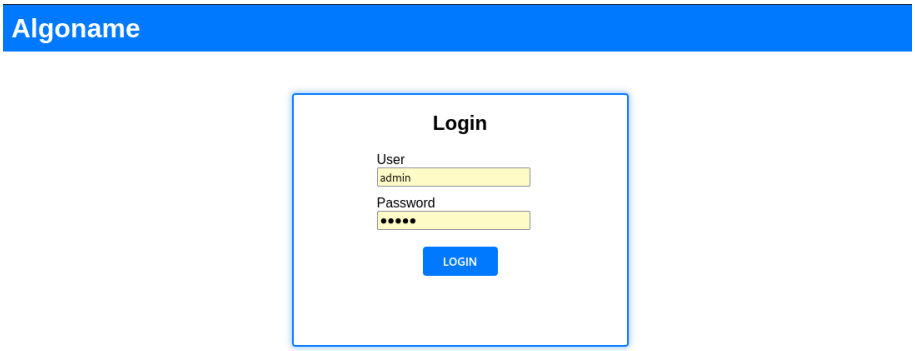
### 4.7.3 Podepisování

Podepisování zón se v jádru věci děje pomocí skriptu *scripts/sign-zonefile.sh*, který k tomu účelu využívá nástroj dnssec-signzone, jež je součástí balíku *bind-utils* a tudíž součástí serveru BIND.

### 4.7.4 Frontend

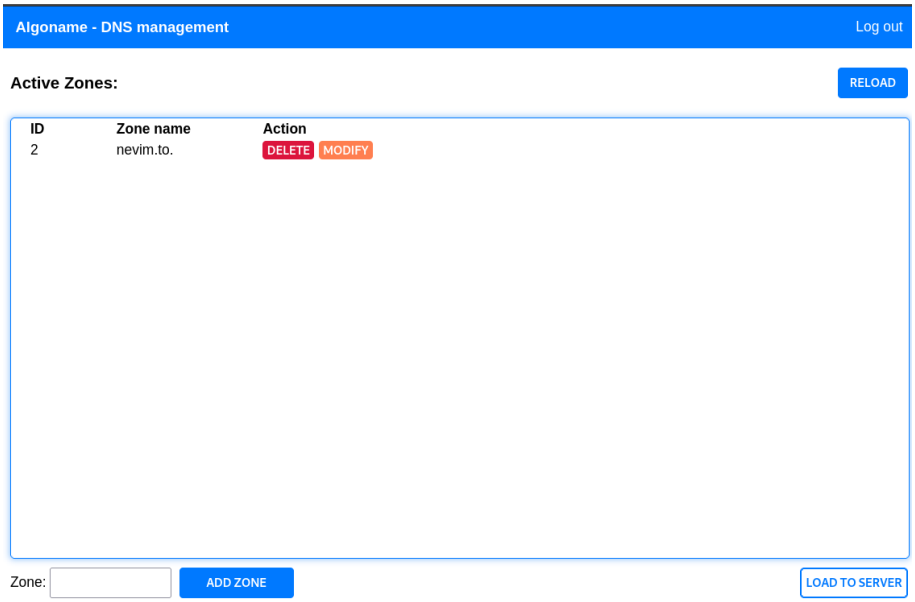
Prezentaci databáze uživateli zaopatří frontend napsaný v Angularu. Grafické rozhraní je dostupné po zadání adresy a příslušného portu, na kterém HTTP server naslouchá ve webovém prohlížeči. Rozhraní má tři části:

1. Přihlašovací obrazovka (obr. 4.3)
2. Přehled zón (obr. 4.4)
3. Přehled záznamů v zóně (obr. 4.5)

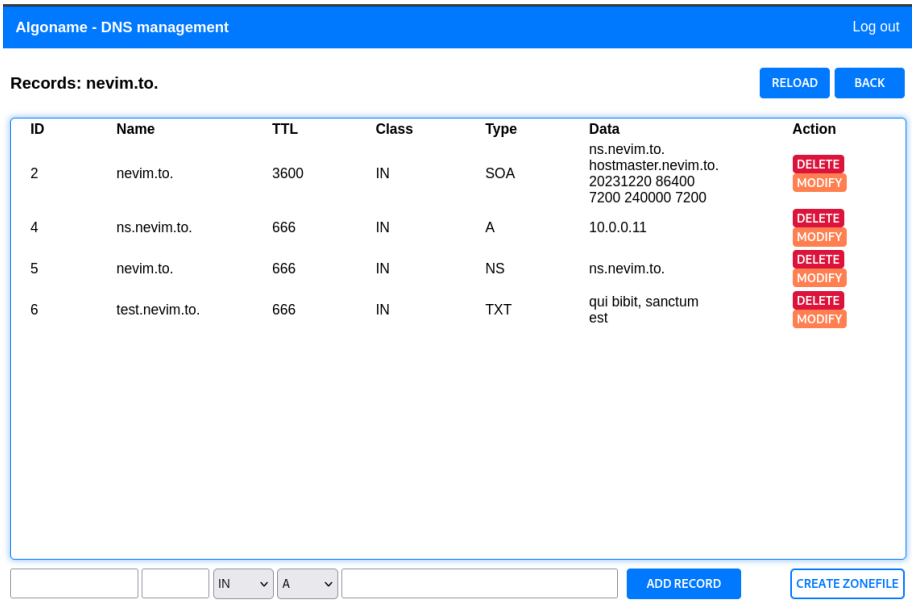



**Obrázek 4.3:** Přihlašovací stránka

Grafické rozhraní nepřináší žádnou vylepšenou či novou funkcionalitu, jen umožňuje v rámci úpravy jednotek záznamů pohodlnější práci pro administrátora.




**Obrázek 4.4:** Stránka s přehledem zón




**Obrázek 4.5:** Stránka s přehledem záznamů v zóně

### 4.7.5 Napojení na DNS servery

Po příchodu HTTP dotazu na */api/dns/propagate* jsou vegenerované konfigurace *named.conf*, *named_zones.conf* pomocí skriptu *propagate.sh* poslány přes SSH na jednotlivé sekundární servery, na kterých se posléze spustí načtení nové konfigurace pro službu *named*.

### 4.7.6 Bezpečnost

Pro modifikaci zón v databázi je nutné být přihlášen. Aplikace v současnosti neumožňuje registraci nových uživatelů, oprávněné uživatele je nutné přidat do tabulky *dns_users* v databázi. Ta obsahuje jméno uživatele a hash hesla (PBKDF2). Pro zpřístupnění metod pro úpravu zón, je nutné pomocí API poslat jméno uživatele a hash zadaného hesla na */api/login* serveru. Server v případě úspěšného přihlášení vrátí JWT token, kterým se klient autentizuje. Token má určitou dobu platnosti po jejímž uplynutí expiruje. Při přístupu k metodám pro prohlížení či úpravu zón se do HTTP hlavičky požadavku vkládá: "Authorization: Bearer <přijatý token>", který server při každém požadavky vždy překontroluje.

Na serveru není implementováno HTTPS, tedy HTTP komunikace mezi klientem a serverem je nešifrovaná. Jako prozatímní opatření by se mohla jevit technika přesměrování portu. Ta spočívá v tvorbě (systémového) uživatele na primárním serveru, který má co nejmenší práva. Na se klienti hlásí pomocí SSH způsobem (příklad pro případ, kdy aplikace naslouchá na portu 8000):

```
1 ssh -L 8000:127.0.0.1:8000 user@dns-master
```

Tímto způsobem služba ssh naslouchá na portu 8000 na stroji klienta a vše, co na tento port příjde, odešle přes vytvořený tunel na adresu 127.0.0.1:8000 (localhost, aplikace běží na primárním jmenném serveru) ovšem ze stroje dns-master. Šifrování provozu je v této technice plně přenecháno na SSH.

Dalším článkem je přihlašování primárního serveru na sekundární pro nahrání konfiguračních souborů. Na sekundárních serverech bude povoleno přihlašování pouze přes SSH klíč pro optimální šifru, dostatečné délky (Ed25519; RSA 4096 b).

Poslední částí je ochrana serverů, především těch, jež jsou vystaveny do Internetu. Nutností je striktní nastavení pravidel na firewallu (služba *firewalld* nebo rovnou *iptables*). Pro veřejná rozhraní sekundárních serverů je možné povolit komunikaci pouze na 53/udp, všechno ostatní by se mělo zahazovat. Co se týče firewallu na rozhraních ve vnitřní síti, i tam je však záhodno povolit komunikaci jen na těch portech, na kterých je opravdu potřeba, tj. 53/udp, 53/tcp, 22/tcp, port pro rozhraní aplikace.

Bezpečnost přenosu zón z primárního serveru je řešena explicitním výčtem IP adres, které mají právo žádat o přenos zón, v konfiguraci BIND serveru.

## 4.8 Shrnutí

Cílem byla tvorba instalace DNS serveru BIND v několika instancích (primární, $N$ sekundárních), dále tvorba nadstavby, která se postará o správu DNS záznamů, s uživatelsky přístupným rozhraním. Aplikace měla dále řešit podepisování zón podle DNSSEC a automaticky řešit expiraci podpisů a klíčů. Velkým kritériem byla bezpečnost, která byla břana na zřetel. Tyto cíle jsou v zásadě splněny, v produkčním nasazení by byl předpoklad implementovaného HTTPS. Práce se zabývala především tvorbou nástroje a integrace do dohledu byla po dohodě z práce vyřazena.

# Kapitola 5

# Testování vlastního řešení

Ověření funkce aplikace probíhalo ve dvou prostředích. Během vývoje aplikace bylo testovacím prostředím skupina virtuálních strojů běžicích na autorově PC. Druhé prostředí bylo již v Algotechu, v podobě taktéž virtuálních strojů, nicméně s možností nastavení veřejných síťových rozhraní směrem do Internetu. Popis testování z obou prostředí bude nyní vyložen.

## 5.1 Testovací stroje na lokálním PC

Pro testovací účely na lokálním stroji byly vytvořeny tři virtuální stroje: *dns-master*, *dns-slave* a *dns-client*. Primární server (*dns-master*) má jedno síťové rozhraní s IP adresou 10.10.10.10. Sekundární server (*dns-slave*) má síťová rozhraní dvě – jedno (10.10.10.11) pro simulaci vnitřní sítě ve které se komunikuje s primárním serverem, druhé pro simulaci veřejného rozhraní (10.0.0.1). Poslední je pak stroj simulující klienta přistupujícího z vnějšku (*dns-client*, 10.0.0.100).

K otestování byla připravena sestava skriptů (v příloze *cli-front/*). Skripty sestávají z HTTP dotazů na rozhraní backendu aplikace pomocí nástroje *curl*. Jsou zde skripty pro vytvoření testovací zóny, přidání, odebrání a modifikaci záznamů, pro vytvoření konfiguračních souborů a pro rozšíření konfigurace na sekundární servery.

Příklad pro vytvoření testovací zóny, v níž je záznam typu SOA, NS, A (adresa NS) a TXT, která se následně podepíše a propíše přímo do služby *named*:

```
1 cli-front/~ ./create-test-zone.sh nevim.to. > /dev/null
2 cli-front/~ ./create-configs.sh > /dev/null
3 cli-front/~ ./propagate.sh > /dev/null
```

**SQL tabulka 5.1:** Příklad tvorby testovacích dat, spouštěno z hostitelského OS

Výsledek se dá zobrazit po doptání na sekundární server ze stroje *dns-client* pomocí nástroje *dig*.

```
[root@dns-client ~]# dig +short @dns-slave nevim.to SOA
ns.nevim.to. hostmaster.nevim.to. 2024010702 86400 7200 240000
    7200
[root@dns-client ~]#
[root@dns-client ~]# dig +short @dns-slave test.nevim.to TXT
"qui" "bibit," "sanctum" "est"
[root@dns-client ~]#
[root@dns-client ~]# dig +short @dns-slave-ext nevim.to RRSIG
DNSKEY 8 2 3600 20240206204020 20240107204020 62676 nevim.to.
    oHF00PmSTZWd0CUDWZ0mMFEWGdpaooQlGtnn+Q/PtyTfrVeZHCK/UwGq
    fwY05B0C+TtUvB3WFY3iDO7PzTfsNbls5bzXMJZZ9QIQLjTjw8a5tz5V
    bAoNZxauvaA0SRjvU14jBTrmDFbe1tb17Fp0rsEmHtO4/AIMXMS3KAoe
    Oy9bA81ZfSfP9McZy2gHdJS6l6wxWhcxNR/QjMkJ9ftYYeG0+ykj79A5
    BkzfcHklI3y7v0lIk5VY4bV6K0gwPmu1jrj+ua3X7Y9gkpo34KVCx5FV
    SbtX92KkOSI+MlJJ2DtnQ+Rn4aguHdb/Ru9e5s+MzfMBTcbUi6TxUpTD QLH9sw
    ==
DNSKEY 8 2 3600 20240206204020 20240107204020 63422 nevim.to.
    m4xpit9sfx1NrNEbCt7hPZQga303FM66lL+yKxGbZY3f0+JIoTUTl7WJ 7
    iAHTwKJZCPotFq/aAmuT6oC4U0rXW61paoVFPTAwJsujc/9cmyTi05p
    nzz385gbR5tJ4j81A3mCoDesWAkFN3bdDv1ph6aDZ33FJRGqt4L7jnb8
    oTAvqf99WjAAyaAo197ofKTWrcnEBvCT31mjrMzS62buV5BBlEeRLJGI +
    FxbLQThY01CKlXnmcdfc9Rja2jO0RTekdg+ye58VNPv1npae3itFEkX
    WpHqvj6LkTQwfyYj5JF+ks1Byw/sBH+wTfLOLiWNVD6FehQPMkgTY908
    YRhlOJBTVd0ah7Ox4GoSvAHlN81pevAi8igedzGvnhoR9FbFXhxDepaq
    Cjvw5YGF0KhhidHkP3RPdnjX2xY5zRlGq/kFtIDyLx4yjowkg7BPb1+v
    ZBZrtVygD7mclsh3C+VF46XHxJzt0AmUFtPOvpgjwe8a2QrHkpgICTou
    dgKLRampiF3aobWOOO+OiJSAtj9v5Ch7A+vObcFYY0BY9NV6Fjj0OCqt
    KDuwh7T+yvflH6tX6uqP+x5CyDwrQVUNC31/kyGdf0hzEJGJr+CclR/s 6
    bdOfW9EJ/IbdUotlZO1nMLjIhQbZx49gX5gOsH5ESvu4o7aEq+2Yh9r
    ra8k7SUNj/o=
NSEC 8 2 3600 20240206204020 20240107204020 62676 nevim.to.
    UxEHkNTxvPwhJWTLI/VXekgEmYB/iyofNPC7hB6ty5IYmhJLCvxxTZ82
    XnKTEDo5s35r0+PqxbI8b9vtpuq9RxbZyvsW22RB9Y8xBQtxmr6QhMqn 7
    o4pITLxyfX7/Dtovx8lEDbzufDrv36s1LptInNr2tzQTj9jutS/IBkY X83+
    MJxSNX7mQ0nIWfz4GKLrOPazZXcwfz+b8WWue0QP5Kf4YOaTgP8r
    szSbuf7I5KvaCTE60iprTZ9J8WCCyxvuBvyMh23/WO08kixVqlRBRmmw
    X0NyYN0qBK2Hx5GrKd1yziirJ3DPsNGY0henxySycIM3QkwD1M0Grxbv 49p2VQ
    ==
NS 8 2 666 20240206204020 20240107204020 62676 nevim.to.
    O3ZIHAglmmoJVLgidzfHJy52S9Ji3ro5FLcSdbHMbuHn4dKwRB0SXrvV
    aioXCl84UlHJfM0MECYsL5qjIe0OLfskI9Zt68/p9NmICOqU+yZKXFhP
    kJh7XujhbWPlpVVEdPe6QGM8qudZiXA3Zox5XO4yE6vbVgRw58YXMArr
    SsAsAWky2LQbUhaNq8CbkQa/UcoNzq0J1hunj5MiVwPy7S4PITHVIX1k
    hIqvaZKytB6kC/QKryOSXNGx71O+1+9YHwL0CSzHiKNj/ia7a7XSb37l GmEg+0
    CGeXTxB6AaA4hjk+n60WO8pvSCc0VM5oUDGvOInHeFExDwI+T6 7lbsYw==
SOA 8 2 3600 20240206204020 20240107204020 62676 nevim.to. sqxbIrj
    +S1EIv1mCEK3pmcRv5bQ5peA98waqDwLSCRHo2psoluXmnDJe
    GbF9oSf3dJdBch6aEAGPVHi02TNVksG04n0OYIwc+/YEUpOnaKtD+8n7
    eRkXizmOQyVQRYAC46ujb9r02Sx07hbg+9mYdqyf2YktXrFpPfVHfvzr
```

```
tfQUo6f3rnCn4R6/r/J1eFd3uyWkjyNSy7UrQT6fHgcbX+81IxiwGMA4 2qcp++
CiYZg0k6Ah/kUjLSWty1jB33SftlSpcCQ+zx2ZPcwXdjSn/438 /jfm/1
EKYR4jRpqX0uk4gFZRADU7tpnt+TCfGQ+TaVRnibOz0kUbplYO x0nK2w==
```

**SQL tabulka 5.2:** Kontrola propsání vytvořených údajů.

Zabezpečení veřejného rozhraní (front) je řešeno pravidly v iptables:

```
[root@dns-slave ~]# iptables -S
-P INPUT DROP
-P FORWARD DROP
-P OUTPUT ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -i back -j ACCEPT
-A INPUT -i front -p tcp -m tcp --dport 53 -j ACCEPT
-A INPUT -i front -p udp -m udp --dport 53 -j ACCEPT
-A INPUT -i front -p tcp -m tcp --sport 53 -j ACCEPT
-A INPUT -i front -p udp -m udp --sport 53 -j ACCEPT
-A INPUT -p icmp -j ACCEPT
[root@dns-slave ~]# ip6tables -S
-P INPUT DROP
-P FORWARD DROP
-P OUTPUT ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -i back -j ACCEPT
-A INPUT -i front -p tcp -m tcp --dport 53 -j ACCEPT
-A INPUT -i front -p udp -m udp --dport 53 -j ACCEPT
-A INPUT -i front -p tcp -m tcp --sport 53 -j ACCEPT
-A INPUT -i front -p udp -m udp --sport 53 -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -p ipv6-icmp -j ACCEPT
```

Povolena je komunikace pouze na portu 53 a komunikace pomocí protokolu ICMP, veškerá snaha o jinou komunikaci je zablokována.

## 5.2 Testovací stroje v Algotechu

Testovací prostředí v Algotechu se lišilo pouze v drobnostech, tj. operační systém byl namísto Rocky Linux 9 použit CentOS 9, nebo sekundární server neměl druhé rozhraní pro veřejnou komunikaci. Na ozkoušení funkce to však nemělo vliv. Testovací stroje byly dva (primární a sekundární), jako klient sloužilo uživatelské PC, které, přihlášené přes VPN, na ně mohlo klást dotazy. Funkce byla totožná jako u testování na lokálních virtuálních strojích.

# Závěr

Jedním cílem práce byl popis a přiblížení principů DNS s bezpečnostními riziky, jejich protiopatření. Tím se zabývaly kapitoly 1 a 2. Kapitola 3 se letmo dotkla současného softwaru pro implementaci DNS a různých řešení přinášející bezpečnostní benefity.

Jádrem práce byla kapitola 4, která řešila další cíl: Navrhnout topologii autoritativních DNS serverů a vytvořit nástroj pro správu zón, těmito servery poskytovanými. Pro topologii bylo vybráno schéma tzv. *hidden master*, ve které je primární jmenný server viditelný pouze ve vnitřní síti a pouze sekundární servery jsou vystaveny některým síťovým rozhraním do veřejné sítě. Primární server spravuje originály zónových souborů a na sekundární servery posílá jejich kopie. Jsou-li z veřejné sítě dostupné pouze sekundární servery, nemůže dojít útokem z vnějšku k narušení originálního zónového souboru.

Na vybrané topologii byl vyvíjen nástroj Algoname, který umožňuje základní úkony se zónami jako je přidání a smazání nebo s jednotlivými záznamy v zónách. Nástroj, bežící na primárním serveru, dále podepisuje zóny, u kterých je to žádané a je tak implementován požadavek na DNSSEC. Nástroj lze ovládat jak pomocí REST API, tak pomocí minimalistického webového rozhraní.

Nástroj jako celek je spíše než funkčním programem vhodným k produkčnímu nasazení software ve velice ranném vývoji, funkční spíše pro ukázku. K vážnějšímu uvažování o nasazení by musel projít ještě notným vývojem.

Navzdory zmíněným nedostatkům však nástroj v kapitole 5 prošel testovacím scénářem, který vytvořil jednu zónu o několika záznamech, kterou podepsal a nahrál na sekundární server. Ověření funkce proběhlo nástrojem *dig*, který prokázal úspěch s nahráním podepsané testovací zóny na sekundární server.

# Příloha

Příloha je tvořena čtyřmi adresáři. Adresář *algoname2/* obsahuje zdrojové kódy vyvíjeného nástroje (*algoname_old/* obsahuje starou verzi). V adresáři *cli-front/* jsou k nalezení testovací skripty. Obsahem adresáře *LaTeX/* jsou zdrojové kódy této práce. Adresář *VMs/* obsahuje skripty pro tvorbu virtuálních strojů pro testovací účely.

# Bibliografie


1. *Distributed denial-of-service attacks on root nameservers* [https://en.wikipedia.org/wiki/Distributed_denial-of-service_attacks_on_root_nameservers]. [B.r.]. [online] Navštíveno 10.6. 2023.
2. *5 of the world's biggest network outages* [https://www.techradar.com/news/5-of-the-worlds-biggest-network-outages]. [B.r.]. [online] Navštíveno 10.6. 2023.
3. *DDoS attacks are powerful enough to cut off an entire country from the internet* [https://www.techradar.com/news/ddos-attacks-are-powerful-enough-to-cut-off-an-entire-country-from-the-internet]. [B.r.]. [online] Navštíveno 10.6.2023.
4. AITCHISON, R. *Pro DNS and BIND 10, A complete reference to DNS and BIND*. Apress, [b.r.]. ISBN 978-1-449-30519-2. počet stran: 722.
5. C. LIU, P. Albitz. *DNS and BIND, 5th Edition*. O'Reilly Media, Inc., [b.r.]. ISBN 9780596100575. počet stran: 616.
6. *How the Internet was born: from the ARPANET to the Internet* [https://theconversation.com/how-the-internet-was-born-from-the-arpanet-to-the-internet-68072]. [B.r.]. [online] Navštíveno 16.6.2023.
7. KARP, P.M. *Standardization of host mnemonics* [Internet Requests for Comments]. RFC Editor, 1971-09. RFC, 226. RFC Editor. ISSN 2070-1721. Dostupné také z: http://www.rfc-editor.org/rfc/rfc226. http://www.rfc-editor.org/rfc/rfc226.
8. KARP, P.M. *Proffered set of standard host names* [Internet Requests for Comments]. RFC Editor, 1971-10. RFC, 247. RFC Editor. ISSN 2070-1721. Dostupné také z: http://www.rfc-editor.org/rfc/rfc247. http://www.rfc-editor.org/rfc/rfc247.
9. MILLS, D.L. *Internet name domains* [Internet Requests for Comments]. RFC Editor, 1981-09. RFC, 799. RFC Editor. ISSN 2070-1721. Dostupné také z: http://www.rfc-editor.org/rfc/rfc799. http://www.rfc-editor.org/rfc/rfc799.

10. MOCKAPETRIS, P. *Domain names: Concepts and facilities* [Internet Requests for Comments]. RFC Editor, 1983-11. RFC, 882. RFC Editor. ISSN 2070-1721. Dostupné také z: http://www.rfc-editor.org/rfc/rfc882. http://www.rfc-editor.org/rfc/rfc882.

11. MOCKAPETRIS, P.V. *Domain names: Implementation specification* [Internet Requests for Comments]. RFC Editor, 1983-11. RFC, 883. RFC Editor. ISSN 2070-1721. Dostupné také z: http://www.rfc-editor.org/rfc/rfc883. http://www.rfc-editor.org/rfc/rfc883.

12. MOCKAPETRIS, P. *Domain names - concepts and facilities* [Internet Requests for Comments]. RFC Editor, 1987-11. STD, 13. RFC Editor. ISSN 2070-1721. Dostupné také z: http://www.rfc-editor.org/rfc/rfc1034. http://www.rfc-editor.org/rfc/rfc1034.

13. MOCKAPETRIS, P. *Domain names - implementation and specification* [Internet Requests for Comments]. RFC Editor, 1987-11. STD, 13. RFC Editor. ISSN 2070-1721. Dostupné také z: http://www.rfc-editor.org/rfc/rfc1035. http://www.rfc-editor.org/rfc/rfc1035.

14. RADER, Ross Wm. *One History of DNS*. 2001-Duben. Tech. zpr. Dostupné také z: http://www.byte.org/one-history-of-dns.pdf. http://www.byte.org/one-history-of-dns.pdf.

15. 3RD, D. Eastlake; PANITZ, A. *Reserved Top Level DNS Names* [Internet Requests for Comments]. RFC Editor, 1999-06. BCP, 32. RFC Editor. ISSN 2070-1721. Dostupné také z: https://www.rfc-editor.org/rfc/rfc2606. https://www.rfc-editor.org/rfc/rfc2606.

16. HUSTON, G. *Management Guidelines and Operational Requirements for the Address and Routing Parameter Area Domain ("arpa")* [Internet Requests for Comments]. RFC Editor, 2001-09. BCP, 52. RFC Editor. ISSN 2070-1721. Dostupné také z: https://www.rfc-editor.org/rfc/rfc3172. https://www.rfc-editor.org/rfc/rfc3172.

17. *File:Example of an iterative DNS resolver.svg* [https://en.m.wikipedia.org/wiki/File:Example_of_an_iterative_DNS_resolver.svg]. [B.r.]. [online] 20.11.2023.

18. VIXIE, Paul; THOMSON, Susan; REKHTER, Yakov; BOUND, Jim. *Dynamic Updates in the Domain Name System (DNS UPDATE)* [Internet Requests for Comments]. RFC Editor, 1997-04. RFC, 2136. RFC Editor. ISSN 2070-1721. Dostupné také z: http://www.rfc-editor.org/rfc/rfc2136. http://www.rfc-editor.org/rfc/rfc2136.

19. VIXIE, Paul. *Extension Mechanisms for DNS (EDNS0)* [Internet Requests for Comments]. RFC Editor, 1999-08. RFC, 2671. RFC Editor. ISSN 2070-1721. Dostupné také z: http://www.rfc-editor.org/rfc/rfc2671. http://www.rfc-editor.org/rfc/rfc2671.

20. *List of DNS record types* [https://en.wikipedia.org/wiki/List_of_DNS_record_types]. [B.r.]. [online] 20.11.2023.

21. *What is Anycast? | How does Anycast work?* [https://www.cloudflare.com/learning/cdn/glossary/anycast-network/]. [B.r.]. [online] 3.1.2024.

22. *root-servers.org* [https://www.iana.org/domains/root/servers]. [B.r.]. [online] 3.1.2024.

23. *Root Servers* [https://root-servers.org/]. [B.r.]. [online] 3.1.2024.

24. *The Tropical Island With the Hot Domain Name* [https://archive.ph/2023.08.31-132124/https://www.bloomberg.com/news/articles/2023-08-31/ai-startups-create-digital-demand-for-anguilla-s-website-domain-name]. [B.r.]. [online] 3.1.2024.

25. ATKINS, D.; AUSTEIN, R. *Threat Analysis of the Domain Name System (DNS)* [Internet Requests for Comments]. RFC Editor, 2004-08. RFC, 3833. RFC Editor. ISSN 2070-1721. Dostupné také z: http://www.rfc-editor.org/rfc/rfc3833. http://www.rfc-editor.org/rfc/rfc3833.

26. R. CHANDRAMOULI, Scott Rose. *Secure Domain Name System (DNS) Deployment Guide* [Internet Requests for Comments]. National Institute of Standards a Technology, 2013-09. Tech. zpr. U.S. Department of Commerce. Dostupné také z: https://csrc.nist.gov/pubs/sp/800/81/2/final. https://csrc.nist.gov/pubs/sp/800/81/2/final.

27. *What is a DNS flood? | DNS flood DDoS attack* [https://www.cloudflare.com/learning/ddos/dns-flood-ddos-attack/]. [B.r.]. [online] 23.11.2023.

28. *DNS amplification attack* [https://www.cloudflare.com/learning/ddos/dns-amplification-ddos-attack/]. [B.r.]. [online] 23.11.2023.

29. *What is DNS cache poisoning? | DNS spoofing* [https://www.cloudflare.com/learning/dns/dns-cache-poisoning/]. [B.r.]. [online] 23.11.2023.

30. *The global DNS hijacking threat* [https://www.cloudflare.com/learning/security/global-dns-hijacking-threat/]. [B.r.]. [online] 28.11.2023.

31. ARENDS, R.; AUSTEIN, R.; LARSON, M.; MASSEY, D.; ROSE, S. *DNS Security Introduction and Requirements* [Internet Requests for Comments]. RFC Editor, 2005-03. RFC, 4033. RFC Editor. ISSN 2070-1721. Dostupné také z: http://www.rfc-editor.org/rfc/rfc4033. http://www.rfc-editor.org/rfc/rfc4033.

32. ARENDS, R.; AUSTEIN, R.; LARSON, M.; MASSEY, D.; ROSE, S. *Resource Records for the DNS Security Extensions* [Internet Requests for Comments]. RFC Editor, 2005-03. RFC, 4034. RFC Editor. ISSN 2070-1721. Dostupné také z: http://www.rfc-editor.org/rfc/rfc4034. http://www.rfc-editor.org/rfc/rfc4034.

33. ARENDS, R.; AUSTEIN, R.; LARSON, M.; MASSEY, D.; ROSE, S. *Protocol Modifications for the DNS Security Extensions* [Internet Requests for Comments]. RFC Editor, 2005-03. RFC, 4035. RFC Editor. ISSN 2070-1721. Dostupné také z: http://www.rfc-editor.org/rfc/rfc4035. http://www.rfc-editor.org/rfc/rfc4035.

34. HOFFMAN, P. *DNS Security Extensions (DNSSEC)* [Internet Requests for Comments]. RFC Editor, 2023-02. BCP, 237. RFC Editor. ISSN 2070-1721. Dostupné také z: https://www.rfc-editor.org/rfc/rfc9364. https://www.rfc-editor.org/rfc/rfc9364.

35. *Comparison of DNS server software* [https://en.wikipedia.org/wiki/Comparison_of_DNS_server_software]. [B.r.]. [online] 28.12.2023.

36. *BIND9, Versatile, classic, complete name server software* [https://www.isc.org/bind/]. [B.r.]. [online] Navštíveno 13.10.2023.

37. *Internet Systems Consortium* [https://www.isc.org/]. [B.r.]. [online] 28.12.2023.

38. *Knot DNS: High-performance authoritative DNS server* [https://www.knot-dns.cz/]. [B.r.]. [online] 28.12.2023.

39. *Knot Resolver: Resolve DNS names like it's 2024* [https://www.knot-resolver.cz/]. [B.r.]. [online] 8.1.2024.

40. *Knot DNS Gitlab* [https://gitlab.nic.cz/knot/knot-dns]. [B.r.]. [online] 28.12.2023.

41. *Knot Resolver Gitlab* [https://github.com/CZ-NIC/knot-resolver]. [B.r.]. [online] 8.1.2024.

42. MATTHEWS, Philip; SULLIVAN, Andrew; BEIJNUM, Iljitsch van; BAGNULO, Marcelo. *DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers* [RFC 6147]. RFC Editor, 2011. Request for Comments, č. 6147. Dostupné z DOI: 10.17487/RFC6147.

43. *PowerDNS Authoritative Nameserver* [https://doc.powerdns.com/md/authoritative/]. [B.r.]. [online] 8.1.2024.

44. *PowerDNS Recursor* [https://doc.powerdns.com/md/recursor/]. [B.r.]. [online] 8.1.2024.

45. *PowerDNS Github* [https://github.com/PowerDNS/pdns]. [B.r.]. [online] 8.1.2024.

46. *CacheServe: Product Brief* [https://www.akamai.com/resources/product-brief/cacheserve]. [B.r.]. [online] 8.1.2024.

47. *The ultimate on-premise protective DNS resolver* [https://www.whalebone.io/peacemaker]. [B.r.]. [online] 6.1.2024.

48. *DNS Secure, Network-based security solution providing threat protection and parental control functions for the consumer market.* [https://www.allot.com/network-security/dns-secure/]. [B.r.]. [online] 6.1.2024.

49. *Allot DNS Secure* [https://www.allot.com/pdf/dns_secure_ds_2023_pdf-lp/]. [B.r.]. [online] 6.1.2024.

50. *Build simple, secure, scalable systems with Go* [https://go.dev/]. [B.r.]. [online] Navštíveno 13.10.2023.

51. *Chapter 1. Setting up and configuring a BIND DNS server* [https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html/managing_networking_infrastructure_services/assembly_setting-up-and-configuring-a-bind-dns-server_networking-infrastructure-services]. [B.r.]. [online] Navštíveno 13.10.2023.

52. *sqlc: A SQL Compiler* [https://github.com/sqlc-dev/sqlc]. [B.r.]. [online] Navštíveno 15.10.2023.

53. *sqlc Documentation* [https://docs.sqlc.dev/en/latest/index.html]. [B.r.]. [online] Navštíveno 15.10.2023.

54. *Gin Web Framework* [https://gin-gonic.com/]. [B.r.]. [online] Navštíveno 16.10.2023.