VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ON-LINE MONITOROVÁNÍ EXPIRACE PODPISU DNSSEC

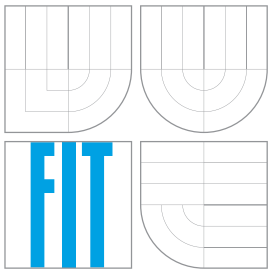
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAKUB MRÁZEK

BRNO 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ON-LINE MONITOROVÁNÍ EXPIRACE PODPISU DNSSEC

ON-LINE MONITORING OF EXPIRATION OF DNSSEC SIGNATURES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAKUB MRÁZEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PETR MATOUŠEK, Ph.D.

BRNO 2011

Abstrakt

V bakalářské práci je popsáno vytvoření programu, který kontroluje podpisy záznamů DNS (Domain Name System) a upozorňuje na jejich neplatnost nebo blížící se vypršení platnosti aby bylo možné včas zabránit vypršení platnosti. V práci je popsána technologie DNSSEC (Domain Name System Security), nové záznamy DNS pro DNSSEC a detaily podepisování záznamů DNS. Dále je zde popsán návrh programu, možnosti nastavení programu, způsoby čtení dat a způsoby výstupu upozornění. Práce vzniká z důvodu potřeby rychlé kontroly podpisů a upozornění na blížící se konec platnosti podpisů, protože žádný podobný program neexistuje.

Abstract

In the bachelor thesis we have described computer software (programme) which is supposed to check the signatures of DNS (Domain Name System) records (resource records) and warn in due time about the expiry date or the approaching expiry date of the signatures in order to prevent from expiring. In the paper we have specified the DNSSEC (Domain Name System Security) technology, new resource records for the DNSSEC and details of signing the resource records. Moreover, we have introduced the programme itself, the possibilities of setting the programme, the ways of data reading and warning output. Since no such programme exists, and because there is a need to check the signatures quickly and warn about their expiry date, we have decided for this work.

Klíčová slova

DNSSEC, expirace podpisu, elektronický podpis, zdrojový záznam

Keywords

DNSSEC, signature expiration, electronic signatures, resource record

Citace

Jakub Mrázek: On-line monitorování expirace podpisu DNSSEC, bakalářská práce, Brno, FIT VUT v Brně, 2011

On-line monitorování expirace podpisu DNSSEC

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Petra Matouška Ph.D. Další informace mi poskytli Ondřej Surý a Ing. Bedřich Košata Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jakub Mrázek
18. května 2011

Poděkování

Chtěl bych poděkovat sdružení CZ.NIC za zajímavé téma bakalářské práce, především konzultantům z CZ.NIC panu Ing. Bedřichu Košatovi Ph.D. za řešení problémů a detailů v práci a Ondřeji Surému za poskytnuté materiály. Dále Ing. Petru Matouškovi Ph.D. za zprostředkování práce a pomoc při řešení formálních náležitostí.

Obsah

Kapitola 1

Úvod

Služba DNS (*domain name system*) umožňuje překlad doménových jmen na číselnou adresu. Technologie DNSSEC (*domain name system security*) zabezpečuje službu DNS elektronickým podpisem, který je vytvořen pro každý typ záznamu DNS. Ověření záznamu DNS elektronickým podpisem potvrzuje, že záznam DNS je originální a tím zabraňuje podvrhnutí falešného záznamu DNS.

1.1 Motivace

Jedním z problémů, se kterými se správci DNS serverů setkávají při nasazování technologie DNSSEC, je nutnost pravidelné aktualizace záznamů. V případě zanedbání aktualizace, ať už z důvodu neinformovanosti nebo selhání personálu, dochází k expiraci podpisů a tím neplatnému ověření technologií DNSSEC. Aby bylo možné těmto problémům předcházet, je třeba vyvinout aplikaci (podobná aplikace neexistuje), která umožní expiraci podpisů monitorovat a upozorňovat na případné problémy. Aplikace by měla být použitelná nejen pro samotné správce DNS serverů (využití v CZ.NIC), ale také např. pro informované vlastníky domén, kteří nemají vlastní DNS servery, ale rádi by situaci aktivně monitorovali.

1.2 Cíl

V této práci chceme vytvořit program pro kontrolování vypršení platnosti DNS podpisů a DNS klíčů. Program bude jednorázově spouštěn a provede kontrolu pro vybrané DNS domény a typy záznamů. Pravidelné spouštění může provádět jiný program (např. `cron`). Program bude řízen konfiguračním souborem formátu INI, ve kterém bude uvedeno, co bude kontrolováno, jakým vstupem budou zadána data, s jakou délkou doby platnosti před vypršením a jakým výstupem program bude upozorňovat na blížící se dobu vypršení.

Kontrolované domény budou moci být zadány různými vstupy:

- Dotazy na autoritativní server, kde kontrolované domény budou získány přečtením textového souboru, ve kterém budou uvedeny jména a typy domén.
- Načtením lokálního zónového souboru ve standardním formátu.
- Načtením dat z serveru DNS pomocí přenosu zóny (*zone transfer*).

Výstup z programu bude řešen způsoby:

- Uložení do souboru.
- Odeslání zprávy e-mailem.
- Předání zprávy na vstup jinému programu.

1.3 Řešení

Program bude vypracován jako skript psaný v jazyce `Python`. Program bude používat knihovnu `python-ldns` pro práci s daty DNS. Při vstupu, který data získává dotazy na autoritativní server, program k přečtené doméně ze souboru vyhledá autoritativní server, ze kterého se dotáže na záznamy DNS podpisů (typ záznamu `RRSIG`). Získané záznamy po té kontroluje. Při načtení lokálního zónového souboru program čte postupně záznamy DNS ze souboru a, pokud přečte záznam DNS podpisu `RRSIG`, je zkontrolován. Při vstupu přenosem zóny se vytvoří spojení s serverem DNS a poté jsou postupně čteny záznamy DNS, v případě záznamu podpisu, je záznam zkontrolován. Kontrola záznamu DNS podpisu je prováděna porovnáváním data a času podpisu, data a času vypršení podpisu s aktuálním datumem a časem.

Pro zaznamenání výstupu program využívá standardní knihovnu `logging` jazyka `Python`, která umožňuje přidat k výstupní zprávě aktuální datum a čas, název a úroveň upozornění.

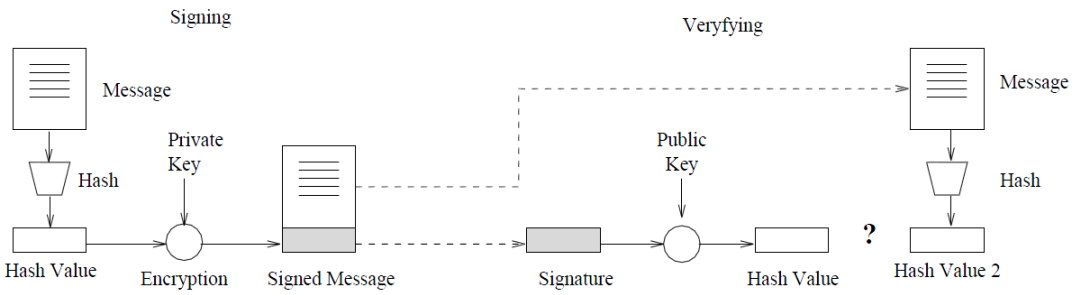
Pro čtení konfiguračních souborů program využívá standardní knihovnu `ConfigParser` jazyka `Python`, která umožňuje přečíst konfigurační soubor formátu INI.

Kapitola 2

DNSSEC

V této kapitole jsou podrobnosti o technologii DNSSEC, která zabezpečuje službu DNS elektronickým podpisem. Dále je vysvětleno, jak podepisování probíhá, a popsány nově vzniklé záznamy DNS pro podepsání a zabezpečení. A nakonec je objasněna výměna starých podpisů za nové, tzv. rotace klíčů.

Na obrázku 2.1 je znázorněn princip podepisování soukromým a veřejným klíčem. V technologii DNSSEC probíhá podepisování stejným způsobem jako na obrázku. Ze záznamu DNS (na obrázku *message*) se vypočítá *hash*, která je zašifrovaná soukromým klíčem a uložena do záznamu RRSIG (na obrázku *signature*). Veřejný klíč je uložen v záznamu DNSKEY. Uživatel při ověření vypočítá *hash* ze záznamu DNS (na obrázku *message*) a porovná ji s *hash* ze záznamu RRSIG, kterou rozšifroval pomocí veřejného klíče uloženém v záznamu DNSKEY.



Obrázek 2.1: Princip podepisování (obrázek přejat z [12])

2.1 Nové záznamy

Technologie DNSSEC přidává k obvyklým DNS záznamům (zdrojovým záznamům, *resource records.*) další typy záznamů pro ukládáni klíčů DNSKEY, pro ukládání podpisů RRSIG, pro zápornou odpověď NSEC nebo NSEC3 a pro uložení podpisu klíče DS.

2.1.1 DNSKEY

Záznam DNSKEY [8] slouží pro ukládání klíčů. Obsahuje položky `Flags`, `Protocol`, `Algorithm`, `Public key`. Položka `Flags` obsahuje údaj o tom, zda se klíč používá k ověření zóny a zda je to klíč podepisující zónu ZSK (*zone singing key*) nebo klíč podepisující jiný klíč KSK (*key singing key*). V poli `Protocol` je uvedena identifikace, pro který protokol je klíč určen (pro DNSSEC rovno 3). V položce `Algorithm` je zadána identifikace algoritmu šifrování. A v položce `Public key` je uložen veřejný klíč pro rozšifrování. V tabulce 2.1 je zobrazen příklad záznamu DNSKEY.

```
example.com. 86400 IN DNSKEY 256 3 5 (  AQPSKmynfzW4kyBv015MUG2DeIQ3
                                        Cbl+BBZH4b/0PY1kxkmvHjcZc8no
                                        kfzj31GajIQKY+5CptLr3buXA10h
                                        WqTkF7H6RfoRqXQeogmMHfpftf6z
                                        Mv1LyBUgia7za6ZEzOJBOztyvhjL
                                        742iU/TpPSEDhm2SNKLijfUppn1U
                                        aNvv4w== )
```

Tabulka 2.1: Ukázka záznamu DNSKEY pro*example.com*

2.1.2 RRSIG

Záznam RRSIG [8] je vytvořený při podepisování zóny DNSSEC, pro každý typ záznamu DNS v podepisovaném zónovém souboru. Záznam RRSIG obsahuje položky uvedené v tabulce 2.2.

Položka	Obsah
`Type Covered`	Typ záznamu, pro který tento záznam platí
`Algorithm`	Algoritmus šifrování
`Labels`	Počet částí v poli `NAME` (viz. syntaxe *domain name* [13])
`Original TTL`	Doba, po kterou si lze, záznam pamatovat
`Signature Expiration`	Datum a čas konce platnosti podpisu
`Signature Inception`	Datum a čas podepsání
`Key Tag`	Identifikace klíče, který lze použít pro ověření
`Signer's Name`	Jméno zóny
`Signature`	Podpis

Tabulka 2.2: Položky záznamu RRSIG

Pro projekt je nejdůležitější údaj `Signature Expiration`, který udává čas konce platnosti podpisu, a `Signature Inception`, který obsahuje datum podepsání. Proto si uvedeme jejich detaily. Může se vyskytovat ve dvou formátech, a to počet sekund od 1. 1. 1970 (stejně jako čas v operačním systému UNIX) nebo druhý formát času `YYYYMMDDHHmmSS`. Pro druhý formát platí pravidla popsané v tabulce 2.3.

Tyto dvě možnosti záznamu se rozpoznání velmi snadno, protože číslo nemůže být delší jak 10 znaků (maximální hodnota typu 32bit unsigned integer). Zatím co formát `YYYYMMDDHHmmSS` je 14 znaků dlouhý. V tabulce 2.4 je zobrazen příklad záznamu RRSIG.

znaky	význam	hodnota
YYYY	rok	(0001-9999)
MM	měsíc	(01-12)
DD	den	(01-31)
HH	hodina	(00-23)
mm	minuta	(00-59)
SS	sekunda	(00-59)

Tabulka 2.3: Význam znaků ve formátu YYYYMMDDHHmmSS

```
host.example.com. 86400 IN  RRSIG A 5 3 86400 20030322173103 (
                            oJB1W6WNGv+ldvQ3WDG0MQkg5IEhjRip8WTr
                            PYGv07h108dUKGMeDPKijVCHX3DDKdfb+v6o
                            B9wfuh3DTJXUAfI/M0zm0/zz8bW0Rznl8O3t
                            GNazPwQKkRN20XPXV6nwwfoXmJQbsLNrLfkG
                            J5D6fwFm8nN+6pBzeDQfsS3Ap3o= )
```

Tabulka 2.4: Příklad záznamu RRSIG pro *host.example.com*

2.1.3 NSEC,NSEC3

Kvůli odposlechnutí záporné odpovědi na dotaz (záznam neexistuje), která by byla vždy stejná, a po odposlechnutí by byla možnost ji podvrhnout. Byl přidán záznam NSEC [8], v němž byl stanoven následující záznam, u posledního záznamu je odkaz na první. Záporná odpověď byla realizována odpovědí předchozím záznamem. Tedy zabezpečuje zápornou odpověď.

Problém záznamů NSEC je, že pomocí jich jde snadno přečíst obsah celé zóny. Proto vznikl záznam NSEC3 [11], který neukládá další prvek, ale jen jeho *hash*. Zabezpečuje tím jak zápornou odpověď, tak snadné přečtení celé zóny. V tabulce 2.5 jsou zobrazeny příklady záznamu NSEC a záznamu NSEC3.

```
alfa.example.com. 86400 IN NSEC host.example.com. A MX RRSIG NSEC TYPE1234
b4um86eghhds6nea196smvmlo4ors995.example.com NSEC3 1 1 12 aabbccdd (
                                gjeqe526plbf1g8mklp59enfd789njgi MX RRSIG )
```

Tabulka 2.5: Příklad záznamu NSEC a záznamu NSEC3

2.1.4 DS

Záznam DS [8] je umístěn v rodičovské zóně a je v něm umístěn otisk klíče sloužící pro ověření pravosti klíče (KSK). V tabulce 2.6 je zobrazen příklad záznamu DS.

2.2 Řetězec důvěry

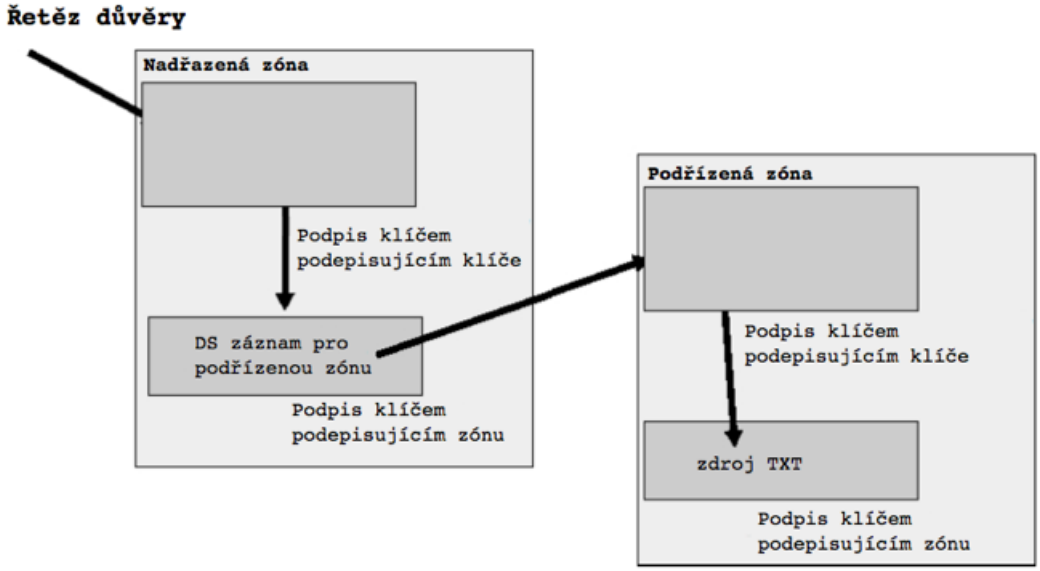
Na začátku kapitoly je popsáno, jak ověřit jednotlivý podpis. Ale pokud by nám byl podvrhnout klíč k ověřování podpisů, nedokázali by jsme rozpoznat, že získaná data nejsou

```
dskey.example.com. 86400 IN DS 60485 5 1 ( 2BB183AF5F22588179A53B0
                                           A98631FAD1A292118 )
```

Tabulka 2.6: Příklad záznamu DS

důvěryhodná. Proto jsou v zóně podepsány klíče pro podpis záznamů ((*zone singing key, ZSK*)), které jsou podepsány klíčem sloužící k podpisu klíčů (*key singing key, KSK*). A tento klíč (KSK) má podpis umístěný v nadřazené (rodičovské) zóně.

Řetězec důvěry [6] je vytvořen sestavením a ověřením posloupnosti klíčů a jejich podpisů až k autoritě, která je důvěryhodná. Důvěryhodný klíčem KSK ze zóny jedna získáme ověření klíče ZKS zóny jedna. Klíčem ZSK zóny jedna se získáme ověření platnosti záznamu DS v zóně jedna. Záznamem DS ze zóny jedna ověříme klíč KSK pro zónu dva. Klíčem KSK ověříme klíč ZSK v zóně dva a tím můžeme ověřit záznam v zóně dva. (Pokud by i zóna dva byla nadřazenou zónou ověření by dále pokračovalo.) Pro lepší názornost je vložen obrázek 2.2, který ukazuje závislosti.



Obrázek 2.2: Řetězec důvěry (obrázek přejat z [4])

Pokud není rodičovská zóna zabezpečena používá se technika *DNSSEC Lookaside Validation (DLV)* [5]. Tato technika umožňuje uložit důvěryhodný klíč (KSK) i jinam než do rodičovské zóny.

2.3 Podepsání zóny

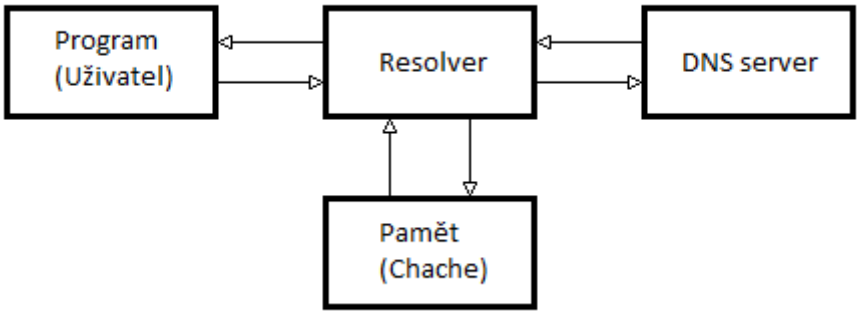
Každý doménový záznam je podepsán, ke skupině záznamů se stejným jménem, typem a třídou je vytvořen nový záznam RRSIG, odkazuje na veřejný klíč (ZSK) v záznamu DNSKEY. Před vlastním podepsáním jsou třeba nejprve vygenerovat dva klíče pro podpis klíčů (KSK) první příkaz v tabulce 2.7 a pro podpis zóny (záznamu) (ZSK) druhý příkaz v tabulce 2.7. Nakonec je zónový soubor podepsán třetím příkazem v tabulce 2.7.

```
dnssec-keygen -a NSEC3RSASHA1 -b 1024 -r /dev/urandom -f KSK example.com
dnssec-keygen -a NSEC3RSASHA1 -b 1024 -r /dev/urandom example.com
dnssec-signzone example.com
```

Tabulka 2.7: Příkazy pro podepsání

2.4 Dotazy na DNS server

Na obrázku 2.3 je znázorněn princip dotazování na DNS server. Uživatelský program položí otázku pomocí *resolver*, který odpověď nejdříve vyhledává v paměti a když ji nenalezne, pokládá dotaz na DNS server. Přijatou odpověď si uloží do paměti a předá uživatelskému programu. *Resolver* si může uchovávat záznam DNS po dobu TTL, která je určená v záznamu DNS.



Obrázek 2.3: Dotazování na DNS server

Pro dotazování na záznamy DNSSEC se může využívat spolehlivé spojení (TCP). V paketu se pro dotaz DNSSEC nastavují příznaky [7]. Ověřující *resolver* odesílá příznak *DO (DNSSEC OK)*, nastavující ověření pomocí DNSSEC. Klient může nastavit příznak *CD (Checking Disabled)*, který nastavuje, že chce vrátit i data, které nelze ověřit. A DNS server může nastavit *AD (Authenticated Data)*, že všechny záznamy v odpovědi byly ověřeny. Pokud nastavíte příznak *DO* a dotazem se zeptáme na záznam DNS z podepsané zóny, v odpovědi bude automaticky kromě požadovaného záznamu DNS, také záznam RRSIG, který obsahuje podpis. Pro získání klíče (záznamu DNSKEY) se musí provádět samostatný dotaz.

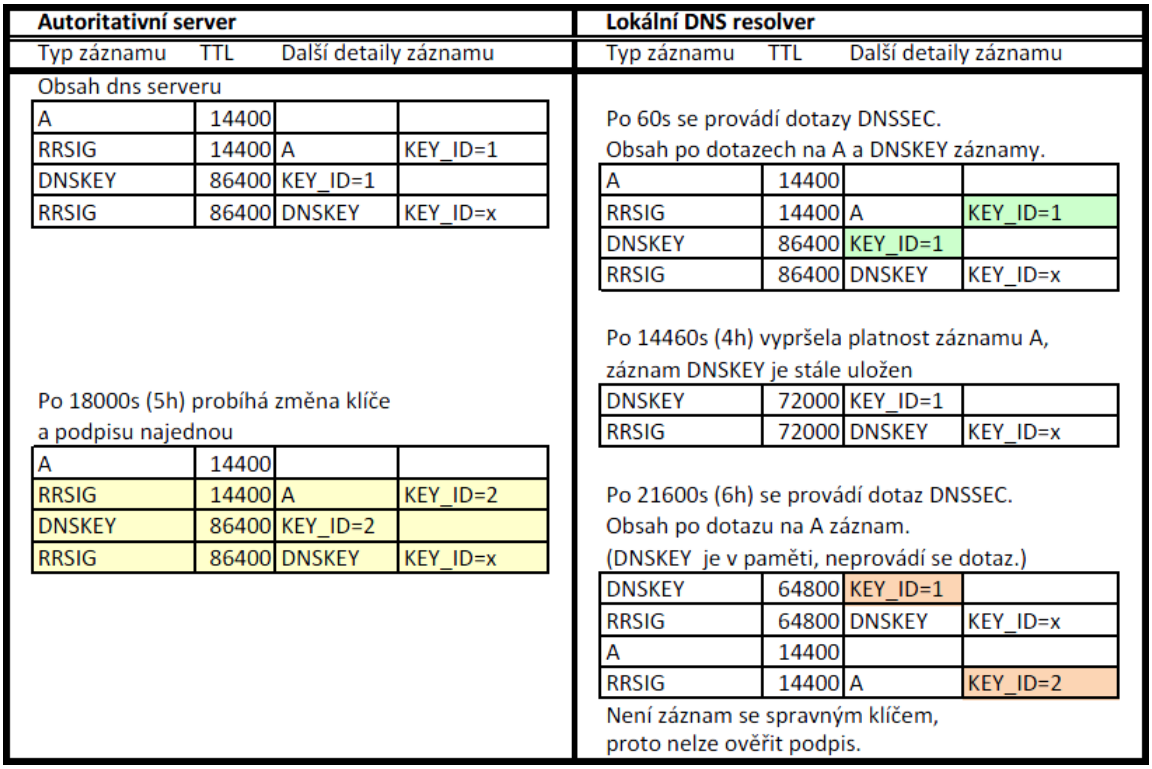
2.5 Rotace klíčů

Aby nemohlo dojít ke kompromitaci klíče, má podpis omezenou platnost a musí být pravidelně měněn. Celá rotace klíčů [10] musí být prováděna, tak aby nedošlo k výpadku ověření zóny.

Záznam DNSKEY a záznam RRSIG jsou samostatné záznamy, které se šíří samostatně. Platnost záznamů (TTL) nemusí být stejná, tedy každému záznamu vyprší platnost samostatně a samostatně se obnoví. Nelze tedy změnit záznam RRSIG a DNSKEY najednou,

protože v případě dříve obnoveného záznamu RRSIG, by odkaz na podepisující klíč byl neplatný a záznam by nešlo ověřit. Platí to i opačně (u dříve obnoveného klíče).

Pro názorný příklad je vložena obrázek 2.4, kde je zobrazena vlevo část obsahu DNS serveru a vpravo obsah paměti, kterou využívá *resolver*, směrem dolů plyne čas. Uživatelský program se připojuje a žádá *resolver* o adresu (záznam A) a pro ověření záznamu se ještě dotazuje na DNSKEY záznam. Tyto záznamy si *resolver* uloží do paměti. Záznam A má menší dobu platnosti (TTL) a po vypršení TTL je odebrán z paměti. V serveru DNS zatím dojde ke změně podpisu a klíče najednou. Uživatelský program se chce po čase znovu připojit a žádá záznam A a záznam DNSKEY. *Resolver* se dotáže na záznam A DNS serveru a DNSKEY vrací z paměti. Uživatelský program, ale tímto starým klíčem nemůže ověřit nový podpis.

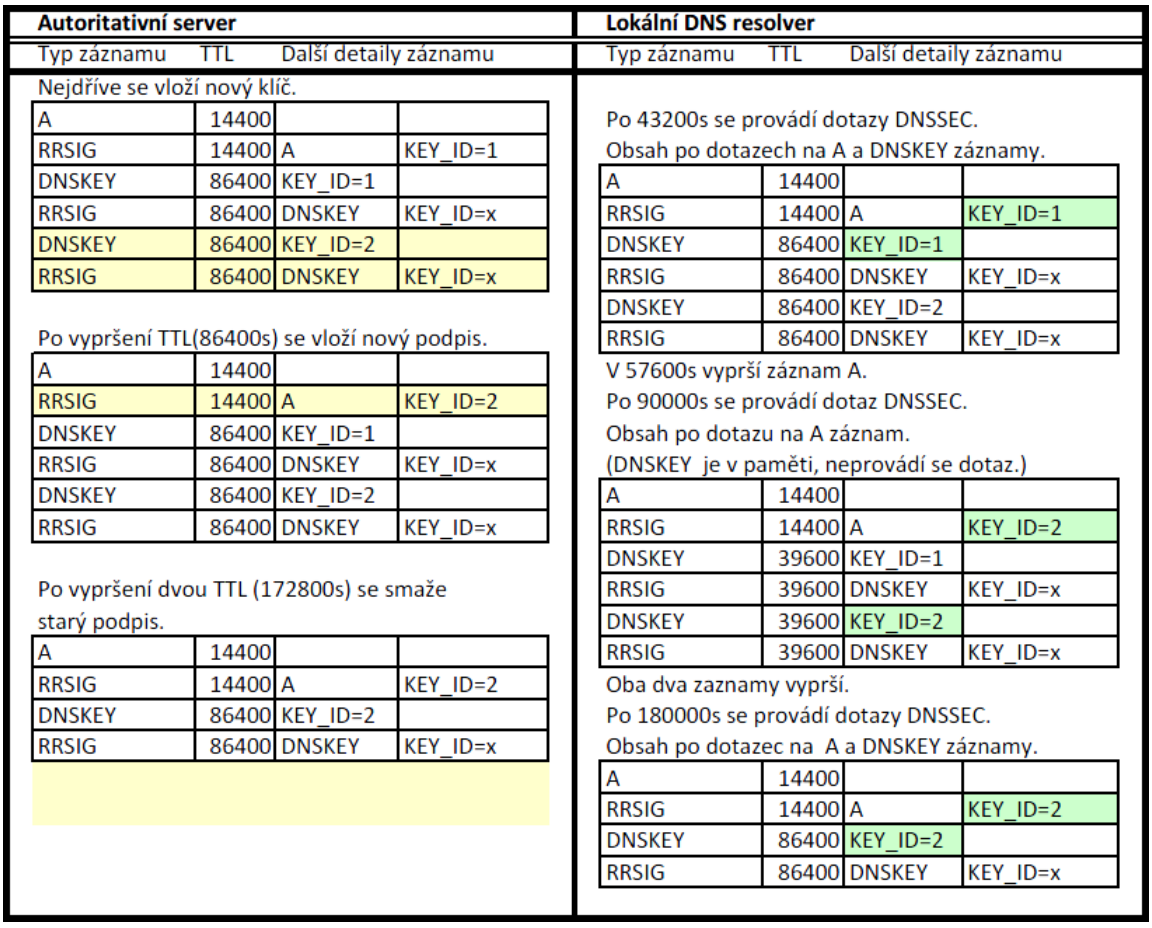


Obrázek 2.4: Chybná změna podpisu a klíče

Aby se této chybě zabránilo, používá se metoda, která do zóny nejdříve vloží nový klíč (záznam DNSKEY). Záznam DNSKEY je zveřejněn nejdéle o čas TTL dříve, než dojde k podepsání novým klíčem (změna RRSIG záznamů), klíč se tak dostane ke koncovým uživatelům dříve než nový záznam RRSIG. Po aktualizaci záznamu RRSIG je odkaz na nový klíč platný. Starý klíč můžeme smazat až po uplynutí dalšího času TTL od nového podepsání, kdy se nový záznam RRSIG dostal ke všem uživatelům.

Pro názorný příklad je vložen další obrázek 2.5. Hned na začátku je vložen nový klíč (záznam DNSKEY), v zóně existují dva klíče na jednou. Uživatelský program se pomocí *resolver* dotáže na záznam A a záznam DNSKEY, *resolver* si je opět ukládá do paměti. Uživatelský program si pro ověření použije klíč, který je identifikován v záznamu RRSIG. Po čase dojde opět k vypršení záznamu A a je vymazán z paměti. Na serveru, po uplynutí doby TTL od vložení nového klíče, dojde ke změně podpisu. Uživatelský program se znovu

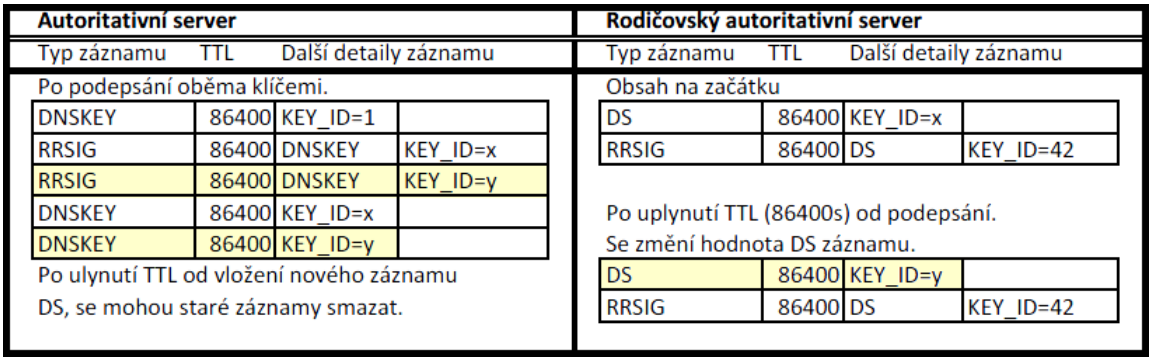
připojuje a žádá záznam A a záznam DNSKEY. *Resolver* vrací DNSKEY z paměti a dotazuje se DNS serveru na záznam A. Uživatelský program pro ověření nově podepsaného záznamu využije klíč určen v záznamu RRSIG (tentokrát ten druhý). Po uplynutí doby TTL na serveru od vložení nového podpisu se starý klíč může smazat. Těmito postupnými změnami lze zabránit chybnému ověření popisovaném výše.



Obrázek 2.5: Správná změna podpisu a klíče

U klíčů podepisující klíče (*KSK*) je problém stejný, ale řešení je jiné. Toto řešení využívá metodu dvojitého podpisu [15]. Vygeneruje se nový klíč a klíče *ZSK* se podepíšou oběma klíči jak starým a novým. Čeká se až se nově podepsané záznamy DNSKEY (*ZSK*) rozšíří podle doby TTL. Pak se vloží do nadřazeného serveru nový DS záznam. Po vypršení TTL záznamu DS můžeme starý klíč *KSK* smazat. Toto řešení je sice jednodušší, ale kvůli velkému množství záznamů v zóně se nedá použít pro klíče *ZSK*.

Příklad je zobrazen na obrázku 2.6. Vpravo je stále autoritativní server, ale vlevo je tentokrát nadřazený autoritativní server, ve kterém je záznam DS, který slouží k ověření klíče (*KSK*). Záznam DNSKEY (*ZSK*), který je vlevo, podepíšeme oběma klíči pro záznam DNSKEY (*ZSK*) budou dva záznamy RRSIG. Také vložíme pro oba záznamy RRSIG jejich klíče (záznamy DNSKEY, *KSK*). Po uplynutí doby potřebné k rozšíření podpisů (TTL). Se může změnit záznam DS v nadřazeném autoritativním DNS serveru. Po uplynutí času k rozšíření záznamu DS se mohou staré záznamy DNSKEY a RRSIG smazat.



Obrázek 2.6: Správná změna *KSK*

2.6 Shrnutí

V této kapitole byla vysvětlena technika DNSSEC, způsob jakým ověřuje platnost dat a co je nutné k vytvoření zabezpečené zóny, jaké nové záznamy DNS se používají a jak vyměnit staré klíče za nové. V další kapitole se budeme zabývat návrhem programu.

Kapitola 3

Návrh aplikace

V této kapitole se seznámíme s návrhem programu, se zápisem konfiguračního souboru, s formátem výstupu, s důležitými knihovnami pro program a se způsobem jakým jsou podpisy kontrolovány.

Návrh aplikace je zobrazen na vývojovém diagramu na obrázku 3.1. Program se spustí a vytvoří objekty, které při vzniku nastaví implicitní hodnoty. Po té jsou přečteny názvy částí v konfiguračním souboru, který byl předán argumentem při spuštění. Program přečte část `default` a upraví podle ní implicitní hodnoty. Po té začne program číst část za částí, když přečte všechny, program končí (na obrázku 3.1 podmíněný výraz vpravo nahoře). Pokud je v konfiguračním souboru část, tak ji celou přečte a hodnoty si uloží (když nejsou v části hodnoty, použijí se implicitní). Dále program nastaví výstupy programu podle uložených hodnot. Poté program přejde k výběru zvoleného vstupu (na obrázku 3.1 jsou to tři podmíněné výrazy ve druhém a třetím řádku). Možné vstupy jsou dotazy na autoritativní server, čtení zónového souboru a přenos zóny. Po vykonání části programu pro konkrétní vstup, které jsou rozepsány v následujících odstavcích, se program vrací přes výpis na výstupy ke čtení další sekce (na obrázku 3.1 podmíněný výraz vpravo nahoře).

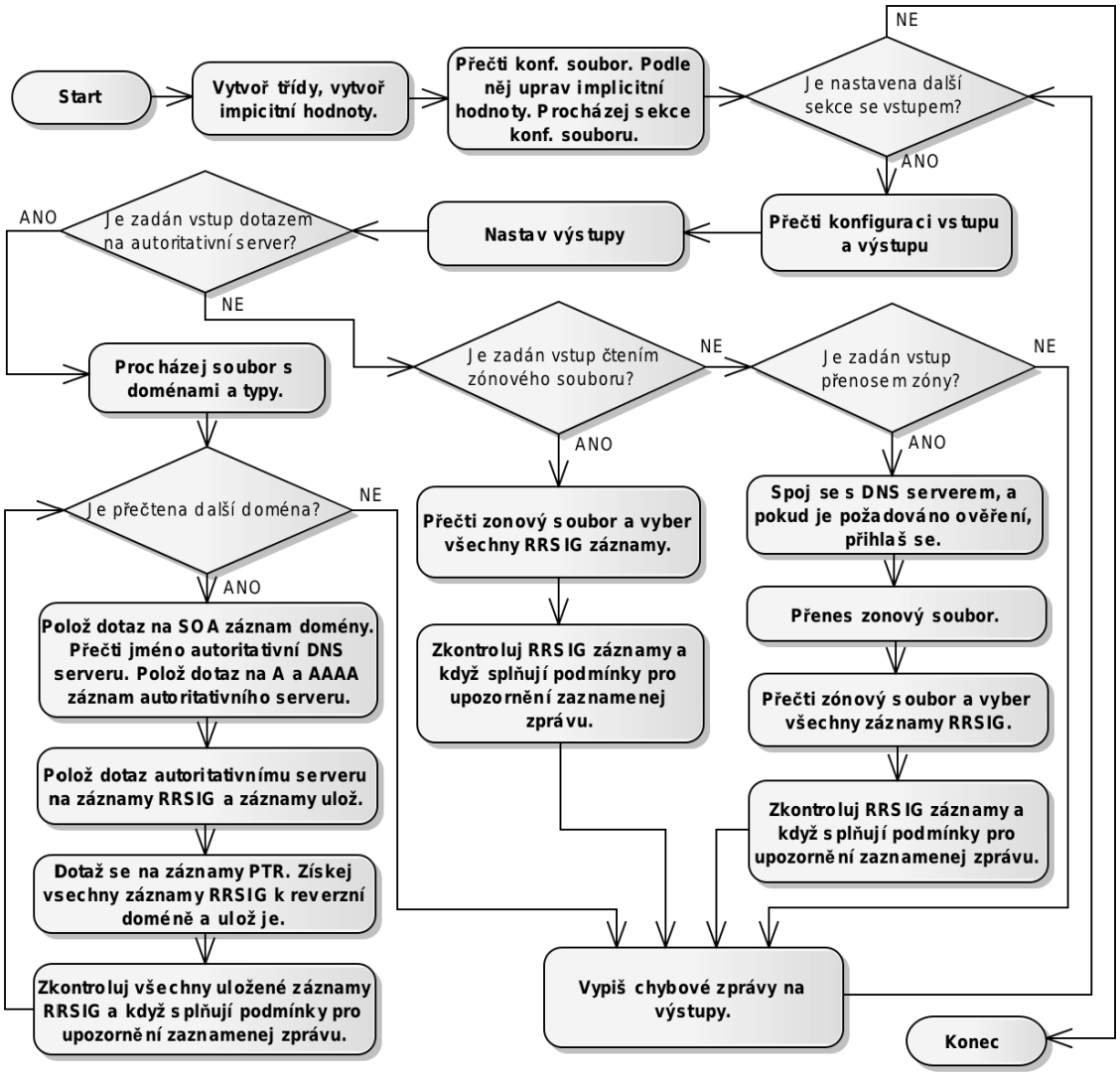
Pokud je zvolen vstup dotazy na autoritativní server (na obrázku 3.1 nastavení vstupu vyhoví podmíněnému výrazu na druhém řádku a program bude vykonávat sloupec pod podmíněným výrazem), prochází program soubor s doménami. Když je zadaná doména, program nejdříve zjistí jméno autoritativního DNS serveru (záznam SOA), pak toto jméno přeloží na IP adresu (získá záznamy A a AAAA). Po té položí dotaz autoritativnímu DNS serveru na všechny RRSIG záznamy. Záznamy z odpovědi si uloží. Když je nastavena kontrola PTR záznamů, program čte záznamy A a AAAA pro kontrolovanou zónu. Z těchto záznamů vytvoří reverzní záznamy a zjistí jméno autoritativního serveru, které opět musí přeložit na adresu a pak se teprve pokouší získat záznamy RRSIG reverzní domény. Když získá záznamy RRSIG, uloží je k ostatním. (Na obrázku 3.1 je kontrola reverzních záznamů zjednodušena, je znázorněna jedním polem pro příkaz, a to druhým od zdola v prvním sloupci.) Po kontrole záznamů PTR jsou zkontrolovány všechny záznamy a pokud splňují podmínky pro upozornění, je na tyto záznamy upozorněno. Program pokračuje se čtením další domény, a pokud další doména není, program se vrací zpět přes výpis na výstupy ke čtení částí konfiguračního souboru.

Při vstupu čtením zónového souboru (na obrázku 3.1 nastavení vstupu vyhoví podmíněnému výrazu na třetím řádku ve druhém sloupci, program bude vykonávat sloupec pod podmíněným výrazem) program čte zónový soubor po jednotlivých záznamech (omezení nároků na operační paměť), pokud narazí na záznam RRSIG, je záznam zkontrolován a upozorní se na něj. Po přečtení zónového souboru program vypíše hlášení na výstupy

a pokračuje čtení další sekce.

Když je zvolen vstup přenos zóny (na obrázku 3.1 nastavení vstupu vyhoví podmíněnému výrazu na třetím řádku ve třetím sloupci, program bude vykonávat sloupec pod podmíněným výrazem), program se spojí s DNS serverem a pokud je třeba ověří se v případě, že má nastaveny hodnoty pro ověření, a poté zahájí přenos. Opět je zóna čtena po záznamu a kontrolovány všechny záznamy RRSIG. Poté program vypíše hlášení na výstupy a pokračuje čtením další části.

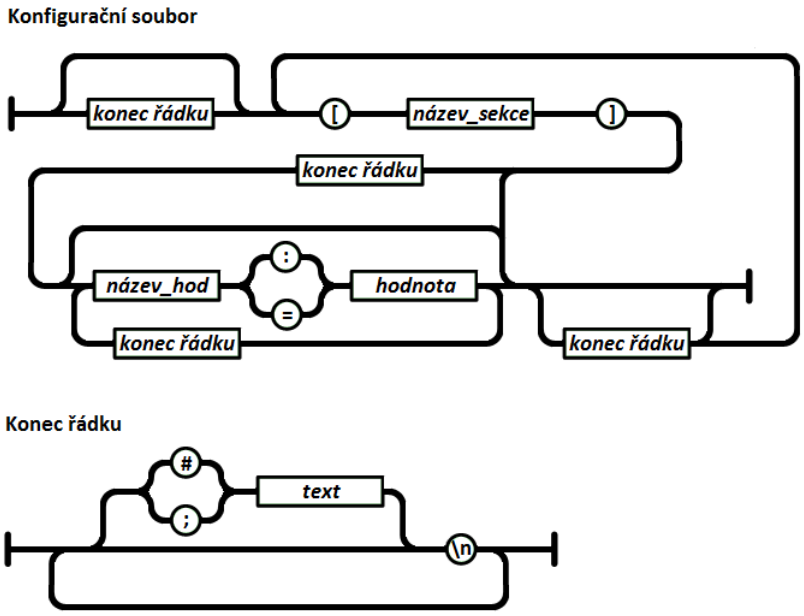
V programu je třeba dbát na zachycení chyb. Kdyby chyba nebyla zachycena program by havaroval a nebyla by dokončena celá kontrola. Mohlo by dojít k chybě ověření nějaké zóny, která nebyla zkontrolována. Chyby v průběhu zpracování sekce jsou zachyceny a zpráva o nich uložena. Pak následuje výpis hlášení a čtení další sekce. Výjimku tvoří část se vstupem dotazy na autoritativní server kde se pokračuje bez výpisu čtením další domény a až po přečtení všech domén dojde k výpisu. Pokud program obdrží signál přerušení (*SIGINT*, vyvolán například `CTRL+C`), program nejdříve vypíše hlášení a pak se teprve ukončí.



Obrázek 3.1: Vývojový diagram

3.1 Konfigurační soubor

Konfigurační soubor umožňuje měnit nastavení programu. Programu je předán jako jediný parametr při spuštění. Pokud je program puštěn bez parametru použije se konfigurační soubor se stejným názvem jako program, jen je jeho přípona změněna na `.ini` a je předpokládáno stejné umístění konf. souboru jako programu. Pokud konfigurační soubor nebude nalezen, či nepůjde otevřít program vypíše chybu na chybový výstup (`stderr`). Konfigurační soubor má formát, který je zobrazen na obrázku 3.2, a zahrnuje v sobě formát *INI*. Nejdříve se podíváme na spodní obrázek `konec řádku`, u této části může být ukončení řádku nebo komentář uvedený znakem `#` nebo `;`, část zahrnuje i následující prázdné řádky či řádky s komentářem. Přejdeme k hlavnímu obrázku. Na začátku mohou být prázdné řádky či řádky s komentářem (vše co je povoleno v části `konec řádku`), pak následuje název sekce. Po názvu sekce musí být `konec řádku` nebo konec souboru. Sekce může být prázdná anebo se v ní nacházejí data. K názvu dat je přiřazena hodnota a to znakem `=` nebo `:`. Pak můžou být další data konec souboru či další sekce.



Obrázek 3.2: Formát konfiguračního souboru

Pro čtení konfiguračního souboru je využita standardní knihovna `ConfigParser` jazyka `Python` [3], která umožňuje získat jednoduše hodnoty z konfiguračního souboru voláním metod třídy `ConfigParser`.

Konfigurační soubor může mít část `default`, jejíž hodnoty upravují implicitní hodnoty, které jsou nastaveny při vytváření objektů. Při procházení sekcí se používají hodnoty přečtené v dané sekci a pokud hodnoty v sekci nejsou, použijí se implicitní hodnoty. Implicitní hodnoty jsou nastaveny, tak aby se dané příkazy neprováděly. Pokud jsou nastaveny, tak aby se příkazy prováděly, je na tyto hodnoty upozorněno v komentáři.

3.1.1 Data v konfiguračním souboru

V části `default` se nacházejí jedna položka, která nemůže být zapsána v jiných sekcích. Je to položka `default_file_log`, která nastaví soubor, do něhož se budou ukládat výstupy ze všech sekcí v konfiguračním souboru. Tyto dvě položky a název sekce je zobrazen v tabulce 3.1.

```
[default]
default_file_log = DNSSEC_expiration.log
```

Tabulka 3.1: Položky pouze pro část `default`

Následující hodnoty mohou být v části `default` i v jiných částech. Nejdříve se zaměříme na časy upozorňování. Položkou `time_warning_TTL` je určená délka doby, vypočítaná součinem tohoto čísla a délky TTL, kdy bude program vypisovat varování o blížícím se konci podpisu (implicitně nastaven násobek dvou TTL). Protože program může být spouštěn pravidelně po nějaké době, ale doba TTL může být menší, může se stát, že program v době na upozornění nebude spuštěn, proto je možnost nastavit dobu pro upozornění pevným časem v sekundách položce `time_warning`. Dále je možno nastavit čas pro informační zprávu o tom, že se blíží doba vypršení, položky `time_info_TTL` a `time_info` určují dobu stejným způsobem jako čas pro varování. Dále je tu nastavení maximálního počtu zpráv položkou `number_logg`, po kterém jsou data vypsána na výstupy (implicitní hodnota je zvolena 100 000, při odesláním emailu je jeho velikost do 15,5 MB). Další položkou je nastavení IP adres DNS serverů pro *resolver*, v konfiguračním souboru jde o položku `dns_resolv_conf`, implicitně je nastaven soubor (`/etc/resolv.conf`), který se používá pro přístup k internetu. Tyto položky jsou zobrazeny v tabulce 3.2.

```
time_warning_TTL = 2.1          ; (default - 2)
time_warning = 172800
time_info_TTL = 3.0             ; doporučeno - 3, (default - 0)
time_info = 259200
number_logg = 100               ; (default - 100000)
dns_resolv_conf = resolv.conf   ; (default - /etc/resolv.conf)
```

Tabulka 3.2: Položky pro všechny části

Nastavení výstupů muže být také v části `default` i v jiných částech. První položka `file_out` je pro nastavení souboru, do kterého se výstup bude zapisovat. Další položka `stdout` je pro výpis na standardní výstup (zapne se hodnotami `true, yes, 1`). Další možností je položka `script_out`, do které se zapíše název programu a parametry programu. Zadaný program pak bude spuštěn a na jeho vstup bude předán výstup z tohoto programu. Poslední možností je odesláním emailu. Pro odeslání bychom měli zadat minimálně komu chcete email doručit a od koho a jaký je poštovní server (SMTP) položky `to_email`, `from_email` a `smtp_server`. Tyto a další položky pro nastavení odchozího emailu jsou zobrazeny v tabulce 3.3.

Vstupy z programu se dají nastavit jen v jiných částech, než v části `default`. Pro každou část musí být zvolen právě jeden vstup, jinak program vypisuje chybové hlášení. První možností vstupu je dotazování na autoritativní server. Pro tento vstup je třeba zadat

```
file_out = log.log
stdout = yes
script_out = grep -n
to_email = xmraze05@stud.fit.vutbr.cz
from_email = xmraze05@stud.fit.vutbr.cz
smtp_server = eva.fit.vutbr.cz
smtp_port = 465                 ; (ssl=False def-25 ,ssl=True def-465)
smtp_login = xmraze05
smtp_password = ***
smtp_tls = false                ; (default - False)
smtp_ssl = true                 ; (default - False)
smtp_keyfile = keyfile
smtp_certfile = certfile
smtp_certfile = 10
```

Tabulka 3.3: Položky výstupů pro všechny části

položku `file_csv`, které je přiřazen soubor formátu *CSV* obsahující domény a typy. Je zde ještě možnost položkou `control_PTR` zapnout i kontrolu reverzních domén. Další možnost je čtení zónového souboru, název zónového souboru je přiřazen položce `file_zone`. Obdoba tohoto vstupu je čtením zónového souboru ze standardního vstupu a to zvolením položky `stdin`. Poslední možnost je přenos zóny. Doména zóny je přiřazena položce `domain`. Pokud nechceme přenášet zónu z autoritativního serveru pro tuto doménu (zapsaném v SOA záznamu), zadáme jméno DNS serveru položce `domain_dns`. Pokud jsou potřebné další údaje pro ověření *TSIG* zadáme je do položek `tsig_algorithm`, `tsig_keydata` a `tsig_keyname`. Tyto záznamy jsou zobrazeny v tabulce 3.4.

```
[název_jiný_než_dafault]
file_csv = domény.csv                    ;(default - SOA záznam)
file_zone = zona.zs
stdin = true
domain = example.com
domain_dns = pirozek.com
tsig_algorithm = hmac-sha1
tsig_keydata = 8zjf20SkfuRW5rCgdjANRg==
tsig_keyname = example.com
```

Tabulka 3.4: Položky vstupů pro jiné části než `default`

3.2 Vstupy programu

V této kapitole je popsána funkce programu pro jednotlivé části programu, které čtou z různých vstupů. Popis pseudokódem ukazuje, jak části programu pracují. K získání přibližného kódu částí programu bychom mohli dosadit metody knihovny pro komunikaci DNS popsané v kapitole 3.4, za slovní popis metod.

3.2.1 Vstup dotazy na autoritativní server

Část programu pro položení dotazu na autoritativní server je zobrazena pseudokódem na obrázku 3.3. Zobrazena je část pro jednotlivý dotaz bez čtení souboru s doménami. Nejdříve program vytvoří dvě spojení s `resolver`. Pak zobrazenou metodou zjistí název autoritativního serveru, pak jeho adresu IP. Potom z autoritativního `resolver` odebere všechny záznamy a vloží do něj získané záznamy. Tím nastaví, že dotazy odchází na autoritativní server. Ještě zapne spolehlivé spojení a přenos dat DNSSEC. Potom metoda končí. Následuje dotaz na všechny záznamy RRSIG, které jsou vloženy do seznamu. Pokud kontrolovaná doména má záznamy A a AAAA, bude zkontrolována i reverzní doména. Data získávají stejně jako v minulém případě, jen výsledný seznam je přidán do seznamu původního. Pak je výsledný seznam procházen a záznamy jsou kontrolovány způsobem popsaným v kapitole 3.5.

```
1  def nastav_resolver_aut_na_aut_server_domény (jméno_domény){
2      záznam_soa  = resolver.dotaz (jméno_domény,SOA)
3      záznam_A    = resolver.dotaz (záznam_soa.jméno_serveru(),A)
4      záznam_AAAA = resolver.dotaz (záznam_soa.jméno_serveru(),AAAA)
5      while (resolver_aut.odeber_záznam()):  pass
6      resolver_aut.vlož_záznam (záznam_A)
7      resolver_aut.vlož_záznam (záznam_AAAA)
8      resolver_aut.nastav_přenos_dnssec()
9      resolver_aut.nastav_přenos_přes_TCP()
10 }
11 resolver = vytvoř_resolver()
12 resolver_aut = vytvoř_resolver()
13 nastav_resolver_aut_na_aut_server_domény (jméno_domény)
14 seznam_záznamů_RRSIG  = resolver_aut.dotaz (jméno_domény,RRSIG)
15
16 záznam_A_domény = resolver.dotaz (jméno_domény,A)
17 if (záznam_A_domény):
18     nastav_resolver_aut_na_aut_server_domény (
19                                     záznam_A_domény.reverzní_adresa())
20     seznam_záznamů_RRSIG_rev  = resolver_aut.dotaz (jméno_domény,RRSIG)
21     seznam_záznamů_RRSIG.vlož_seznam(seznam_záznamů_RRSIG_rev)
22
23 záznam_AAAA_domény = resolver.dotaz (jméno_domény,AAAA)
24 if (záznam_AAAA_domény):
25     nastav_resolver_aut_na_aut_server_domény (
26                                   záznam_AAAA_domény.reverzní_adresa())
27     seznam_záznamů_RRSIG_rev  = resolver_aut.dotaz (jméno_domény,RRSIG)
28     seznam_záznamů_RRSIG.vlož_seznam(seznam_záznamů_RRSIG_rev)
29
30 for záznam in seznam_záznamů_RRSIG
31     proveď_kontrolu_záznamu(záznam)
```

Obrázek 3.3: Pseudokód dotazu na autoritativní server

3.2.2 Vstup čtením zónového souboru

Část programu pro čtení zónového souboru je zobrazena pseudokódem na obrázku 3.4. Nejdříve program zjistí, zda bude číst data ze souboru nebo ze standardního vstupu. Pak si nastaví pomocné proměnné. Čte jeden záznam po druhém. Pokud čtení vrací status `ok` program si ukládá předchozí jméno, to se použije když je pro RRset (skupinu záznamů se stejným jménem) zapsáno jméno pouze u prvního záznamu. Pokud je záznam typu RRSIG, bude předán kontrole. Když čtení vrací hodnotu statusu jinou než `ok` a `konec_souboru`, program ukládá chybu a další čtení neprobíhá. Pokud je status `konec_souboru` další čtení neprobíhá. A v obou případech část programu pro čtení zónového souboru končí.

```
1  if (název_souboru):
2      místo_čtení = otevři_soubor (název_souboru)
3  else:
4      místo_čtení = standardní_vstup
5  minulé_jméno = prázdné
6  status = ok
7  while (status == ok):
8      (status, záznam) = přečti_záznam (místo_čtení, minulé_jméno)
9      if (status == ok):
10         minulé_jméno = záznam.jméno()
11         if záznam.typ()==typ_RRSIG:
12             proveď_kontrolu_záznamu(záznam)
13     elif (status != konec_souboru):
14         ulož_chybu_čtení_souboru()
```

Obrázek 3.4: Pseudokód čtení ze zónového souboru

3.2.3 Vstup přenosem zóny

Část programu pro přenos zónového souboru je zobrazena pseudokódem na obrázku 3.5. Program vytvoří spojení s *resolver*. Pokud nebylo zadáno jméno serveru DNS, ze kterého má být zóna přenesena, program zjistí jméno autoritativního serveru pro doménu, která bude přenesena. Pak program získá adresy IP serveru a nastaví *resolver* na adresy IP, které získal. Pokud jsou zadána data pro ověření, nastaví je do *resolver*. Dále naváže spojení s serverem DNS, pokud dojde k chybě uloží hlášení o chybě a část končí. Pokračuje přenášením záznamů DNS. Pokud je přenesen záznam typu RRSIG je předán kontrole. Po přenosu je zkontrolováno zda byla zkontrolována celá zóna.Pokud, je výsledek negativní program uloží chybu. Tímto ověřením část pro přenos zóny končí.

3.3 Výstup programu

V této části kapitoly je popsán formát výstupních zpráv a použití knihovny, která nastavuje formát výstupu a vypisuje do zvolených umístění.

3.3.1 Formát výstupu programu

Výstup z programu má dva formáty zprávy umožňující rychlé vizuální rozlišení. První je určen pro vypisování zpráv o upozornění na blížící se konec platnosti. Nabývá čtyři

```
resolver = vytvoř_resolver()
if (not jméno_serveru):
    záznam_soa = resolver.dotaz (jméno_domény,SOA)
    jméno_serveru = záznam_soa.jméno_serveru()
záznam_A    = resolver.dotaz (jméno_serveru,A)
záznam_AAAA = resolver.dotaz (jméno_serveru,AAAA)
while (resolver.odeber_záznam()):  pass
resolver.vlož_záznam (záznam_A)
resolver.vlož_záznam (záznam_AAAA)
if (data_pro ověření):
    resolver.nastav_algoritmus(algoritmus)
    resolver.nastav_klíč(klíč)
    resolver.nastav_jméno_klíče (jméno_klíče)
status = resolver.navaž_spojení (jméno_domény)
if (status != ok):
    ulož_chybu_přenosu_souboru()
    konec()
while True:
    záznam = resolver.přenes_záznam()
    if not záznam: break
    if záznam.typ()==typ_RRSIG:
        proveď_kontrolu_záznamu(záznam)
if (not resolver.je_přenos_kompletní()):
    ulož_chybu_přenosu_souboru()
```

Obrázek 3.5: Pseudokód přenosu zóny

úrovně důležitosti (`info`, `warning`, `error`, `critical`). Rozlišení těchto úrovní je popsáno v podkapitole 3.5. Druhý formát je určen pro chyby vzniklé při běhu programu. Ten má dvě úrovně důležitosti chybovou (`error`), pro případy jako je špatný záznam v souboru se zónou nebo neexistující DNS server pro přenos zóny apd., a varovnou (`warning`), například pro chybu ve jméně domény, která může nastat v souboru s doménami, jméno domény je rozděleno mezerou, program se pokusí pokračovat dále s první částí, ale je možné, že kontroluje jinou doménu, než zadávající chtěl.

Ukázky výstupního formátu a obsahu zpráv jsou vloženy tabulky 3.5 a 3.6. Záznamy těchto dvou tabulek na sebe navazují podle písmene uvedeného na konci nebo začátku tabulky. Do druhé tabulky se nevešel zápis data a času. Je nahrazen trojicí znaků `XXX` a význam uveden v posledním řádku tabulky, který je oddělen čarou.

datum	čas	název	úroveň	doména	typ záz.	zpr.
2011-01-21	13:22:55,109	DNSSEC	INFO	example.com	A	I
2011-01-21	13:22:55,109	DNSSEC	WARNING	example.org	AAAA	W
2011-01-21	13:22:55,110	DNSSEC	ERROR	example.cz	MX	E
2011-01-21	13:22:55,111	DNSSEC	CRITICAL	example.eu	CNAME	C

Tabulka 3.5: Formát zpráv

Na ukázku chybového formátu je vložena tabulka 3.7, textová zpráva je doplněna podle chyby, která nastala.

zpr.	text	koncový čas	TTL
I	validity expiring in 002 day(s) 18:32:42	end date: XXX	TTL: 86400
W	validity expiring in 001 day(s) 18:32:42	end date: XXX	TTL: 86400
E	validity expiring inless that TTL	end date: XXX	TTL: 86400
C	validity expired	end date: XXX	TTL: 86400
C	actutal time < start time : XXX		
C	start time > end time - start date: XXX	end date: XXX	
XXX	2011-05-17 12:02:42 (datum a čas)		

Tabulka 3.6: Obsah zpráv

název	úroveň	datum	čas	zpráva
DNSSEC expiration	Program WARNING	2011-01-21	13:22:55,109	text
DNSSEC expiration	Program ERROR	2011-01-21	13:22:55,110	text

Tabulka 3.7: Formát chybových zpráv

3.3.2 Použití knihovny pro výstup programu

Pro výstup bude použita standardní knihovna `logging` [3] jazyka `Python`. Knihovna umožňuje ukládat zprávy do různých výstupů, které obalí informacemi o datu a času generování (s přesností na tři desetinná místa sekundy), jménu, které se předává při vytvoření, a důležitostí zprávy, která se zadává při zápisu zprávy. Pro zápis je třeba nejdříve vytvořit objekt záznamníku (`logger`) metodou `logging.getLogger(název)`, kde název udává jméno, které je použito při výstupu.

Aby bylo možné ze záznamníku vypisovat, nastaví se objekt manipulátoru (`handler`). Pro jeden záznamník může být nastaveno více manipulátorů. Knihovna `logging` umožňuje použít manipulátor `FileHandler` zapisující do souboru , nebo manipulátor `StreamHandler` pro ukládání do otevřených souborů nebo na standardní či chybový výstup. Pro použití dalších manipulátorů je potřeba naimportovat část knihovny `logging.handlers`. Tím vznikne přístup k dalším manipulátorům, jako jsou `SMTPHandler`, umožňující odesílání upozorňujících zpráv e-mailem, `MemoryHandler` umožňující hromadění zpráv a po dosažení zadaného počtu ne přepsat na jiný manipulátor a jeho obecnější třídu `BufferingHandler`, která nedefinuje, jak budou data vypsána.

Pří implementaci nastal problém, které manipulátory použít. Zpočátku to vypadalo dobře, ale postupem času se začaly objevovat chyby v řešení. Manipulátor `FileHandler` při vypisování velkého množství dat postupně zvětšoval dobu zápisu jednoho hlášení. Při vypisování 880 000 hlášení trval běh programu přes deset hodin. Manipulátor `SMTPHandler` odesílá email pro každou zprávu zvlášť. Řešením se zdálo být použití manipulátoru `MemoryHandler`, ovšem pomohlo pouze se záznamníkem `FileHandler`. Z původních deseti hodin se běh programu zkrátil na deset minut. Protože předání záznamů z paměti dochází opět postupně po jednom záznamu, manipulátor `SMTPHandler` odesílá stále více e-mailů. Pro spuštění dalšího skriptu a předání dat na jeho vstup, také manipulátor `MemoryHandler` není vhodný, protože data je třeba předat jako text vcelku ne po jednotlivých záznamech. A vytvářet manipulátor `MemoryHandler` pro každý výstup samostatně zvyšuje nároky na operační paměť.

Řešení problému je vytvořit si vlastní manipulátor třídy `MyMemHandler`, který je podobný

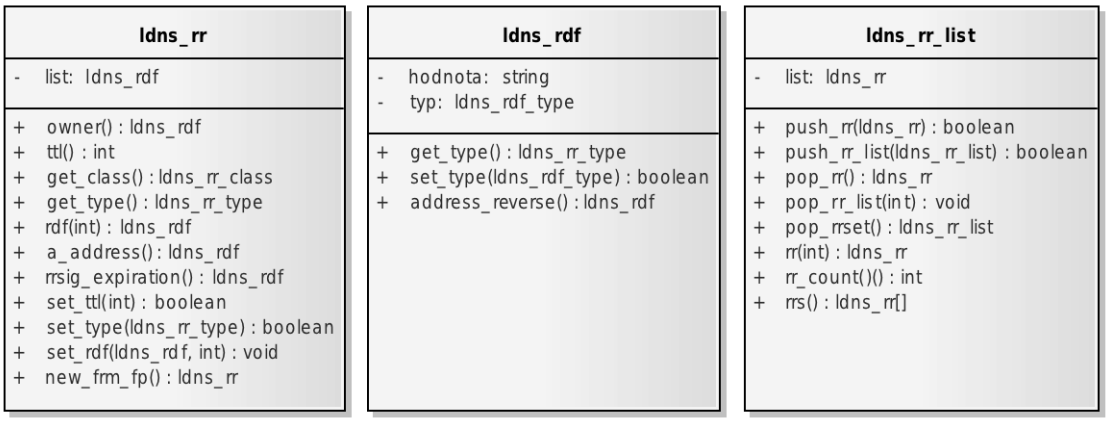
manipulátoru `MemoryHandler` a dědí stejně jako on ze třídy manipulátoru `BufferingHandler`. Manipulátor `MyMemHandler` neuchovává na rozdíl od `MemoryHandler` pouze jeden záznamník pro výstup, ale uchovává jich více v poli. Pro výstup do souboru je použit manipulátor `FileHandler` a manipulátor `SreamHandler` pro výstup na standardní výstup, protože zprávy potřebujeme, také odeslat emailem nebo předat na vstup jiného programu je v manipulátoru `MyMemHandler` při vypisování ze zpráv sestavena jedna textová zpráva, která je odeslána e-mailem pomocí knihovny `SMTP` anebo předána na vstup jiného programu pomocí knihovny `subprocess`. Třídou manipulátoru `MyMemHandler` je vyřešen problém s pomalým zápisem i s redundantním využíváním operační paměti. Všechny data pro výstupy jsou předány třídě manipulátoru při vytváření, pouze další manipulátory jdou do objektu přidávat.

V předchozí podkapitole 3.3.1 bylo řečeno, že výstup má dva formáty zpráv, pro každý tento formát je vytvořen objekt záznamníku, tím jsou od sebe upozorňující a chybové zprávy odděleny. Každému objektu záznamníku je vytvořen objekt manipulátoru třídy `MyMemHandler`. Další manipulátory jsou opět vytvářeny v páru a přidávány na oba manipulátory `MyMemHandler`. Při vytváření manipulátoru `MyMemHandler` je mu předán kromě dat na výstup, také odkaz na chybový záznamník, kdyby při vypisování záznamů došlo k chybě. Důležité je tedy nejdříve vyprázdnit (vypsat na výstupy) upozorňující záznamník a pak teprve chybový záznamník, kdyby došlo k chybě při vyprazdňování, bude chyba uložena a při vyprazdňování chybového záznamníku vypsána na ostatní výstupy. Chybový záznamník je automaticky také vypisován na chybový výstup programu.

3.4 Použití knihovny pro komunikaci DNS

Knihovna `ldns` napsaná v jazyce *C* umožňuje programování DNS. Knihovna potřebuje knihovnu `OpenSSL` na využití kryptografických funkcí. Knihovna podporuje IPv4 i IPv6, zabezpečení TSIG a DNSSEC. V programu bude využívána knihovna `python-ldns`, ta je nadstavbou nad knihovnu `ldns`, která umožňuje její použití v jazyce `python`, při čemž zachovává její rychlost a zjednodušuje její používání (nemusí se uvolňovat paměť).

Uvedeme si některé použité třídy a metody této knihovny. Další dokumentaci naleznete na internetových stránkách [14].



Obrázek 3.6: Třídy knihovny python-ldns

3.4.1 Třída `ldns_rr`

Třída `ldns_rr` je třída pro záznam DNS (resource records). Data v záznamu jsou uloženy v `ldns_rdf` pro každou hodnotu samostatně. Třída obsahuje metody pro získávání a nastavování různých hodnot ze záznamu. Pro přístup k prvním čtyřem hodnotám záznamu DNS, slouží první čtyři metody z obrázku 3.6. K dalším hodnotám v záznamu se lze dostat metodou `rdf(číslo)`, které se předá pozice hodnoty. Pro některé typy záznamů existují metody pro získání konkrétních hodnot (např. `rrsig_expiration()`, `a_address()`). Ve třídě jsou další metody, příklady některých na obrázku 3.6.

Pro čtení zónového souboru lze využít metoda `new_frm_fp()`. Tato metoda vrací přečtený záznam `ldns_rr`. V případě neúspěšného přečtení vyvolá výjimku, která nelze rozlišit pro chybný zápis v souboru nebo konec souboru. Při průzkumu zápisu této metody jsem zjistil, že tato metoda volá metodu `ldns_rr_new_frm_fp_l_()`, která navíc vrací hodnotu `status`, podle které jde rozeznat konec souboru a chybu v souboru.

3.4.2 Třída `ldns_rdf`

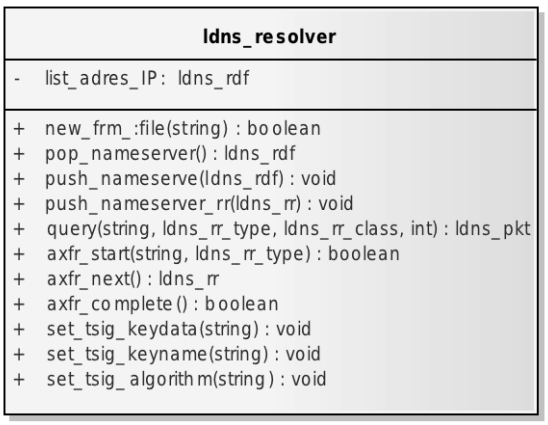
Třída slouží k ukládání jednotlivých hodnot záznamu DNS. Tyto záznamy mají své typy podle obsahu. Pokud je uložena adresa IP (IPv4 i IPv6), lze zavolat metoda `address_reverse()`, která vrátí reverzní doménové jméno k adrese. Některé další metody třídy jsou na obrázku 3.6.

3.4.3 Třída `ldns_rr_list`

Třída seskupuje záznamy DNS (`ldns_rr`) do seznamu. Důležitá metoda této třídy je `rrs()`, která vrací standardní seznam DNS záznamů (`ldns_rr`), který je možný procházet v cyklu (`for`). Příklady některých dalších metod třídy jsou v obrázku 3.6.

3.4.4 Třída `ldns_resolver`

Pro spojení a získání dat z DNS serveru se používá resolver. Resolver je klientský program, který umožňuje posílat dotazy na DNS servery, interpretovat odpovědi. V knihovně `ldns` je resolver představován třídou `ldns_resolver`. Třída je znázorněna na obrázku 3.7.



Obrázek 3.7: Třída ldns_resolver knihovny python-ldns

Resolver se vytváří metodou `new_frm_file("/etc/resolv.conf")`, nelze vytvořit jen objekt bez této metody. Ke vzniku je třeba alespoň jedena IP adresa DNS serveru. Pokud soubor nebude obsahovat aspoň jednu IP adresu DNS serveru, metoda vyvolá výjimku. Pro použití jiného DNS serveru než, který je uveden v souboru, musíme `ldns_resolver` vyprázdnit například cyklem `while` a metodou `pop_nameserver()`. Pro vložení nového DNS serveru použijeme metodu `push_nameserver()` nebo metodu podobnou.

Přenos DNSSEC obsahu zapneme metodou `set_dnssec(True)`, také musíme zapnout spolehlivý přenos (TCP) metodou `set_usevc(True)`.

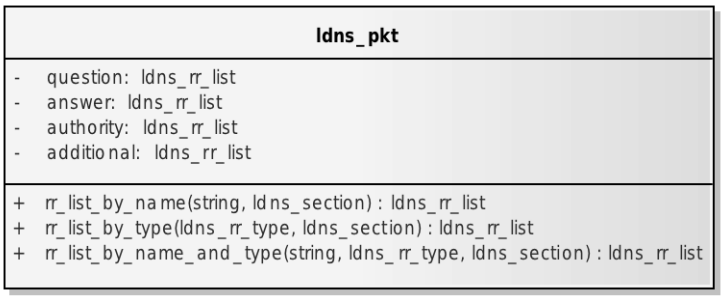
Nejdůležitější metoda resolveru je `query(doména, typ, třída, flags)`. Metoda pokládá nastavenému DNS serveru dotaz na doménu a typ. Metoda vrací třídu `ldns_pkt`.

Resolver také umožňuje přenos zóny (*zone transfer*). Nejdříve se spojí s DNS serverem metodou `axfr_start(jméno, třída)`. První parametr je jméno DNS serveru, ze kterého bude přenos uskutečněn. Dále se volá v cyklu metoda `axfr_next()`, která vrací záznam `ldns_rr`. Pokud metoda nevrátí hodnotu, znamená to konec přenosu. Je ještě potřeba zkontrolovat, zda byla zóna přenesena celá, k tomu slouží metoda `axfr_complete()`.

DNS servery mohou používat rozšíření DNS TSIG (*Transaction SIGnatures*, [16] a [9]), které zamezuje k neautorizovanému přístupu a získání zónového souboru. Aby bylo možné přenést zabezpečenou zónu systémem TSIG, musíme resolveru před začátkem přenosu nastavit klíč metodou `set_tsig_keydata(klíč)`, jméno klíče metodou `set_tsig_keyname(jméno_klíče)` a algoritmus šifrování `set_tsig_algorithm(algoritmus)`.

3.4.5 Třída `ldns_pkt`

Tato třída tvoří abstrakci nad daty přijatými v DNS paketu. Umožňuje číst a nastavovat všechny možné data a bity v paketu. DNS Paket je rozdělen do několika částí, ve kterých mohou být zapsány DNS záznamy. Třída je znázorněna na obrázku 3.8. Metody pro získávání záznamů DNS z částí jsou uvedené na obrázku.



Obrázek 3.8: Třída ldns_pkt knihovny python-ldns

3.5 Způsob kontroly podpisů

Kontrola záznamů RRSIG probíhá kontrolováním časů podepsání a konce platnosti podpisu. Kontrola je znázorněna na obrázku 3.9. Čas v záznamech může mít dva formáty (viz kapitola 2.1.2 RRSIG). Před kontrolou je potřeba oba časy převést na jednotný formát a to jako počet vteřin od 1. 1. 1970. K převodu využijeme knihovnu `time`. Pak se ještě vypočítá rozdíl mezi časem konce platnosti podepsání a aktuálním časem (vzorec 3.1). Rozdíl je čas do konce

platnosti. Tento čas (rozdíl) je používán pro zrychlení kontroly. Když hodnoty záznamu RRSIG nesplní podmínku kontroly, přejde se na kontrolu další podmínky. Pokud hodnoty záznamu RRSIG splní podmínku je uložena zpráva a končí kontrola tohoto záznamu. Zprávy jsou zobrazeny v tabulce 3.8.

Nejdříve je prováděná kontrola mezi datumem a časem podepsání a datumem a časem určující konec platnosti. V případě, že datum a čas začátku platnosti je po datumu a času konce platnosti, je uložena kritická zpráva *cr1*. Pak je prováděna kontrola datumu a času podepsání proti aktuálnímu datumu a času. V případě, že aktuální datum a čas je dříve než datum a čas podepsání, je uložena kritická zpráva *cr2*. Pokud je vypočtený rozdíl menší než nula, udává to, že platnost záznamu již vypršela a je zaznamenána kritická zpráva *cr3*.

Pokud je rozdíl menší než hodnota TTL, znamená to, že záznam může být v některé vyrovnávací paměti, kde dojde k jeho vypršení, a tím nemožnosti ověřit podpis. Takže ani okamžité nové podepsání zóny nedokáže zamezit všem chybám. Zaznamenává se tedy chybová zpráva *err*.

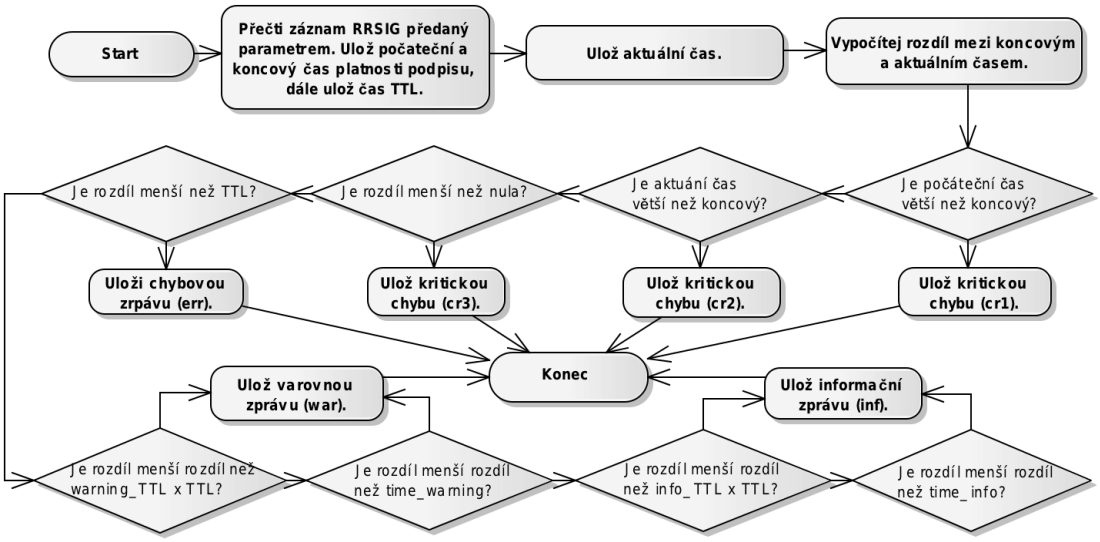
Pokud je rozdíl menší než dvakrát hodnota TTL, je čas nově podepsat zónu. Nové záznamy nahradí staré záznamy včas před vypršením platnosti. Je třeba mít v zóně vložen nový klíč (záznam DNSKEY). Dvakrát hodnota TTL je implicitní (z uvedených důvodů), lze ji změnit v konfiguračním souboru (položkou `time_warning_TTL`). Je uložena zpráva *war*. Když je rozdíl menší, jak pevný čas pro varování, je generována stejná varovná zpráva *war*. Pevný čas pro varování je možné nastavit v konfiguračním souboru (`time_warning`), implicitně hodnota není nastavena.

Další porovnání rozdílu s násobkem TTL pro informování (`time_info_TTL`). Implicitně hodnota není nastavena, ale je vhodná doba tři TTL před vypršením, protože je potřeba vložit nový DNSKEY záznam do zóny, aby po uplynutí dalšího TTL mohla být zóna nově podepsána. Toto ověření ukládá stejnou zprávu *inf*. Poslední porovnání je s pevným časem pro informování o času na potřebu změny podpisu. Tento čas je nastavitelný v konfiguračním souboru (položka `time_info`) a implicitně je vypnutý. Zpráva je stejná jako v minulém případě *inf*.

$$rozdil \quad = koncovy\_cas - aktualni\_cas \tag{3.1}$$

3.6 Shrnutí

V této kapitole byl probrán návrh celého programu, ale i podrobnější návrhy jeho jednotlivých částí vstupů a výstupů. Byla zde také uvedena konfigurace programu. Dále zde byly rozebrány použité knihovny jejich třídy a metody. V další kapitole se podíváme na spouštění a testování programu.



Obrázek 3.9: Kontrola záznamu RRSIG

cr1	název domény - typ podepisovaného záznamu - `start time > end time - start time:` datum a čas počátku `> end date:` datum a čas konce
cr2	název domény - typ podepisovaného záznamu - `actual time < start time:` datum a čas počátku
cr3	název domény - typ podepisovaného záznamu - `validity expired - end date:` datum a čas vypršení - `TTL -` hodnota TTL
err	název domény - typ podepisovaného záznamu - `validity are expiring per less TTL - end date:` datum a čas vypršení - `TTL -` hodnota TTL
war	název domény - typ podepisovaného záznamu - `validity are expiring in` počet dní `day(s)` a čas do konce platnosti - `end date:` datum a čas vypršení - `TTL` - hodnota TTL
inf	název domény - typ podepisovaného záznamu - `validity are expiring in` počet dní `day(s)` a čas do konce platnosti - `end date:` datum a čas vypršení - `TTL` - hodnota TTL

Tabulka 3.8: Zprávy generované kontrolou

Kapitola 4

Testování

V kapitole testování se seznámíme s tím co je třeba ke spuštění programu, jak byl program testován a také s výsledky testování.

Program byl testován v linuxovém prostředí *Arch Linux*. Skriptovací jazyk *Python* byl použit pro testování ve verzi *Python 2.7.1*. Knihovny *ldns* byla použita verze 1.6.9-1 se změněným souborem ldns.i. Pro testování funkce programu byl vytvořen zvláštní konfigurační soubor s několika sekcemi. Kromě testování správně nastavených hodnot, byl program také testován na všechny různé chybné nastavení.

4.1 Instalace knihovny ldns

Pro otestování celého programu je potřeba knihovna *python-ldns* ve verzi vyšší než 1.6.9-1 (to je aktuální verze k datu 10. května 2011). V této a starších verzích knihovny *python-ldns* je při používání metody `ldns.ldns_rr_new_frm_pf_l_()` chyba při dealokaci. Program se snaží dealokovat proměnné obsahující `None`. Program je po několika pokusech o dealokování ukončen. Tuto chybu vyřešil konzultant Ing. B. Košata Ph.D, který dodal upravený soubor `ldns.i`, který je potřeba nahrát do zdrojových souborů a pak je znovu přeložit. Zdrojové soubory lze stáhnout z internetových stránek [2]. Pro instalaci jsou potřebné programy `gmake` na přeložení a *swig* pro konfiguraci překladu a dále knihovna *openSSL*, proto aby knihovna *ldns* umožňovala kryptografické funkce. Překlad je odzkoušen v systému *Arch Linux* s prostředím *gnome*. V systému je třeba aby příkaz *python* otevíral *python* verze 2.x.x, pokud to tak není, musí se udělat změna. Nejdříve se spustí konfigurace příkazem `./configure --with-pyldns`. Další krok je překlad příkazem `make` (v prostředí jiném než *gnome* příkazem `gmake`). Posledním krokem je instalace příkazem `make install`, pro instalaci jsou nutná oprávnění správce.

4.2 Testování vstupu dotazy na autoritativní server

Tento vstup je zvolen v konfiguračním souboru položkou `file_csv`, které se přiřazuje cesta a soubor s domény a typy (ve formátu CSV), které budou kontrolovány. K tomuto vstupu je možno nastavit ještě položku `control_PTR`, hodnota může být `True` nebo `False` (výchozí nastavení). Její nastavení na `True` zapne kontrolu podpisů reverzních záznamů.

Příklad prvního výstupu programu platí pro dotaz na `nic.cz` s kontrolovaným typem `ANY` a povolenou kontrolou reverzního záznamu a s nastavením zasíláním informace 30 dní před vypršením platnosti záznamu (`time_info = 2592000`). Příklad druhého chybového

výstupu je pro doménu `seznam.cz`, která nemá podepsané záznamy. Příklad třetího chybového hlášení je pro neexistující doménu `nesmysl` a je stejný jako pro doménu, která nemá podepsané záznamy. Při dotazu na autoritativní server neexistující domény přijde odpověď s autoritativním serverem a při položení dotazu na RRSIG, také přijde prázdná odpověď. Při chybném zápisu domény a typu řádkem `nesmy sl, A AAA` do souboru CSV, bude program varovat před špatným zápisem a pro hledání použije první část, která není rozdělena bílými znaky. Čtvrtý výstup znázorňuje tuto chybu. Pokud nebude program moci přečíst soubor, vypisuje program páté hlášení.

Výstup 1: správný výstup

```
2011-05-01 09:56:01,045 - DNSSEC expiration - INFO - nic.cz. - SOA - validity are expiring
in 011 day(s) 04:07:00 - end date: 2011-05-11 13:03:02 - TTL: 1800
2011-05-01 09:56:01,045 - DNSSEC expiration - INFO - nic.cz. - NS - validity are expiring
in 011 day(s) 04:07:00 - end date: 2011-05-11 13:03:02 - TTL: 1800
2011-05-01 09:56:01,045 - DNSSEC expiration - INFO - nic.cz. - A - validity are expiring in
011 day(s) 04:07:00 - end date: 2011-05-11 13:03:02 - TTL: 1800
2011-05-01 09:56:01,045 - DNSSEC expiration - INFO - nic.cz. - MX - validity are expiring
in 011 day(s) 04:07:00 - end date: 2011-05-11 13:03:02 - TTL: 1800
2011-05-01 09:56:01,045 - DNSSEC expiration - INFO - nic.cz. - AAAA - validity are expiring
in 011 day(s) 04:07:00 - end date: 2011-05-11 13:03:02 - TTL: 1800
2011-05-01 09:56:01,046 - DNSSEC expiration - INFO - nic.cz. - NSEC - validity are expiring
in 011 day(s) 04:07:00 - end date: 2011-05-11 13:03:02 - TTL: 7200
2011-05-01 09:56:01,046 - DNSSEC expiration - INFO - nic.cz. - DNSKEY - validity are
expiring in 011 day(s) 04:07:00 - end date: 2011-05-11 13:03:02 - TTL: 3600
2011-05-01 09:56:01,046 - DNSSEC expiration - INFO - nic.cz. - DNSKEY - validity are
expiring in 011 day(s) 04:07:00 - end date: 2011-05-11 13:03:02 - TTL: 3600
2011-05-01 09:56:01,046 - DNSSEC expiration - INFO - 50.205.31.217.in-addr.arpa. - PTR -
validity are expiring in 012 day(s) 14:06:59 - end date: 2011-05-12 23:03:01 - TTL: 18000
2011-05-01 09:56:01,046 - DNSSEC expiration - INFO - 50.205.31.217.in-addr.arpa. - NSEC -
validity are expiring in 012 day(s) 14:06:59 - end date: 2011-05-12 23:03:01 - TTL: 7200
2011-05-01 09:56:01,046 - DNSSEC expiration - INFO -
2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.3.0.0.0.0.0.0.0.8.8.4.1.1.0.0.2.ip6.arpa. - PTR - validity
are expiring in 011 day(s) 05:22:53 - end date: 2011-05-11 14:18:55 - TTL: 18000
2011-05-01 09:56:01,047 - DNSSEC expiration - INFO -
2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.3.0.0.0.0.0.0.0.8.8.4.1.1.0.0.2.ip6.arpa. - NSEC - validity
are expiring in 011 day(s) 05:22:53 - end date: 2011-05-11 14:18:55 - TTL: 7200
```

Výstup 2: doména bez podpisu

```
DNSSEC expiration - Program ERROR - 2011-05-01 09:39:25,563 - Nejsou známe žádné RRSIG
záznamy pro doménu: "seznam.cz"
```

Výstup 3: chybná doména

```
DNSSEC expiration - Program ERROR - 2011-05-01 09:39:26,135 - Nejsou známe žádné RRSIG
záznamy pro doménu: "nesmysl"
```

Výstup 4: chybný záznam v souboru

```
DNSSEC expiration - Program WARNING - 2011-05-01 09:39:26,135 - Doména "nesmy sl"
v souboru "test's/test.csv"obsahuje bile znaky.
DNSSEC expiration - Program WARNING - 2011-05-01 09:39:26,135 - Typ domény "nesmy sl"
v souboru "test's/test.csv"obsahuje bile znaky.
DNSSEC expiration - Program ERROR - 2011-05-01 09:39:27,909 - Nejsou známe žádné RRSIG
záznamy pro doménu: "nesmy"
```

Výstup 5: neexistující zónový soubor

```
DNSSEC expiration - Program ERROR - 2011-05-01 09:39:29,723 - Soubor "test's/__error__"
nelze otevřít.
```

4.3 Testování vstupu čtení zónového souboru

Tento vstup je zvolen přiřazením položce `file_zone` v konfiguračním souboru cesty a jména zónového souboru. Nebo nastavením položky `stdin` na `true`, pak program čte zónový soubor ze standardního vstupu. Pro testování byl vygenerován zónový soubor programem `maketestzone` z nástrojů `DNSSEC-tools`. Informace a možnost stáhnutí je na stránkách [1]. K úspěšnému generování je třeba program `bind`. Pro vygenerování zóny `example.com` použijeme příkaz s přepínači `maketestzone -k -d com -P example --a-addr=127.10.0.0`, přepínač `-k` zapne generování klíčů a podepsání zóny, přepínačem `-d` zvolíme název požadované domény, přepínačem `-P` zvolíme prefix DNS serveru a přepínačem `--a-addr=127.10.0.0` zvolíme IPv4 adresu, která je přiřazena A záznamům. Tímto příkazem je vygenerováno více souborů, podepsaný zónový soubor je `db.example.com.zs.signed`.

Příklad výstupu s nastaveným time_info_TTL na hodnotu tři a upravenými šesti RRSIG záznamy, ve kterých jsou změněny hodnoty času a data podepsání a času a data konce platnosti podpisu, tak aby došlo ke všem možným chybám a upozorněním. Ostatní záznamy jsou ve vygenerovaném stavu, mají tedy platnost 30 dní, projdou tedy všemi kontrolami a nejsou ve výstupu zapsány. První výstup odpovídá takto nastavenému programu. Ve druhém příkladu je chybný soubor zóny a ve třetím soubor zóny neexistuje.

Výstup 1: správný výstup

```
2011-05-02 18:02:46,809 - DNSSEC expiration - CRITICAL - example.com. - SOA - start time >
end time - start: 2011-05-02 14:45:12 > end time: 2011-05-01 14:45:12
2011-05-02 18:02:46,809 - DNSSEC expiration - CRITICAL - example.com. - NS - actual time <
start time: 2011-05-03 14:45:12
2011-05-02 18:02:46,809 - DNSSEC expiration - CRITICAL - example.com. - TXT - validity
expired - end date: 2011-05-02 14:45:12 - TTL: 86400
2011-05-02 18:02:46,809 - DNSSEC expiration - ERROR - example.com. - NSEC - validity are
expiring per less TTL - end date: 2011-05-03 14:45:12 - TTL: 86400
2011-05-02 18:02:46,810 - DNSSEC expiration - WARNING - example.com. - DNSKEY - validity
are expiring in 002 day(s) 21:42:25 - end date: 2011-05-04 14:45:12 - TTL: 86400
2011-05-02 18:02:46,810 - DNSSEC expiration - INFO - example.com. - DNSKEY - validity are
expiring in 003 day(s) 21:42:25 - end date: 2011-05-05 14:45:12 - TTL: 86400
```

Výstup 2: chybný zónový soubor

```
DNSSEC expiration - Program ERROR - 2011-05-01 09:56:08,214 - Chyba v zónovém souboru
domeny.csv
```

Výstup 3: neexistující zónový soubor

```
DNSSEC expiration - Program ERROR - 2011-05-01 09:56:08,215 - Soubor "__error__" nelze
otevřít.
```

4.4 Testování vstupu přenos zóny

Pro testování přenosu zóny jsem si vygeneroval dva zónové soubory pro domény `example.com` a `example.org` s adresami IP z lokální smyčky 127.10.0.0 a 127.20.0.0 . Postup generování zónových souboru je popsán u předchozího vstupu v kapitole 4.3. Protože při generování je generována jedna IP adresa ručně jsem přepsal v obou souborech konec adresy IP. Poté jsem ručně vytvořil reverzní zónové soubory.

Pro nastavení zabezpečení přenosu zóny TSIG (*Transaction SIGnatures*, [16] a [9]), jsem musel vygenerovat klíč `dnssec-keygen -a HMAC-SHA1 -b 128 -n HOST -r /dev/urandom example.com`. Tento klíč bude nastaven k zónovým souborům, jak je popsáno dále, a také nastaven do konfiguračního souboru.

Na počítači (dostupným pod doménou `pirozek.com`) se systémem *ubuntu 10.10* jsem

spustil program *bind9*, který vytváří server DNS. Před spuštěním jsem do souboru `named.conf.local` vložil všechny čtyři zónové soubory. K doméně `example.com` a její reverzní doméně `0.10.127.in-addr.arpa.` jsem přidal záznam `allow-transfer { key "example.com";};` znamenající, že je potřeba ověření klíčem `example.com` pro přenos zóny. Musel jsem také upravit soubor `db.127` lokální smyčky, aby platil pro adresy 127.0.0, (původně platil pro 127.) a také nastavení domény lokání smyčky v souboru `named.conf.default-zones`. Nakonec jsem musel přidat klíč do souboru named.conf.option, klíč se přidá záznamem `key "example.com"{ algorithm hmac-sha1; secret "8zjf20SkfuRW5rCgdjANRg==";};`.

Na prvním výstupu je vidět příklad část zóny, která byla přenesena v pořádku, ale vypršela jí platnost. Na druhém výstupu je část přenesené zóny, u které bylo vyžádáno ověření. V druhém výstupu s ověřením a prvním výstupu bez ověření není rozdíl. Třetí až šestý výstup jsou případy kdy nastala chyba.

Výstup 1: správný výstup bez ověření

```
2011-05-07 09:21:42,781 - DNSSEC expiration - CRITICAL - example.org. - SOA - validity
expired - end date: 2011-04-16 13:58:37 - TTL: 600
2011-05-07 09:21:42,781 - DNSSEC expiration - CRITICAL - example.org. - NS - validity
expired - end date: 2011-04-16 13:58:37 - TTL: 86400
2011-05-07 09:21:42,781 - DNSSEC expiration - CRITICAL - example.org. - TXT - validity
expired - end date: 2011-04-16 13:58:37 - TTL: 86400
2011-05-07 09:21:42,781 - DNSSEC expiration - CRITICAL - example.org. - NSEC - validity
expired - end date: 2011-04-16 13:58:37 - TTL: 600
2011-05-07 09:21:42,782 - DNSSEC expiration - CRITICAL - example.org. - DNSKEY - validity
expired - end date: 2011-04-16 13:58:37 - TTL: 86400
2011-05-07 09:21:42,782 - DNSSEC expiration - CRITICAL - example.org. - DNSKEY - validity
expired - end date: 2011-04-16 13:58:37 - TTL: 86400
```

Výstup 2: správný výstup s ověřením

```
2011-05-07 09:21:43,484 - DNSSEC expiration - CRITICAL - example.com. - SOA - validity
expired - end date: 2011-04-16 13:58:37 - TTL: 600
2011-05-07 09:21:43,485 - DNSSEC expiration - CRITICAL - example.com. - NS - validity
expired - end date: 2011-04-16 13:58:37 - TTL: 86400
2011-05-07 09:21:43,485 - DNSSEC expiration - CRITICAL - example.com. - TXT - validity
expired - end date: 2011-04-16 13:58:37 - TTL: 86400
2011-05-07 09:21:43,485 - DNSSEC expiration - CRITICAL - example.com. - NSEC - validity
expired - end date: 2011-04-16 13:58:37 - TTL: 600
2011-05-07 09:21:43,485 - DNSSEC expiration - CRITICAL - example.com. - DNSKEY - validity
expired - end date: 2011-04-16 13:58:37 - TTL: 86400
2011-05-07 09:21:43,485 - DNSSEC expiration - CRITICAL - example.com. - DNSKEY - validity
expired - end date: 2011-04-16 13:58:37 - TTL: 86400
```

Výstup 3: neexistující doména

```
DNSSEC expiration - Program ERROR - 2011-05-07 09:21:44,189 - Přenos zóny domény:
"example.cz" se nezdařil nebo není kompletní.
```

Výstup 4: chybný algoritmus ověření

```
DNSSEC expiration - Program ERROR - 2011-05-07 09:21:44,559 - Nebylo navázáno spojeni AXFR
pro doménu: "example.com"
```

Výstup 5: chybné heslo

```
DNSSEC expiration - Program ERROR - 2011-05-07 09:21:45,095 - Nebylo navázáno spojeni AXFR
pro doménu: "example.com"
```

Výstup 6: chybný název hesla

```
DNSSEC expiration - Program ERROR - 2011-05-07 09:21:45,644 - Přenos zóny domény:
"example.com" se nezdařil nebo není kompletní.
```

4.5 Testování velkého množství dat

Byl také proveden zátěžový test, který procházel soubor zóny *cz.*, který má tři a půl miliónu záznamů, z toho 880 tisíc záznamů tvoří záznamy RRSIG. Na počítači s dvoujádrovým procesorem (*Intel Core 2 Duo*) s frekvencí 2,27 GHz trvá kontrola takto velké zóny pět minut s výstupem do souboru. Chybové zprávy byly do souboru zapisovány po nahromadění 100000 zpráv a všechny záznamy RRSIG měly vypršenou platnost. Náročnost na paměť RAM se pohybovala okolo 400 MB. Při odesílání zpráv na email se doba prodloužila na hodinu a deset minut, protože přes internetové připojení (*Wifi*) bylo odesláno 136 MB. Pro 100000 záznamů RRSIG, které jsou pro doménové záznamy (délka názvu je různá), je vytvořen email o velikosti přibližně 14 MB, a pro 100000 záznamů RRSIG, které podepisují záznamy NSEC3, je vytvořen email velikosti 15,7 MB.

V reálném použití by neměl nastat případ, že by se všechny záznamy zóny objevily ve výpisu najednou, nejdříve se objeví záznamy s nejdelší TTL, a pak postupně přibývají další. Při nastavení systémového času tak, aby při kontrole souboru zóny *cz.* nevznikla žádná zpráva úrovně `ERROR`, bylo vypsáno přibližně 2000 zpráv úrovně `WARNING` a 4000 zpráv úrovně `INFO`.

4.6 Shrnutí

V této kapitole jsme se seznámili s výstupy programu pro určitá vstupní data a s nároky programu. Dále jsme se seznámili s kroky, které jsou potřeba udělat před samotným testováním.

Kapitola 5

Závěr

V bakalářské práci se mi podařilo vytvořit program pro kontrolu platnosti podpisů DNSSEC. Pro vytvoření jsem nastudoval technologii DNSSEC, jejíž funkce je v této práci objasněna. DNSSEC zabezpečuje službu DNS elektronickým podpisem. Je zde tedy vysvětlen princip a způsob podepsání i jsou popsány nové záznamy, které byly vytvořeny pro DNSSEC. Protože podpisy DNSSEC mají omezenou platnost, popisuji průběh rotace klíčů. V návrhu aplikace je vysvětleno, jak program funguje i jak kontroluje záznamy, jak pracují jeho vybrané části, které knihovny jsou použity a jaké jsou jejich třídy a metody, které se používají. Je zde také uveden formát konfiguračního souboru a formát výstupních zpráva. Nakonec jsou zobrazeny a popsány výsledky programu, které byly získány při testování programu.

Vypracovaný program má další uplatnění v rámci sdružení CZ.NIC, které již tento program pravidelně spouští pro kontrolu zóny *cz.*. Program bude také dostupný na internetu pro správce domén, ale také pro vlastníky domén, kteří si budou moci provést kontrolu údržby domén.

Program má možnost dalšího rozšíření, tak aby vznikla možnost zapnout při dotazování na DNS servery ověření platnosti získaných dat, aby bylo průkazné, že program nepracuje s podvrhnutými daty. Zvýšení rychlosti kontroly by bylo možné dosáhnou při rozdělení programu do více vláken. Například oddělit kontrolování od výpisu dat nebo ještě oddělit odesílání emailů, které brzdí běh programu, jak vyplývá z předchozí kapitoly 4.5.

Literatura

[1] DNSSEC - tools. `http://www.dnssec-tools.org/`.

[2] ldns. `http://nlnetlabs.nl/projects/ldns/`.

[3] The Python Standard Library. `http://docs.python.org/library/`.

[4] Zabezpečení dat. `http://www.nic.cz/page/580/dnssec-howto/`.

[5] Andrews, M.; Weiler, S.: The DNSSEC Lookaside Validation (DLV) DNS Resource Record, RFC 4431. February 2006.

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

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

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

[9] Eastlake, D.: HMAC SHA TSIG Algorithm Identifiers, RFC 4635. August 2006.

[10] Kolkman, O.; Gieben, R.: DNSSEC Operational Practices, RFC 4641. September 2006.

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

[12] Matoušek, P.: *Opora do předmětu ISA*. FIT VUT, 2011.

[13] Mockapetris, R. V.: Domain Names - Implementation and Specification, RFC 1035. November 1987.

[14] Slaný, K.; Vašíček, Z.: PyLDNS documentation. `http://www.fit.vutbr.cz/ slany/nic-vip/pyldns/`.

[15] StJohns, M.: Automated Updates of DNS Security (DNSSEC) Trust Anchors, RFC 5011. September 2007.

[16] Vixie, M.; Gudmundsson, O.; Eastlake, D.; aj.: Secret Key Transaction Authentication for DNS (TSIG), RFC 2845. May 2000.

Příloha A

Obsah CD

Datový nosič CD obsahuje:

- Text bakalářské práce ve formátu PDF.
- Text bakalářské práce ve formátu TEX.
- Obrázky použité při tvorbě textu.
- Konfigurační soubor
- Konfigurační soubor a další soubory pro testování
- Vygenerovaný zónový soubor
- Vytvořené a změněné soubory umístěné na DNS serveru
- Vygenerovaný klíč pro přenos zóny (TSIG)
- Soubor `ldns.i` pro instalaci knihovny ldns

Příloha B

Konfigrační soubor

```
;; predem vytvoreny konfiguracni soubor:

;; kazda sekce urcuje nastaveni pro jeden vstup
;; sekce 'default' urcuje nastaveni, ktere se pouzije
;;     pokud v neni v sekci vstupu urceno jinak

;; smazanim ';' odkomentujete

;[default]                       ; implicitni hodnoty

;; casy pro zasilani varovani a informace
;time_warning = 172800          ; ve vterinach
;time_warning_TTL = 2           ; pouzije se nasobek TTL (default - 2)
;time_info = 259200             ; ve vterinach
;time_info_TTL = 3              ; pouzije se nasobek TTL doporuceno - 3 (default - 0)

;number_logg = 100              ; po kolika zpravach se provadi vystup (default - 100000)

;dns_resolv_conf = resolv.conf              ; soubor s DNS serverem (default - /etc/resolv.conf)

;default_file_log = DNSSEC_expiration.log  ; logovanim vseho do souboru

;; vystupy:

;file_out = log.log                         ; logovani do souboru

;stdout = yes                               ; vypis na standartni vystup

;script_out = grep -n "TXT -"              ; spusteni skriptu s predanim vystupu na vstup

;to_email = xmraze05@stud.fit.vutbr.cz     ; nastavení odchozích emailů
;from_email = xmraze05@stud.fit.vutbr.cz
;smtp_server = eva.fit.vutbr.cz
;smtp_port = 465                           ; (ssl=False def-25 ,ssl=True def-465)
;smtp_login = xmraze05
```

```
;smtp_password = ***
;smtp_tls = true                          ; (default - False)
;smtp_ssl = true                          ; (default - False)
;smtp_keyfile =
;smtp_certfile =
;smtp_timeout =

;; sekce:

;[nazev0] ; sekce vstupu

;; casy pro zasilani varovani a informace
;time_warning = 172800          ; ve vterinach
;time_warning_TTL = 2           ; pouzije se nasobek TTL (default - 2)
;time_info = 259200             ; ve vterinach
;time_info_TTL = 3              ; pouzije se nasobek TTL doporuceno - 3 (default - 0)

;number_logg = 100              ; po kolika zpravach se provadi vystup (default - 100000)

;dns_resolv_conf = resolv.conf             ; soubor s DNS serverem (default - /etc/resolv.conf)

;; vstupy:                     (v kazde sekci muze byt jeden vstup)

;file_csv = domeny.csv          ; dotazy na autoritativni server
;control_PTR = True             ; kontrola inverzni domeny (default - False)

;file_zone = zona.zs            ; cteni zonoveho souboru

;stdin = true                   ; cteni zonoveho souboru ze standartniho vstupu
;domain = ahoj                  ; název přenášené zóny
;domain_dns = pirozek.com       ; (default - SOA zaznam)
;tsig_algorithm = hmac-sha1
;tsig_keydata =   8zjf20SkfuRW5rCgdjANRg==
;tsig_keyname =   ahoj

;; vystupy:

;file_out = log.log                        ; logovani do souboru

;stdout = yes                              ; vypis na standartni vystup

;script_out = grep -n "TXT -"              ; spusteni skriptu s predanim vystupu na vstup

;to_email = xmraze05@stud.fit.vutbr.cz     ; nastavení odchozích emailů
;from_email = xmraze05@stud.fit.vutbr.cz
;smtp_server = eva.fit.vutbr.cz
;smtp_port = 465                           ; (ssl=False def-25 ,ssl=True def-465)
;smtp_login = xmraze05
```

```
;smtp_password = ***
;smtp_tls = true                          ; (default - False)
;smtp_ssl = true                          ; (default - False)
;smtp_keyfile =
;smtp_certfile =
;smtp_timeout =

;; dalsi sekce maji nastavitelne stejne hodnoty

;[nazev1]
;[nazev2]
```