DNSSEC

I. ZABEZPEČENÍ DAT DNS

Tato část pojednává o zabezpečení dat v zónových souborech. Popisujeme, jak vygenerovat a spravovat klíče, nastavit rekurzivní jmenný server, aby ověřoval data podepsané zóny a jak podepsat a poskytovat zóny.
Popíšeme zde následující kroky k zabezpečení technologií DNSSEC:

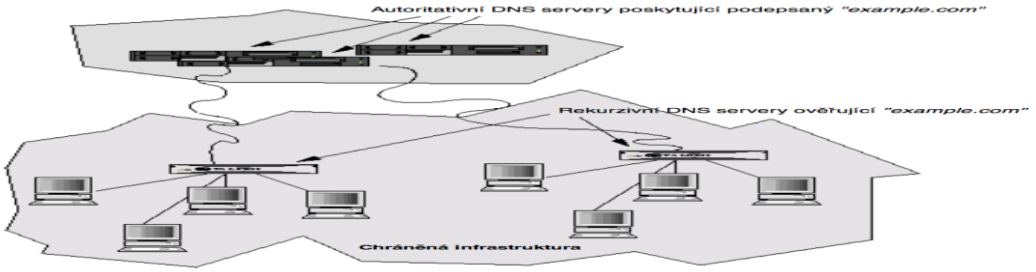
* Vytvoření tzv. „zabezpečeného ostrova" (Kapitola 1. [„Konfigurace rekurzivního jmenného serveru pro ověřování odpovědí"](/page/578/dnssec-howto/) a Kapitola 2., [„Zabezpečení zóny DNS"](/page/584/dnssec-howto/)) nakonfigurováním rekurzivního jmenného serveru pro ověřování podepsaných zón poskytovaných autoritativními jmennými servery vaší organizace. Pokud jste to zjistili a řešení implementovali, můžete si být jisti, že jsou data DNS ve vaší organizaci chráněna před pozměněním. Pokud jste vytvořili „zabezpečený ostrov", zbývá už jen malý krůček k tomu stát se součástí řetězu důvěry.
* Delegování podepisující autority; sestavení řetězu důvěry (Kapitola 3., [„Delegování podepisující autority; přechod na globální zabezpečení"](/page/592/dnssec-howto/)). Dozvíte se, jak si vyměňovat klíče se svou nadřazenou zónou a vnořenými zónami.
* Kapitola 4., [„Rotace klíčů"](/page/596/dnssec-howto/) pojednává o údržbě klíčů a zajištění stálého přístupu klientů k datům vašeho DNS během procesu rotace.

I. ZABEZPEČENÍ DAT DNS

1. KONFIGURACE REKURZIVNÍHO JMENNÉHO SERVERU PRO OVĚŘOVÁNÍ ODPOVĚDÍ

1.1 ÚVOD

Plánujeme nakonfigurovat rekurzivní jmenný server tak, aby ověřoval data, která přijímá. Uživatelé, kteří využívají tento rekurzivní jmenný server jako svůj resolver, pak budou přijímat pouze data, která jsou buď bezpečná a ověřená nebo zcela nezabezpečená. Výsledkem bude, že se bezpečná data, která nebudou ověřena, k uživatelům nedostanou. Pokud máte ověřovací rekurzivní jmenný server, chrání všechny, kteří jej používají jako forwarder, proti přijetí podvodných dat DNS.
Obrázek 1 ukazuje, jak nakonfigurovat rekurzivní servery DNS pomocí důvěryhodného klíče pro „example.com" tak, že všechna data poskytnutá autoritativními servery pro "example.com" jsou ověřena před předáním chráněné infrastruktuře, ve které jsou rekurzivní servery konfigurovány jako forwardery (jmenné servery, které jsou obvykle přiděleny přes DHCP nebo konfigurovány v /etc/resolv.conf).



Obrázek 1: DNS prostředí

Konfigurací veřejného klíče pro konkrétní zónu informujeme caching forwarder o tom, že všechna data vycházející z této zóny by měla být podepsána odpovídajícím soukromým klíčem. Zóna vystupuje jako důvěryhodný vstupní bod pro vstup do stromu DNS a klíč konfigurovaný v rekurzivním jmenném serveru je počátkem pro řetěz důvěry. V ideálním případě máte jako důvěryhodný přístupový bod nakonfigurovaný pouze jeden klíč: klíč root (kořenové) zóny.
Předpokládáme, že jste váš jmenný server nakonfigurovali pouze jako rekurzivní.
Předpokládáme rovněž, že jmenný server ve vaší organizaci byl nakonfigurován jako autoritativní server pro zabezpečenou zónu pojmenovanou example.net. Poznámky k nastavení zabezpečené zóny naleznete níže v Kapitole 2, „Zabezpečení zóny DNS".

1. KONFIGURACE REKURZIVNÍHO JMENNÉHO SERVERU PRO OVĚŘOVÁNÍ ODPOVĚDÍ

1.2 UPOZORNĚNÍ

Váš rekurzivní jmenný server bude považovat zónu, pro kterou jste nakonfigurovali trust-anchors (pevné body důvěry) jako zabezpečenou. Pokud zóny, pro které jste konfigurovali trust-anchors, změní své klíče, budete muset znovu nakonfigurovat také trust-anchors. Pokud tak neučiníte, dojde k tomu, že data v těchto zónách nebo v jakékoliv vnořené zóně budou označena jako podvodná a stanou se pro uživatele neviditelná.

1. KONFIGURACE REKURZIVNÍHO JMENNÉHO SERVERU PRO OVĚŘOVÁNÍ ODPOVĚDÍ

1.3 KONFIGURACE CACHING FORWARDERU

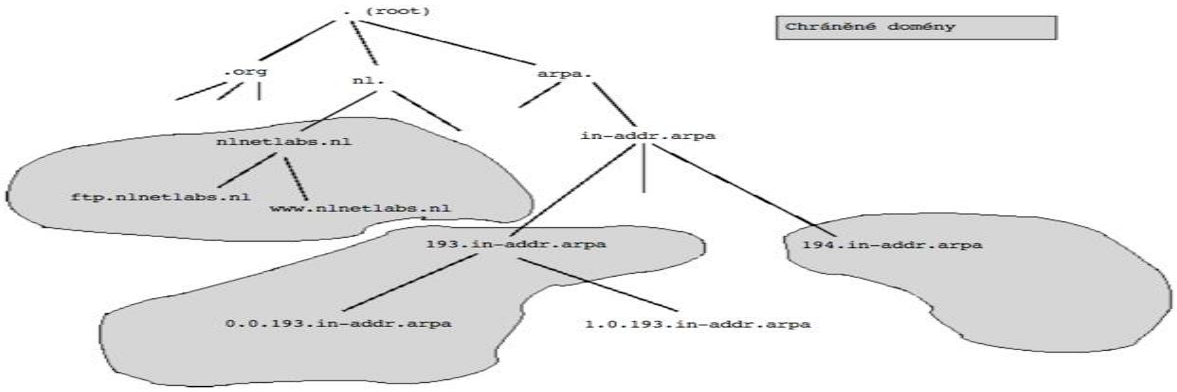
Informace o kompilaci BIND pomocí správných přepínačů, které povolují DNSSEC, naleznete v Příloze A. Nezapomeňte zadat příkaz

```
dnnsec-enable yes;
```

v možnostech vašeho named.conf.

1.3.1 KONFIGURACE TRUST ANCHOR

Trust-anchor, neboli pevný bod důvěry je veřejným klíčem, který je nakonfigurovaný jako přístupový bod pro řetěz pravomocí. V ideálním případě – kdy je root podepsaný a mohou být vytvářeny řetězy důvěry od vrcholových domén ke koncovým uzlům – bude k ověření jmenného serveru zapotřebí nakonfigurovat jeden z těchto trust-anchors. V prvních fázích nasazení budete pravděpodobně chtít konfigurovat vícenásobné trust-anchors.



Obrázek 2: Trust-anchors v DNS stromu

Na obrázku 2 představujeme strom zóny. Předpokládáme, že domény ripe.net, 194.in-addr.arpa, 193.in-addr.arpa a 0.0.193.in-addr.arpa v tomto stromu jsou podepsány. Předpokládáme rovněž, že mezi 193.in-addr.arpa a 0.0.193.in-addr.arpa existuje zabezpečené delegování. Aby bylo možné všechny tyto domény ověřit, musí ověřovací klient DNS nakonfigurovat trust-anchors pro ripe.net, 194.in-addr.arpa a 193.in-addr.arpa..

Pro konfiguraci trust-anchor musíte získat veřejný klíč zóny, kterou chcete použít jako počátek řetězu pravomocí, podle kterého se bude při ověřování dat řídit. Je možné jej získat přímo z DNS, ale jsou dva důvody, proč to nedoporučujeme.
Za prvé musíte ověřit autenticitu klíče, který hodláte konfigurovat jako svůj trust anchor. Jak to uděláte, závisí na tom, jakou metodu nerovnovážného ověření klíče povolil vlastník zóny.

- Můžete to provést prostřednictvím návštěvy zabezpečené webové stránky vlastníka zóny, kde ověříte informace klíče. Například RIPE NCC podepisuje několik reverzních zón. Svůj veřejný klíč publikují prostřednictvím projektu DISI.
- Pokud vlastníka zóny osobně znáte, můžete mu zatelefonovat.
- Můžete za důvěryhodný považovat klíč uvedený na účtu, který jste právě obdrželi od vlastníka zóny.
- Můžete se pouze domnívat, že váš dodavatel OS provedl ověření vaším jménem.

Za druhé můžete mít na výběr z několika veřejných klíčů. V takovém případě potřebujete zvolit správný klíč „důvěryhodného přístupového bodu". V DNSSEC je rozdíl mezi klíčem podepisujícím klíče a klíčem podpisujícím zóny. Klíč podepisující klíče podepisuje pouze DNSKEY RRset v apexu, zatímco klíč podepisující zóny podepisuje všechny RRsety v zóně. Klíče podepisující klíče jsou často užívány jako klíče důvěryhodných přístupových bodů (SEP). Tyto klíče SEP jsou klíče, které použijeme jako první při sestavování řetězu pravomocí od trust-anchor k podepsaným datům.
Doporučujeme jednoznačné mapování mezi klíči SEP a klíči podepisující klíče. V praxi mají klíče podepisující klíče nižší frekvenci rotace než klíče podepisující zóny, a tak byste měli nakonfigurovat SEP tj., klíče podepisující klíče.
Mimo to, že musíte mít správný veřejný klíč, měli byste se seznámit se zásadami rotace klíčů vlastníka zóny, nebo se ujistit, že máte nástroj, který obstará automatickou rotaci. Selhání při úpravě trust anchor předtím, než je obnoven odpovídající klíč SEP, bude mít za následek selhání při ověření.
Předpokládejte, že jste získali klíče podpisující klíče nlnetlabs.nl., 193.in-addr.arpa., a 195.in-addr.arpa.. Ke konfiguraci těchto klíčů jako trust-anchors budete muset přiřadit tyto klíče pomocí příkazů pro důvěryhodné klíče v named.conf rekurzivního jmenného serveru. Viz obrázek 3.

```
// Trusted keys
// These are examples only, do not use in production

trusted-keys {
"nlnetlabs.nl." 3 5
"AQPzzTWMz8qSWIQlfRnPckx2BiVmkVN6LPupO3mbz7Fh
LSnm26n6iG9NLby97Ji453aWZY3M5/xJBSOS2vWtco2t
8C0+xeO1bc/d6ZTy32DHchpW6rDH1vp86Ll+ha0tmwyy
9QP7y2bVw5zSbFCrefk8qCUBgfHm9bHzMG1UBYtEIQ==";

"193.in-addr.arpa." 257 3 5
"AwEAAc2RnCT1gjU22FbNC1baMQec77fq60z2HlCKscYl
3idBZTp703ApMfAAFcMZQGkSmo8NP+47KqZJwG9ISLaT
bUais3khgFVrf7lIRzPJAMlXHsmOMmpq5xBORF66EDt/
u2dau3qqzOfb/BrKCklGgnwBosgqaSPmWBQTuzJFqzi3
4FQIt4xFHWYyt3B5qZ9h4dpUL96etvvx1N+z8tlXjhlm
Vauw1EPZnz2rmY6HEJFS2zjaI1FrDtY5/pooJjRWjobk
RXL3iqjd5J/cmDikxjQCjwnbwBS+YvcwZCos4n9Xh2l5
kf2kOcq9xCmZvEplfWJ9lWbVkfhpWaM8qXXPN8E=";

"195.in-addr.arpa." 257 3 5
"AwEAAaMN4kOrGaiHJBikvcf+mhPxzprL85Q40VA0hbRc
a8FDDn6Xlkuj95Nizy2vMrOy1MjIjo7a+GACGp6C/Rdj
6nDimsRrUBr/G/dq+zBgg8qvRXWJZhx+zNCgkfv9gs1B
eRlPnjXr1K/x5viTzQRDK3SYfHiCMVNxuYN+T7kniDLx
QRUI/ASF3YxqNQ+Oo+T5L6nYtO7uLeAUdxzToRdIHaey
iSnq52boA/3Yg6X8Kbo1uAUpeU4QDD7bOwq+obmaToLU
m/FvNUKx0l9U2P2ItcsqRCHQut/RxK2pj8GGRDCDco1J
5UAi7hiwP1eEWmbigbPnDQg++QDjegV39vTJQ2c=";
};
```

Obrázek 3: Konfigurace trust anchor

Formát je podobný jako u DNSKEY RR kromě toho, že se vynechává jmenovka „DNSKEY", CLASS a TTL a jméno a materiál veřejného klíče jsou v uvozovkách.

1.3.2 TESTOVÁNÍ

Jakmile byl důvěryhodný klíč nakonfigurován, byla data z této zóny nebo jejích subzón ověřena caching forwarderem. Můžete si to ověřit dotazováním vašeho serveru. Pokud jsou data ověřena caching forwarderem, nastaví jmenný server ad-bit (viz ‚příznaky' v následujícím příkladu).

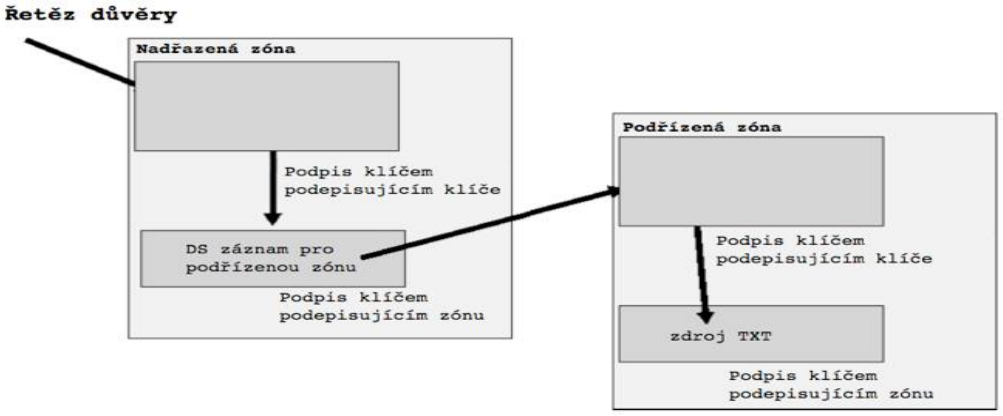
```
; <<>> DiG 9.3.2 <<>> @192.168.2.204 example.net SOA +dnssec +multiline +retry=1
; (1 server found)
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 8343
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;example.net. IN SOA

;; ANSWER SECTION:
example.net.            100 IN SOA ns.example.net. olaf.nlnetlabs.nl. (
                                2002050501 ; serial
                        100       ; refresh (1 minute 40 seconds)
                        200       ; retry (3 minutes 20 seconds)
                        604800    ; expire (1 week)
                        100       ; minimum (1 minute 40 seconds)
                        )
example.net.            100 IN RRSIG SOA 5 2 100 20070429180414 (
                        20070330180414 17000 example.net.
                        lfUdUgEIYyR2/f47RVq6Jdb6otZYnn3d5WGyEPHRiqXD
                        ELkwjRgznub8zDkk7cTCz2ZZ2fLTDwk9XXdwE7NvqCG+
```

```
r1Ti5GTSWW8K/Ehx6hCiRd9NDQI7hVwwqKMOCgaku3nm
yrpGLyMej+zaYapEeQWRRfvWg00DxpERXSlAEow= )

;; Query time: 55 msec
;; SERVER: 192.168.2.204#53(192.168.2.204)
;; WHEN: Fri Mar 30 21:04:20 2007
;; MSG SIZE  rcvd: 267
```

Je důležité, abyste prověřili, že validace pracuje správně. Je možné to provést pomocí log vybavení BIND na počítači, který je konfigurován jako ověřovací rekurzivní jmenný server.
Zprávy BIND určité kategorie mohou být logovány do oddělených kanálů. Kanály určí, kam zprávy doručit a s jakým stupněm závažnosti budou reportovány. Odpovídající kategorie pro ověření DNSSEC je dnssec. V níže uvedeném příkladě jsou chyby kategorie dnnsec směřovány na kanál dnssec_log. Aby byl spuštěn proces ověření, musí se kanál přihlásit s minimální závažností chyby 3.

```
logging {
channel dnssec_log {            // a DNSSEC log channel
file "log/dnssec" size 20m;
print-time yes;       // timestamp the entries
print-category yes;   // add category name to entries
print-severity yes;   // add severity level to entries
severity debug 3;     // print debug message <= 3 t
};

category dnssec  { dnssec_log;  };
}
```

Výstup v souboru log bude obdobný níže uvedenému výstupu. Pokus o pozitivní ověření odpovědi ukazuje, jak se validátor snaží prokázat, že je RRset důvěryhodný sledováním řetězu důvěry k odpovídajícímu důvěryhodnému přístupovému bodu vaší instrukce důvěryhodného klíče. Řetězy důvěry (viz obrázek 4) začínají ověřením podpisu přes DNSKEY RRset, pak jsou tyto klíče použity k ověření DS RRsetu, který odkazuje na DNSKEY RR ve vnořené zóně – který ověří DNSKEY RR ve vnořené zóně – , nebo může být k ověření dat, která požadujete, použito DNSKEY. V logu se zobrazuje aktivita validátoru, který sleduje řetěz důvěry.



Obrázek 4: Řetěz důvěry

```
validating @0x1823e00: example.net SOA: starting
validating @0x1823e00: example.net SOA: attempting positive response validation
validating @0x182a000: example.net DNSKEY: starting
validating @0x182a000: example.net DNSKEY: attempting positive response validation
validating @0x182a000: example.net DNSKEY: verify rdataset: success
validating @0x182a000: example.net DNSKEY: signed by trusted key; marking as secure
validator @0x182a000: dns_validator_destroy
validating @0x1823e00: example.net SOA: in fetch_callback_validator
validating @0x1823e00: example.net SOA: keyset with trust 7
validating @0x1823e00: example.net SOA: resuming validate
validating @0x1823e00: example.net SOA: verify rdataset: success
validating @0x1823e00: example.net SOA: marking as secure
validator @0x1823e00: dns_validator_destroy
```

1.4 VYHLEDÁNÍ TRUST-ANCHORS

Není snadné vyhledat a spravovat trust-anchors, neboli pevné body důvěry. Pokud chcete začít s ověřováním DNSSEC, uvádíme několik míst, kde o daném tématu naleznete více informací.

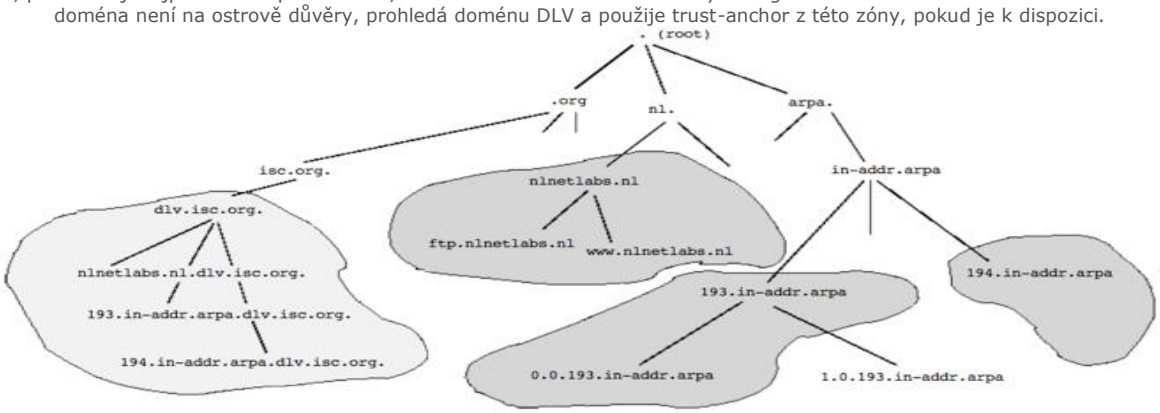
- IANA spravuje klíč kořenové zóny na adrese https://www.iana.org/dnssec/(Nezapomeňte, že je to zabezpečená stránka, prověřte certifikát).
- RIPE NCC spravuje sadu klíčů na své zabezpečené webové stránce na adrese https://www.ripe.net/projects/disi/keys/index.html (Nezapomeňte, že je to zabezpečená stránka, prověřte certifikát).
- Spider DNSSEC se snaží lokalizovat zabezpečené zóny a prověřuje jejich statut. Pro nalezení zabezpečených zón můžete použít tuto stránku. Vizhttp://secspider.cs.ucla.edu/islands.html.

Jelikož není správa trust-anchors vůbec jednoduchá, doporučujeme, abyste si přečetli také oddíl 1.5 zabývající se lookaside validation.

1.5 LOOKASIDE VALIDATION

Vzpomeňte si na obrázek 2. Pokud chcete ověřit všechny tyty ostrovy, budete muset nakonfigurovat mnoho trust-anchors(„pevných bodů důvěry"), jak je uvedeno na příkladu na obrázku 3.
Aby mohl BIND řešit tento problém s absencí delegování zabezpečení z malého počtu trust anchors (ideálně pouze 1, root), podporuje ve verzi 9.3.2 mechanismus pojmenovaný lookaside validation.
Při lookaside validation bude registr DLV spravovat všechny trust-anchors, kterým důvěřujete, že „neškodí". Správci zabezpečených zón, kteří zapisují své trust-anchors do registru DLV a (nestandardní rozšíření) BIND (ve verzi 9.3.2) vám jako operátorovi ověřovacího jmenného serveru umožní využít všechny trust anchors, které jsou ve stromu DLV.

Ve schématu DLV jsou trust anchors publikovány v přidělené doméně (dlv.isc.org na obrázku 5). Kdykoliv ověřovací resolver rozpozná, že je zóna podepsaná, pokusí se ji nejprve ověřit posouzením, zda se nachází na ostrově důvěry konfigurovaném místními trust anchors. Pokud ověřená doména není na ostrově důvěry, prohledá doménu DLV a použije trust-anchor z této zóny, pokud je k dispozici.



Obrázek 5: Trust anchors ve stromu lookaside

1.5.1 KONFIGURACE LOOKASIDE VALIDATION

Následuje obecný popis. Pokud chcete nakonfigurovat DLV společnosti ISC jako vaši autoritativní lookaside doménu, doporučujeme, abyste si přečetlihttp://www.isc.org/ops/dlv/.

V níže uvedeném příkladu předpokládáme, že jste zvolili registr **dlv-registry.org**

Abyste aktivovali lookaside validation, musíte provést dva (další) kroky.

Nakonfigurujte trust anchor pro registr DLV. Provedete to definováním trust anchor pro ostrov důvěry definovaný dlv-registry.org v named.conf. Tento trust-anchor samozřejmě není výhradní, jakýkoliv trust-anchor konfigurovaný ve vaší instrukci důvěryhodných klíčů bude mít přednost před daty v registru DLV.

```
trusted-keys {
//
// this trust-anchor defines dlv-registry.org as a trusted island.
//
"dlv-registry.org." 257 3 5
"AQPXP7B3JTdPPhMI ... u82ggY2BKPQ==";
//
// Other trust anchors below.
//

"nlnetlabs.nl."  3 5
"AQPzzTWMz8qSWI ... zMG1UBYtEIQ==";

"193.in-addr.arpa." 257 3 5
"AwEAAc2RnCT1gj ... pWaM8qXXPN8E=";

"195.in-addr.arpa." 257 3 5
"AwEAAaMN4kOrGai ... DjegV39vTJQ2c=";
};
```

Konfigurování toho, jak je name space DNS ukotven v name space DLV. Přitom se využije instrukce dnssec-lookaside v oddíle nastavení named.conf. Tento příkaz má dva argumenty. První je doména v DNS, na kterou se vztahuje lookaside validation. Obvykle to bývá celý name space, a tak je „.“ (root) nakonfigurován. Druhým argumentem je jméno trust-anchor, kde má být provedeno hledání záznamu DLV. Nejlepší je konfigurovat pouze jeden trust-anchor DLV.

```
options {
// DNSSEC should be turned on, do not forget
dnssec-enable yes;

// This sets the dlv registry "dlv-registry.org"
dnssec-lookaside "." trust-anchor "dlv-registry.org.";

// other options are skipped in this example
};
```

1.5.1.1 TESTOVÁNÍ

Pokud jste logování nakonfigurovali tak, jak je popsáno v oddíle 1.3.2 tj., logované chyby kategorie dnssec jsou směrovány na kanál, který zaznamenává chyby nejméně se stupněm závažnosti 3(severity level 3) a výše, pak bude váš výstup logu pro dotazování na example.net S) A stejný, jako je uvedeno níže.

Za prvé, hodnota výstupu logu pro ověření jednoho dotazu zabírá více než jednu stránku drobným písmem. Na produkčním serveru budou tato data pro několik ověřovacích sekvencí tištěna společně. Bude velice těžké hledat chyby v log souborech na produkčních serverech, když jste nejprve nesledovali, co se děje při jednotlivém dotazu.

Za druhé, struktura je taková, že validátor nejprve vyhledá DNSSEC RRs, upozorní, že tyto záznamy nejsou podle ‚běžného DNSSEC‘ bezpečné a pak pokročí k ověřování DLV.

Za třetí, jsou sestavovány menší řetězy důvěry od trust-anchor DLV přes DNSKEY RR k podpisům dat. Pokuste se sledovat tyto trust anchors v příkladu výstupu. Bude tak snazší identifikovat je v produkčním logu.

```
validating @0x1828c00: . NS: starting
validating @0x1828c00: . NS: looking for DLV
validating @0x1828c00: . NS: plain DNSSEC returns unsecure (.): looking for DLV
validating @0x1828c00: . NS: looking for DLV dlv-registry.org
validating @0x1828c00: . NS: DLV lookup: wait
validating @0x182ba00: example.net SOA: starting
validating @0x182ba00: example.net SOA: looking for DLV
validating @0x182ba00: example.net SOA: plain DNSSEC returns unsecure (.): looking for DLV
validating @0x182ba00: example.net SOA: looking for DLV example.net.dlv-registry.org
validating @0x182ba00: example.net SOA: DNS_R_COVERINGNSEC
validating @0x182ba00: example.net SOA: covering nsec: trust 1
```

```
validating @0x182ba00: example.net SOA: DLV lookup: wait
validating @0x1827c00: dlv-registry.org DLV: starting
validating @0x1827c00: dlv-registry.org DLV: attempting negative response validation
validating @0x182dc00: dlv-registry.org SOA: starting
validating @0x182dc00: dlv-registry.org SOA: attempting positive response validation
validating @0x1830c00: dlv-registry.org DNSKEY: starting
validating @0x1830c00: dlv-registry.org DNSKEY: attempting positive response validation
validating @0x1830c00: dlv-registry.org DNSKEY: verify rdataset: success
validating @0x1830c00: dlv-registry.org DNSKEY: signed by trusted key; marking as secure
validator @0x1830c00: dns_validator_destroy
validating @0x182dc00: dlv-registry.org SOA: in fetch_callback_validator
validating @0x182dc00: dlv-registry.org SOA: keyset with trust 7
validating @0x182dc00: dlv-registry.org SOA: resuming validate
validating @0x182dc00: dlv-registry.org SOA: verify rdataset: success
validating @0x182dc00: dlv-registry.org SOA: marking as secure
validating @0x182f400: example.net.dlv-registry.org DLV: starting
validating @0x182f400: example.net.dlv-registry.org DLV: attempting positive response validation
validating @0x182f400: example.net.dlv-registry.org DLV: keyset with trust 7
validating @0x182f400: example.net.dlv-registry.org DLV: verify rdataset: success
validating @0x182f400: example.net.dlv-registry.org DLV: marking as secure
validator @0x182f400: dns_validator_destroy
validating @0x182ba00: example.net SOA: in dlvfetched: success
validating @0x182ba00: example.net SOA: DLV example.net found
validating @0x182ba00: example.net SOA: dlv_validator_start
validating @0x182ba00: example.net SOA: restarting using DLV
validating @0x182ba00: example.net SOA: attempting positive response validation
validator @0x182dc00: dns_validator_destroy
validating @0x1827c00: dlv-registry.org DLV: in authvalidated
validating @0x1827c00: dlv-registry.org DLV: resuming nsecvalidate
validating @0x182dc00: dlv-registry.org NSEC: starting
validating @0x182dc00: dlv-registry.org NSEC: attempting positive response validation
validating @0x182dc00: dlv-registry.org NSEC: keyset with trust 7
validating @0x182dc00: dlv-registry.org NSEC: verify rdataset: success
validating @0x182dc00: dlv-registry.org NSEC: marking as secure
validator @0x182dc00: dns_validator_destroy
validating @0x1827c00: dlv-registry.org DLV: in authvalidated
validating @0x1827c00: dlv-registry.org DLV: looking for relevant nsec
validating @0x1827c00: dlv-registry.org DLV: nsec proves name exists (owner) data=0
validating @0x1827c00: dlv-registry.org DLV: resuming nsecvalidate
validating @0x1827c00: dlv-registry.org DLV: nonexistence proof found
validator @0x1827c00: dns_validator_destroy
validating @0x1828c00: . NS: in dlvfetched: ncache nxrrset
validating @0x1828c00: . NS: DLV not found
validating @0x1828c00: . NS: marking as answer
validator @0x1828c00: dns_validator_destroy
validating @0x182e000: example.net DNSKEY: starting
validating @0x182e000: example.net DNSKEY: looking for DLV
validating @0x182e000: example.net DNSKEY: plain DNSSEC returns unsecure (.): looking for DLV
validating @0x182e000: example.net DNSKEY: looking for DLV example.net.dlv-registry.org
validating @0x182e000: example.net DNSKEY: DLV example.net found
validating @0x182e000: example.net DNSKEY: dlv_validator_start
validating @0x182e000: example.net DNSKEY: restarting using DLV
validating @0x182e000: example.net DNSKEY: attempting positive response validation
validating @0x182e000: example.net DNSKEY: dlv_validatezonekey
validating @0x182e000: example.net DNSKEY: Found matching DLV record: checking for signature
validating @0x182e000: example.net DNSKEY: verify rdataset: RRSIG failed to verify
validating @0x182e000: example.net DNSKEY: verify rdataset: success
validating @0x182e000: example.net DNSKEY: marking as secure
validator @0x182e000: dns_validator_destroy
validating @0x182ba00: example.net SOA: in fetch_callback_validator
validating @0x182ba00: example.net SOA: keyset with trust 7
validating @0x182ba00: example.net SOA: resuming validate
validating @0x182ba00: example.net SOA: verify rdataset: success
validating @0x182ba00: example.net SOA: marking as secure
validator @0x182ba00: dns_validator_destroy
```

Pokud využíváte lookaside validation, je posuzování výstupu logu v případě poškození dat zóny náročné. Níže je uveden výstup validátoru, když se snaží zjistit, zda je dotaz, který navrátí poškozený výsledek, platný nebo ne. Řešení se ukazuje v několika posledních řádcích.

```
validating @0x1828c00: . NS: starting
validating @0x1828c00: . NS: looking for DLV
validating @0x1828c00: . NS: plain DNSSEC returns unsecure (.): looking for DLV
validating @0x1828c00: . NS: looking for DLV dlv-registry.org
validating @0x1828c00: . NS: DLV lookup: wait
validating @0x182be00: corrupt.example.net A: starting
validating @0x182be00: corrupt.example.net A: looking for DLV
validating @0x182be00: corrupt.example.net A: plain DNSSEC returns unsecure (.): looking for DLV
validating @0x182be00: corrupt.example.net A: looking for DLV corrupt.example.net.dlv-registry.org
validating @0x182be00: corrupt.example.net A: DNS_R_COVERINGNSEC
validating @0x182be00: corrupt.example.net A: covering nsec: trust 1
validating @0x182be00: corrupt.example.net A: DLV lookup: wait
validating @0x182de00: dlv-registry.org DLV: starting
validating @0x182de00: dlv-registry.org DLV: attempting negative response validation
validating @0x1829c00: dlv-registry.org SOA: starting
validating @0x1829c00: dlv-registry.org SOA: attempting positive response validation
validating @0x1830400: corrupt.example.net.dlv-registry.org DLV: starting
validating @0x1830400: corrupt.example.net.dlv-registry.org DLV: attempting negative response validation
```

validating @0x1830c00: dlv-registry.org SOA: starting
validating @0x1830c00: dlv-registry.org SOA: attempting positive response validation
validating @0x1832c00: dlv-registry.org DNSKEY: starting
validating @0x1832c00: dlv-registry.org DNSKEY: attempting positive response validation
validating @0x1832c00: dlv-registry.org DNSKEY: verify rdataset: success
validating @0x1832c00: dlv-registry.org DNSKEY: signed by trusted key; marking as secure
validator @0x1832c00: dns_validator_destroy
validating @0x1829c00: dlv-registry.org SOA: in fetch_callback_validator
validating @0x1829c00: dlv-registry.org SOA: keyset with trust 7
validating @0x1829c00: dlv-registry.org SOA: resuming validate
validating @0x1829c00: dlv-registry.org SOA: verify rdataset: success
validating @0x1829c00: dlv-registry.org SOA: marking as secure
validator @0x1829c00: dns_validator_destroy
validating @0x182de00: dlv-registry.org DLV: in authvalidated
validating @0x182de00: dlv-registry.org DLV: resuming nsecvalidate
validating @0x1830c00: dlv-registry.org SOA: in fetch_callback_validator
validating @0x1830c00: dlv-registry.org SOA: keyset with trust 7
validating @0x1830c00: dlv-registry.org SOA: resuming validate
validating @0x1830c00: dlv-registry.org SOA: verify rdataset: success
validating @0x1830c00: dlv-registry.org SOA: marking as secure
validator @0x1830c00: dns_validator_destroy
validating @0x1830400: corrupt.example.net.dlv-registry.org DLV: in authvalidated
validating @0x1830400: corrupt.example.net.dlv-registry.org DLV: resuming nsecvalidate
validating @0x1830c00: example.net.dlv-registry.org NSEC: starting
validating @0x1830c00: example.net.dlv-registry.org NSEC: attempting positive response validation
validating @0x1830c00: example.net.dlv-registry.org NSEC: keyset with trust 7
validating @0x1830c00: example.net.dlv-registry.org NSEC: verify rdataset: success
validating @0x1830c00: example.net.dlv-registry.org NSEC: marking as secure
validator @0x1830c00: dns_validator_destroy
validating @0x1830400: corrupt.example.net.dlv-registry.org DLV: in authvalidated
validating @0x1830400: corrupt.example.net.dlv-registry.org DLV: looking for relevant nsec
validating @0x1830400: corrupt.example.net.dlv-registry.org DLV: nsec range ok
validating @0x1830400: corrupt.example.net.dlv-registry.org DLV: resuming nsecvalidate
validating @0x1830400: corrupt.example.net.dlv-registry.org DLV: in checkwildcard: *.example.net.dlv-registry.org
validating @0x1830400: corrupt.example.net.dlv-registry.org DLV: looking for relevant nsec
validating @0x1830400: corrupt.example.net.dlv-registry.org DLV: nsec range ok
validating @0x1830400: corrupt.example.net.dlv-registry.org DLV: nonexistence proof found
validator @0x1830400: dns_validator_destroy
validating @0x1829c00: dlv-registry.org NSEC: starting
validating @0x1829c00: dlv-registry.org NSEC: attempting positive response validation
validating @0x1829c00: dlv-registry.org NSEC: keyset with trust 7
validating @0x1829c00: dlv-registry.org NSEC: verify rdataset: success
validating @0x1829c00: dlv-registry.org NSEC: marking as secure
validator @0x1829c00: dns_validator_destroy
validating @0x182de00: dlv-registry.org DLV: in authvalidated
validating @0x182de00: dlv-registry.org DLV: looking for relevant nsec
validating @0x182de00: dlv-registry.org DLV: nsec proves name exists (owner) data=0
validating @0x182de00: dlv-registry.org DLV: resuming nsecvalidate
validating @0x182de00: dlv-registry.org DLV: nonexistence proof found
validator @0x182de00: dns_validator_destroy
validating @0x1828c00: . NS: in dlvfetched: ncache nxrrset
validating @0x1828c00: . NS: DLV not found
validating @0x1828c00: . NS: marking as answer
validator @0x1828c00: dns_validator_destroy
validating @0x182be00: corrupt.example.net A: in dlvfetched: ncache nxdomain
validating @0x182be00: corrupt.example.net A: looking for DLV example.net.dlv-registry.org
validating @0x182be00: corrupt.example.net A: DLV lookup: wait
validating @0x1829c00: example.net.dlv-registry.org DLV: starting
validating @0x1829c00: example.net.dlv-registry.org DLV: attempting positive response validation
validating @0x1829c00: example.net.dlv-registry.org DLV: keyset with trust 7
validating @0x1829c00: example.net.dlv-registry.org DLV: verify rdataset: success
validating @0x1829c00: example.net.dlv-registry.org DLV: marking as secure
validator @0x1829c00: dns_validator_destroy
validating @0x182be00: corrupt.example.net A: in dlvfetched: success
validating @0x182be00: corrupt.example.net A: DLV example.net found
validating @0x182be00: corrupt.example.net A: dlv_validator_start
validating @0x182be00: corrupt.example.net A: restarting using DLV
validating @0x182be00: corrupt.example.net A: attempting positive response validation
validating @0x182a000: example.net DNSKEY: starting
validating @0x182a000: example.net DNSKEY: looking for DLV
validating @0x182a000: example.net DNSKEY: plain DNSSEC returns unsecure (.): looking for DLV
validating @0x182a000: example.net DNSKEY: looking for DLV example.net.dlv-registry.org
validating @0x182a000: example.net DNSKEY: DLV example.net found
validating @0x182a000: example.net DNSKEY: dlv_validator_start
validating @0x182a000: example.net DNSKEY: restarting using DLV
validating @0x182a000: example.net DNSKEY: attempting positive response validation
validating @0x182a000: example.net DNSKEY: dlv_validatezonekey
validating @0x182a000: example.net DNSKEY: Found matching DLV record: checking for signature
validating @0x182a000: example.net DNSKEY: verify rdataset: RRSIG failed to verify
validating @0x182a000: example.net DNSKEY: verify rdataset: success
validating @0x182a000: example.net DNSKEY: marking as secure
validator @0x182a000: dns_validator_destroy
validating @0x182be00: corrupt.example.net A: in fetch_callback_validator
validating @0x182be00: corrupt.example.net A: keyset with trust 7
validating @0x182be00: corrupt.example.net A: resuming validate
validating @0x182be00: corrupt.example.net A: verify rdataset: RRSIG failed to verify

validating @0x182be00: corrupt.example.net A: failed to verify rdataset
validating @0x182be00: corrupt.example.net A: verify failure: RRSIG failed to verify
validating @0x182be00: corrupt.example.net A: no valid signature found
validator @0x182be00: dns_validator_destroy

1. KONFIGURACE REKURZIVNÍHO JMENNÉHO SERVERU PRO OVĚŘOVÁNÍ ODPOVĚDÍ
1.6 TIPY NA ODSTRAŇOVÁNÍ NĚKTERÝCH PROBLÉMŮ

Předpokládejme, že jste nakonfigurovali trust anchor a zaznamenali jste problémy. Například, váš jmenný server odpovídá na určité dotazy „SERVFAIL". „SERVFAIL" je základním odpovědním kódem, který navrací ověřovací jmenný server, pokud jsou příznaky dat podvodné. Podvodná data mohou být zapříčiněna dvěma důvody. Útočí na vás, nebo jste zaznamenali chybu v konfiguraci operátora jedné ze zón v řetězu důvěry, nebo operátora ověřovacího rekurzivního jmenného serveru.

Kromě prohlížení logů je vám k dispozici řada nástrojů (viz III). K vyhledání problémů, ke kterým došlo v důsledku špatné konfigurace, nebo chyb na vašem ověřovacím jmenném serveru, nebo v důsledku problémů s podepsanými zónami budete potřebovat strategii pro odstraňování problémů. Jedním z postupů, které můžete využít je nejprve využít „drill" (viz 6) nebo „dig" (viz 7) k provedení ‚sigchase' nebo ‚trace' s klíčem zkopírovaným do vašeho souborového systému, kdy obejdete ověřovací rekurzivní jmenný server. Tímto způsobem budete moci zkontrolovat, zda může být z dat skutečně sestaven řetěz důvěry. Ujistěte se, že při sledování dat používáte správný „pevný bod důvěry"(trust-anchor).

Pokud jste ověřili, že může být řetěz důvěry z dat DNS sestaven, je na čase odstranit problémy s ověřovacím jmenným serverem. To je snadné, pokud máte přístup do souborů logu, v opačném případě to může být náročné. Můžete využít „dig" k dotazování na ověřovací jmenný server s příznakem a bez příznaku +c. Tento příznak nastavuje v dotazu bit, který dává jmennému serveru instrukce, aby neprováděl ověření. Pokud testujete jednotlivé články řetězu důvěry („drill" je vrátil zpět při volbě sledovat) můžete nalézt nesrovnalosti, které poukazují na to, že byl konfigurován prošlý trust anchor. Začněte dotazováním DNSKEY RRsetů, u kterých předpokládáte, že se vyskytuje neplatný trust-anchor a pokračujte tímto způsobem. Nebo se můžete dotazovat na data, která hledáte s využitím dat v RRSIG RR, abyste zjistili, kterého DNSKEY RR se dotazovat, pak se dotažte DS RR toho samého a postupujte tímto způsobem (podobně jako u ‚sigchase' v „drill").

Při odstraňování problémů existuje spousta chyb, ke kterým snadno dochází. Podívejte se na následující příklad.

Ve svém sloupci ISP [6, 7, 8] dokumentuje Geoff Huston své zkušenosti jako jednoho z prvních, kdo řešení implementoval. Poslepu konfiguruje trust-anchor pro zónu „nlnetlabs.nl". Zóna byla podepsána v experimentálním nastavení, s jehož pomocí byly servery v zóně nakonfigurovány různými verzemi protokolu. V tomto případě jeden ze serverů neposkytoval spolu s odpověďmi RRSIG, což mohlo vést k opakujícím se, ale těžko předvídatelným chybám.

Ponaučením nám mohou být dvě věci: Nikdy nekonfigurujte trust anchors u ověřovacích resolverů naslepo a ujistěte se, že když poskytujete zóny, jsou všechny servery v souladu se specifikací protokolu DNSSEC.

Další chybou je, že má jeden z RRsetů v řetězu důvěry prošlý podpis. Prověřte to kontrolou datových polí v RRSIG RR.

Problematičtější může být vyhledání rotace, kde byly DNSKEY RR nebo RRSIG RR vymazány příliš brzy, a tak nastal nesoulad mezi daty v cache a daty, která je potřeba ověřit (viz také oddíl 4). Použití +cd při „dig" a hledání v TTL může pomoci rozlišit, zda se pokoušíte ověřit RRSIG, pro které není k dispozici DNSKEY (nebo naopak).

2. ZABEZPEČENÍ ZÓNY DNS
2.1 ÚVOD

Pokud byla zóna podepsána a její klíč byl nakonfigurován na ověřovacím rekurzivním jmenném serveru, obvykle o ní hovoříme jako o „ostrovu důvěry". Očividně nemá žádnou zabezpečenou nadřazenou zónu a leží osamoceně v moři jiných nezabezpečených domén. Vytvoření „ostrova důvěry" je obvykle prvním krokem na cestě stát se součástí zabezpečeného DNS. „Ostrov důvěry" zůstává „nedůvěryhodný" pro resolvery, které nemají pro doménu konfigurovaný „pevný bod důvěry"(trust-anchor).

Pokud se vlastník zóny rozhodne vytvořit „ostrov důvěry" podepíše své zóny a distribuuje „důvěryhodné přístupové body" systémovým administrátorům, kteří chtějí ověřit data zóny. Jakmile je ostrov důvěry vytvořen, může se stát součástí stromu důvěry prostřednictvím výměny „důvěryhodného přístupového bodu" s nadřazenou zónou.

Po vytvoření párů klíčů používaných pro podepisování a ověřování chceme podepsat data zóny pro naši vlastní organizaci (např. example.net) a nakonfigurovat caching forwardery naší sítě v organizaci pro ověření dat oproti veřejnému klíči naší organizace.

Dále v textu předpokládáme, že jsou jména domén vaší organizace spravována v jedné zóně. Pokud je správa jména domény delegována na subzóny, viz oddíl Kapitola 3, „Delegování podepisující autority; přechod na globální zabezpečení".

Podepsání dat zóny je úkolem administrátora zóny, konfigurace caching forwarderu je úkolem systémových administrátorů.

Příklady vycházejí z příkladové zóny v oddíle 2.4, obrázek 7.

2. ZABEZPEČENÍ ZÓNY DNS
2.2 KONFIGURACE AUTORITATIVNÍCH SERVERŮ

Je třeba nakonfigurovat všechny autoritativní servery tak, aby komunikovaly s protokolem DNSSEC. Jak to lze provést v BIND je vysvětleno v příloze A. Základními kroky jsou kompilace bind pomocí openssl a povolení dnssec prostřednictvím **dnssec-enable yes**; příkazu v oddíle možnosti **named.conf**.

To je k tomuto tématu vše.

2. ZABEZPEČENÍ ZÓNY DNS
2.3 VYTVÁŘENÍ PÁRŮ KLÍČŮ
2.3.1 ZÁSADY ÚDRŽBY KLÍČŮ

Před vygenerováním klíčů se budete muset zamyslet nad zásadami údržby klíčů. Tyto zásady by měly řešit

- Jaký bude rozměr vašich klíčů?
- Rozdělíte funkční závislost klíčů podepisujících klíče a klíčů podepisujících zóny?
- Jak často budete rotovat klíče?
- Jak dostanou systémoví administrátoři, kteří zamýšlí využívat vaši zónu jako trust anchor, správný veřejný klíč a jaký mechanismus jim nabídnete pro povolení ověřování autenticity vašeho veřejného klíče?
- Jak upozorníte na rotaci klíče nebo jak se ujistíte, že všechny zúčastněné strany jsou si vědomy rotace?

Některé z těchto problémů mohou být vyřešeny snadno. Například, vaše organizace může vytvořit mechanizmus na distribuci veřejných klíčů, kde mohou být jasné způsoby publikace nadcházející rotace jako je možnost publikovat událost v novinách organizace. Nebo je možné upozornit všechny zúčastněné strany e-mailem, pokud je hierarchie organizace X.509 dostupná pro ověřování e-mailů.

2.3.1.1 KLÍČE PODEPISUJÍCÍ KLÍČE A KLÍČE PODEPISUJÍCÍ ZÓNY

Autor se domnívá, že používání klíčů podepisujících zóny a klíčů podepisujících klíče je osvědčenou metodou (viz také Kapitola 4, „Rotace klíčů"). Klíče podepisující klíče jsou obvykle prvními klíči z vaší zóny, které se využívají při sestavování řetězu pravomocí pro data, která je potřeba ověřit. Proto jsou tyto klíče často nazývány klíče důvěryhodného přístupového bodu (nebo klíč SEP). Tyto klíče SEP jsou těmi, které byste si měli vyměňovat s vaší nadřazenou zónou, nebo těmi, které konfigurují ověřovací resolvery jako jejich trust anchors.

V tomto dokumentu předpokládáme, že používáte oddělené klíče podepisující klíče a klíče podpisující zóny, a že klíče podpisující klíče jsou vyhrazeny pro použijí jako klíče důvěryhodných přístupových bodů a mohou být identifikovány bitem SEP[10] v příznakovém poli, které se odlišuje.

2.3.2 VYTVÁŘENÍ KLÍČŮ

Usage:
dnssec-keygen -a alg -b bits -n type [options] name

Version: 9.3.2
Required options:
-a algorithm: RSA | RSAMD5 | DH | DSA | RSASHA1 | HMAC-MD5
-b key size, in bits:
RSAMD5: [512..4096]

```
RSASHA1: [512..4096]
DH: [128..4096]
DSA: [512..1024] and divisible by 64
HMAC-MD5: [1..512]
-n nametype: ZONE | HOST | ENTITY | USER | OTHER
name: owner of the key
Other options:
-c <class> (default: IN)
-e use large exponent (RSAMD5/RSASHA1 only)
-f keyflag: KSK
-g <generator> use specified generator (DH only)
-t <type>: AUTHCONF | NOAUTHCONF | NOAUTH | NOCONF (default: AUTHCONF)
-p <protocol>: default: 3 [dnssec]
-s <strength> strength value this key signs DNS records with (default: 0)
-r <randomdev>: a file containing random data
-v <verbose level>
-k : generate a TYPE=KEY key
Output:        K<name>+<alg>+<id>.key, K<name>+<alg>+<id>.private
```

Obrázek 6: argumenty dnssec-keygen

dnssec-keygen je nástroj, který používáme ke generování párů klíčů. Argumenty, které musíme do dnssec-keygen zadat ukazuje obrázek 6. Výstup nalezneme ve dvou souborech. Jména souborů obsahují příslušné informace:

```
Kdomain_name+algorithm_id+key_id.extension
```

domain_name je jméno zadané do příkazové řádky. Využívají jej ostatní nástroje BIND DNSSEC. Pokud použijete jiné jméno než je jméno domény, může to být pro tyto nástroje zavádějící. algorithm_id identifikuje použitý algoritmus: 1 pro RSAMD5, 3 pro DSA, 5 pro RSASHA1 a 54 pro HMAC-MD5 (pouze TSIG). key_id je identifikátor materiálu klíče. key_id je využíváno zdrojovým záznamem RRSIG. Rozšířením je key nebo private; první je veřejný klíč a druhý je soukromý klíč.

Pomocí algoritmu RSASHA1 vytvoříme pár klíčů podepisujících zóny pro example.net:

```
# dnssec-keygen -r/dev/random  -a RSASHA1 -b 1024 -n ZONE example.net
Kexample.net.+005+17000
```

Podle pokynů z oddílu 2.3.1 budete muset vytvořit také klíče SEP. Pomocí dnssec-keygen vytvořte s nastavením bitu SEP určeným příznakem KSK –f klíče.

```
# dnssec-keygen -r/dev/random  -f KSK -a RSASHA1 -b 1280  -n ZONE example.net
Kexample.net.+005+49656
```

Podívejme se na obsah těchto souborů.

```
cat Kexample.net.+005+17000.key
example.net. IN DNSKEY 256 3 5 (
AQPI4+0M1V055RS2Hqv+8w8V20Dh+SQmFzHQtZMuzLH3UxWE0GmG5Gfj
ijandJeAZTKLpERXB6RfHTHGG8lD3IO1azWN6DiVFEVzgr0otAdDonfY
+oEsRw== )
```

Veřejný klíč (rozšíření .key) je přesně takový, jak se objevuje ve vašem zónovém souboru. Nezapomeňte, že hodnota TTL není určena. Klíč má „příznakovou" hodnotu 256. Jelikož je tato hodnota celým číslem, nemůže být klíč označen jako klíč SEP a měl by být použit k podepsání zóny. Soukromý klíč (rozšíření .private) obsahuje všechny parametry, které tvoří soukromý klíč vytvořený algoritmem RSASHA1. Soukromý klíč klíče RSA obsahuje odlišné parametry než DSA. Zde je soukromý klíč (se zkráceným base64 materiálem):

```
cat Kexample.net.+005+17000.private
Private-key-format: v1.2
Algorithm: 5 (RSASHA1)
Modulus: yOPtDNVdOeUUth6r/vMPFdtA4fkkJhcx0LWTLsyx91MVhNBphu...
PublicExponent: Aw==
PrivateExponent: he1Iszjo0UNjJBRyqfdfY+eAlqYYGWTL4HkMyd3L+j...
Prime1: +X0kNW1JrepBnVw5o9fDUyWAT5zqxKt0YR4vJZ19991tLZAmdO4...
Prime2: ziIX5qfpZGBuzfd847TqtDfYcwv5UfUrPAIa/11g3leUUNERmsB...
Exponent1: plNtePOGc/GBE5LRF+Us4hkANRNHLcei62l0w75T+pOeHmAZ...
Exponent2: iWwP7xqbmEBJ3qT97SNHIs/logf7i/jHfVa8qj5AlDpi4Ith...
Coefficient: rmmgD9P7/ywQJ4F0epdGqOUoQZmqrPQsraDTD8vkU1wLju...
```

Tento soukromý klíč by měl zůstat důvěrný, tj. přístupová oprávnění k souboru by měla být natavena tak, že k nim bude mít administrátor zóny přístup v případě, že bude třeba zónu podepsat. Nástroje BIND budou implicitně vyhledávat klíče v adresáři, kde se provádí podepisování (viz oddíl 2.4), což nemusí být nejbezpečnější místo vašeho OS.

2. ZABEZPEČENÍ ZÓNY DNS

2.4 PODEPISOVÁNÍ ZÓNY

Pokud vytváříte páry klíčů, měli byste je připojit k vašim zónovým souborům. Podívejte se na příklad na obrázku 7, kde používáme pro připojení klíčů příkaz**$include**. Před podepsáním zvyšujeme sériové číslo v záznamu SOA.

V níže uvedeném příkladu použijeme pro klíče podpisující klíče a klíče podepisující zóny klíče typu RSASHA1.

```
; example.net zone
;
$TTL 100
$ORIGIN example.net.
@                100     IN      SOA     ns.example.net. (
                 olaf.nlnetlabs.nl.
                 2002050501
                 100
                 200
                 604800
                 100
                 )
```

```
                NS          ns.example.net.

ns.example.net.     A           192.168.2.203

a               A           192.168.2.1
b               A           192.168.2.2


*               A           192.168.2.10
b.a             A           192.168.2.11


; These are the keys that need to be publised in the DNSKEY RRset
;
$include Kexample.net.+005+17000.key    ; ZSK
$include Kexample.net.+005+49656.key    ; KSK
```

Obrázek 7: Příkladová zóna example.net

Jakmile je klíč přidán do zónového souboru, jsme připraveni podepsání zóny pomocí nástroje dnssec-signzone (všechny argumenty viz obrázek 8). Pro určení původu zóny používáme příznak –o; implicitně je původ odvozován ze jména zónového souboru.

Pomocí '-k key_name' určujeme, který klíč má být použit jako klíč podepisující klíče. Klíč podepíše pouze DNSKEY RRset v apexu zóny. Klíče, které zadáváme jako argumenty na konci příkazu jsou užívány k podpisu všech dat RR, pro která je zóna autoritativní. Pokud neurčíte klíče, použije BIND ty, pro které jsou přidány veřejné klíče v zóně a pro rozlišení mezi klíči podepisujícími klíče a klíči podepisujícími zóny použije příznak SEP.

V praxi nedoporučujeme, abyste se spoléhali na implicitní nastavení, protože ve scénářích rotace klíčů budete mít veřejný klíč ve vašem zónovém souboru, ale nebudete jej chtít používat pro podepsání zóny (abyste předešli zdvojení podpisů, a tak delším časům generování podpisu a zabírání více zdrojů na vašem jmenném serveru). Níže uvádíme příkaz vydaný k podepsání zóny pomocí 49656 klíče jako klíč podepisujícího klíče a 17000 klíč jako klíč podpisující zóny.

```
/usr/local/sbin/dnssec-signzone  \
-o example.net  \
-k Kexample.net.+005+49656 \
db.example.net \
Kexample.net.+005+17000.key
```

Obrázek níže ukazuje podepsaný zónový soubor. Všimněte si, že apex DNSKEX RRset je pouze RRset se dvěma podpisy vytvořený pomocí klíčů podepisujících zóny a klíčů podepisujících klíče. Ostatní RRsety jsou podepsány pouze klíči podepisujícími zóny.

Proces podepisování dokončil následující:

- Roztřídil zónu podle ‚kanonického' pořádku.
- Pro každou jmenovku vložil záznamy NSEC.
- Přidal key-id jako komentář ke každému DNSKEY-record.
- Podepsal DNSKEY RR set dvěma klíči; klíčem podepisujícím klíče a klíčem podepisujícím zóny.
- Podepsal ostatní RRsety klíčem podepisujícím zóny.
- Vytvořil dva soubory, dsset-example.net a keyset-example.net. Tyto soubory jsou důležité při sestavování řetězu důvěry. Implicitně se soubory vytvářejí v ‚aktuálním adresáři' tj., adresáři, ve kterém spouštíte příkaz thednssec-signzone, ale když určíte tomuto adresáři -d, pak budou soubory vytvořeny tam.

Podpisy byly vytvořeny s implicitní životností 30 dnů od okamžiku podepsání. Jakmile podpisy vyprší, nemohou být data ověřena a zóna bude označena jako ‚podvodná'. Proto musíte vaši zónu znovu podepsat nejpozději do 30 dnů! Níže se zabýváme opětovným podepsáním zóny.

Podepsaná zóna je uložena v db.example.net.signed, ujistěte se, že jste nakonfigurovali named tak, aby použil tento soubor při poskytování zón.

```
Usage:
dnssec-signzone [options] zonefile [keys]

Version: 9.3.2
Options: (default value in parenthesis)
-c class (IN)
-d directory
directory to find keyset files (.)
-g: generate DS records from keyset files
-s [YYYYMMDDHHMMSS|+offset]:
RRSIG start time - absolute|offset (now - 1 hour)
-e [YYYYMMDDHHMMSS|+offset|"now"+offset]:
RRSIG end time  - absolute|from start|from now (now + 30 days)
-i interval:
cycle interval - resign if < interval from end ( (end-start)/4 )
-v debuglevel (0)
-o origin:
zone origin (name of zonefile)
-f outfile:
file the signed zone is written in (zonefile + .signed)
-r randomdev:
a file containing random data
-a: verify generated signatures
-p: use pseudorandom data (faster but less secure)
-t: print statistics
-n ncpus (number of cpus present)
-k key_signing_key
-l lookasidezone
-z: ignore KSK flag in DNSKEYs
Signing Keys: (default: all zone keys that have private keys)    keyfile (Kname+alg+tag)
```

Obrázek 8: Argumenty dnssec-signzone

```
; File written on Fri Mar 30 21:04:12 2007
; dnssec\_signzone version 9.3.2
```

```
example.net.            100     IN SOA  ns.example.net. olaf.nlnetlabs.nl. (
                                2002050501 ; serial
                        100       ; refresh (1 minute 40 seconds)
                        200       ; retry (3 minutes 20 seconds)
                        604800     ; expire (1 week)
                        100       ; minimum (1 minute 40 seconds)
                                )
                100     RRSIG   SOA 5 2 100 20070429180412 (
                                20070330180412 17000 example.net.
                                Q7QT/Y3MhD9Zx6/UK3jy09lCv+hEox1Sf4sI
                                hYjSp873y962zXRtXcFp9oa2LgBq/aGM+LLV
                                qFOpK7WlhxyZRxKdNbl4f6F7hRps11k7bPCv
                                ZbjJ0nsiRsFfhjtCyll7EWeXYEmkUqtsbKC4
                                dsUEc/jjo8jm1CBntmk7FBLtw9I= )
                        100     NS      ns.example.net.
                100     RRSIG   NS 5 2 100 20070429180412 (
                                20070330180412 17000 example.net.
                                k4Dy4YRfMwTUsKtpjWJ2IyFFvnLbR7TBVeU3
                                j3+y28E5r8l8g6Se4peXfzB+2Ks+wUPmZwzm
                                UFt9KVNzIJ7GA26VmEACWcuDGIvS65GuCrO4
                                +sNmfsZXW2txtDWedvlnmytDxEIKvbtRu4QJ
                                rgCGlYaoC1v6/mNDa6ZBlDYxeLk= )
        100     NSEC    *.example.net. NS SOA RRSIG NSEC DNSKEY
                100     RRSIG   NSEC 5 2 100 20070429180412 (
                                20070330180412 17000 example.net.
                                fEnDtTdDyYrC7DqMTkIPaYeo2yml2Woq97Gc
                                oAP50Rc0bCDdIz+UPXfj3PVSOuFunaHNYYUq
                                mangw8n75B9g7Uqrb6brAteEJSHqKkxYUcdG
                                AzpY+m+Yl82TjxOj7jFGZxiwqyjF2PQLaY13
                                wfrjb5CVDnF8WbREP6/Jhm45Plg= )
                                100     DNSKEY  256 3 5 (
                                AQPI4+0M1V055RS2Hqv+8w8V20Dh+SQmFzHQ
                                tZMuzLH3UxWE0GmG5GfjijandJeAZTKLpERX
                                B6RfHTHGG8lD3IO1azWN6DiVFEVzgr0otAdD
                                onfYF8gUT03ZnRcXlkJk41h12NOfq6rkODaF
                                nfMHCppI3WZ/MJqe+9hLJtis+oEsRw==
                                ) ; key id = 17000
                                100     DNSKEY  257 3 5 (
                                AQOzgs4qea+ImJ1OCworkabHqFnvPKybVT7b
                                nDIkJ2HvXWslbwNWJ66Ox3N6ftpCTc9wWBMw
                                5+xOh7ilTwFPruMa2gURwEywZaMG9ipILOXm
                                KO4a5I+8R2QTH4BM0WaIKnv5jCHose/l9LL3
                                Y8MApsjP6gOWNM8b9aVTjBFnf0xEF7sOSBBB
                                E4G2/og5Fr+H8DYaotqgJ3nrzRfYA0gSXwwb
                                ) ; key id = 49656
                        100     RRSIG   DNSKEY 5 2 100 20070429180412 (
                                20070330180412 17000 example.net.
                                hFcUzcQnsQbiOhnQS+oDf+/g7kGsDqpWrqZV
                                kedbecrJQy9RK/IMy5WFmhP9g158PReEFdES
                                lh/wRD+j0hKqOpRXKbGeOi231AUSqvhou8VF
                                zBq5vSe8NiRK+1eLzmHerA8GEXqgP8mxdy/A
                                5Q/8hdyA/DSYe8u1gfrB41MSmj4= )
                        100     RRSIG   DNSKEY 5 2 100 20070429180412 (
                                20070330180412 49656 example.net.
                                oyum/nlrNZ7XdxiA/tpdmzAp16YmXS8YTPyJ
                                EdCJUYZPp3ioJ83EnOmxzQK+YLqwjw8thNGw
                                cCOd3sKBjlR04C0d+Wjth8OEXuqo2bWkKdfI
                                IyIq3SQr5rrNFpgJtxAsse2j6sdDVqOV3VSf
                                syPHRsOd2DpGmFaoiLq0ORpRfzyV6afk6u4L
                                FgIOJJQXkZiLWbLA4GfGHr1g6Ae3Bf1FtQ== )
                *.example.net.          100     IN A    192.168.2.10
                        100     RRSIG   A 5 2 100 20070429180412 (
                                20070330180412 17000 example.net.
                                aRpTVJ7TI1LJ706fBMOLhorgA601KVUE6j4t
                                AFJTzt7LQTnZQF/+czAkW39ETFYt351psfXB
                                ZThiDsLwkuLe5buJPzftDEhbB4CJrYd9/fFe
                                dX6KdI0PTizO20dpMJld0sP0NO/JTDwSrhaa
                                xAYeCCz/1C/AkYT9hKpa5YXVKjY= )
                        100     NSEC    a.example.net. A RRSIG NSEC
                100     RRSIG   NSEC 5 2 100 20070429180412 (
                                20070330180412 17000 example.net.
                                X9ViqG4mT8lDS+hckEdk2XqwHTpnDxRp1dxK
                                Gfayhgt+svMtXmhjcJZ0Kz/c6ByU/JIXv/eE
                                3gsoJFMNU2iUIbWbMeX6l4Gy3B+gMeQ9F+fT
                                FHuwVMHOVLih+Zlm3OpDbUUBwYJj4XOBY6aA
                                7mBaSZ7lUEnjDF3MeCZCTshkuUs= )
                a.example.net.          100     IN A    192.168.2.1
                        100     RRSIG   A 5 3 100 20070429180412 (
                                20070330180412 17000 example.net.
                                oN1QemG7B47dWBod6ecdvmb87VegMa5mN16P
                                CsJGnIwNLOWDWxvSiuKOH/COvrGIHegqQle/
                                VQBvjl6XVPdxNQ0HWUlIRCaFKYEiMcxdn6er
                                bH5nHYNHsFD4SHy2q0uEi6kqlRpNSuza+oDi
                                ASbGdWhELy5gFsdpZIFS1naUNNM= )
                        100     NSEC    b.a.example.net. A RRSIG NSEC
                100     RRSIG   NSEC 5 3 100 20070429180412 (
```

```
20070330180412 17000 example.net.
iXgRHchMpXLYwrfrwSAfqAHHhbf7EePgaiEa
3U2GhkvlhKCACh6pGA4vZwFgey4cnA4nP+rk
CE2p0BNGhPC0Mdx3eby6NX2Dbuzkc6TctFSj
Y1eZLXdju8EfAdk5Jhs+9noBcZFNNTM8Wid5
8MW2ur0187CTdWIXq8jAqodngAs= )
b.a.example.net.        100     IN A    192.168.2.11
100     RRSIG   A 5 4 100 20070429180412 (
20070330180412 17000 example.net.
hf5hrMDv+BlTkTECLuEGukknfDuH/nPpUWpz
+gfzzeRChQReSXDXJn1259jXSkt8IUhD3v7G
KY1MTQEOQlWkB8OOC9r86yAW+hLnyIWXqe+t
I+JR7ga00qx4gy6LhMYlQ/2Ywdo= )
100     NSEC    b.example.net. A RRSIG NSEC
100     RRSIG   NSEC 5 4 100 20070429180412 (
20070330180412 17000 example.net.
Kon6z25uqnHpGc91BcbR6hs0KKQ4z0nm9SsE
OJFgSfhYDBygiHl84eQFLvP/BjsJd4BQHxke
v9X/J6QK4vM1NDsVVvYdPw7ybh6MOLU8Lo8p
k4lbeOXvChD8ONjDebULKP+Iz6fxXjQ6WsG7
NvHskJBZfZPHxigi+SlQILHVMgQ= )
b.example.net.          100     IN A    192.168.2.2
100     RRSIG   A 5 3 100 20070429180412 (
20070330180412 17000 example.net.
lWXfx2ebTpOBvCxk+mX882u1oiodeRkAS3ck
IbSni18DxEh3sTBjj9FaWEO3uiTYS5OK+ZnP
G00qlAljL1znWuPsv1X0/olIqSEYNgBJI98r
fDhfDXzVs/zCtFm1WesaNs+akohz9zOzITGh
U3j5DkLx1gk62SWrlDhcpCnt0qI= )
100     NSEC    ns.example.net. A RRSIG NSEC
100     RRSIG   NSEC 5 3 100 20070429180412 (
20070330180412 17000 example.net.
Pg4u+EI+HzeFFQ3EV8hzDlaDNx00ijGV5OMP
V3U0j1FE2BoMIn+7aBvCQ719p9aGIFE5iDU/
nMLQ0M0lA1dHD/aXhVCDp4gsM0hhqa2nuEUq
uUV6kRWzjS6czHg66+g0GilxBK2m6TCF7dQI
A7LyJZvUWSOVKrfTTpGWuPjZIMU= )
ns.example.net.         100     IN A    192.168.2.203
100     RRSIG   A 5 3 100 20070429180412 (
20070330180412 17000 example.net.
Bm3vcmaEC3kdcJuYljUSverccng5qK+fUI79
Eyfh+hkPRh6XgByQdheBQEQmIZ8DCFbF3QVC
R9257CSyVEL0JAcWldsfVFkMHb1Y/T00UtpE
3Y+hjgKmaNZPCEEOgWxhBEoutP1Sb+ielJOw
TL0+pb/cIyEkFsbBH5rGKcdCBOw= )
100     NSEC    example.net. A RRSIG NSEC
100     RRSIG   NSEC 5 3 100 20070429180412 (
20070330180412 17000 example.net.
wEsZezaBDr+LpSTIuz2zG4oskTnCM+e8PC36
uIQcq0koygY+cluYdSfEETWl2wiTmnkTpSm0
p78BtcfAIiyKsV6AwwSsw1mYexhH5ARYHtHv
yeWYuaKel+d/f4qa0ToUOOnF3aYum5Ss1joB
Zw+7POOeJ8tve6gaUvLlYCAW7Ko= )
```

Obrázek 9: Příklad podepsaného zónového souboru

2.5 KONFIGURACE CACHING FORWARDERU

Teď, když servery DNS publikují podepsaná data, potřebujeme nakonfigurovat ‚klienty', aby tato data ověřily. Klienty rozumíme v tomto kontextu rekurzivní jmenné servery. Jednoduše nakonfigurujte rekurzivní jmenný server pomocí veřejného klíče SEP vygenerovaného pro danou zónu. Popsáno je to v oddíle 1.3, a také viz obrázek 3 tamtéž.

2.6 ZNOVUPODEPSÁNÍ ZÓNY

Pokud jsou podpisy vaší zóny před vypršením nebo pokud jste přidali do vaší zóny nové záznamy, budete muset zónu znovu podepsat. Jsou dva způsoby, jak znovu podepsat data zóny. Můžete zvolit možnost závisející na úrovních automatizace, velikosti zóny a frekvenci, s jakou jsou RRSIG RR generovány.

- Z nepodepsaného zónového souboru můžete znovu generovat podepsanou zónu. Podepisující nástroj musí znovu uspořádat zónu, vygenerovat všechny záznamy NSEC a vygenerovat všechny záznamy RRSIG. Pokud generujete váš zónový soubor ze záložní databáze, je to metoda, které byste měli dát přednost.
- Můžete přidat nové záznamy do již existující podepsaného zónového souboru a zpracovat jej podepisujícím nástrojem. Podepisující nástroj BIND vloží nové záznamy a propojí NSEC v již uspořádaném zónovém souboru a podepíše pouze nové záznamy a záznamy, u kterých se blíží ke konci doba platnosti podpisů.

Měli byste si sestavit nástroje pro správu vašich podepsaných zón např. využít **cron**, **perl** a **make**. (viz také oddíl D v přílohách)

2.7 ODSTRAŇOVÁNÍ PROBLÉMŮ PODEPSANÝCH ZÓN

Můžete zkontrolovat formát vašeho **named.conf** použitím programu **named-checkconf**. Pro kontrolu zónových souborů můžete použít program **named-checkzone**. Tyto programy používají při syntetické analýze konfigurace stejné postupy a zónové soubory jako **named**, ale kontrolují pouze syntax.

Pro ověření svého nastavení můžete použít **„dig"** a jmenný server konfigurovaný pomocí důvěryhodného klíče. Pokud nemohou být data kryptograficky ověřena odešle je forwarder zpět se statutem **SERVFAIL**. Můžete to otestovat záměrně poškozeným zdrojovým záznamem v podepsaném zónovém souboru. Toto je typický výstup „dig" pokud se dotazuje na poškozená data:

```
; <<>> DiG 9.3.2 <<>> @192.168.2.204 corrupt.example.net A +dnssec +multiline +retry=1
; (1 server found)
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 50607
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
```

```
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;corrupt.example.net.   IN A

;; Query time: 40 msec
;; SERVER: 192.168.2.204#53(192.168.2.204)
;; WHEN: Fri Mar 30 21:04:32 2007
;; MSG SIZE  rcvd: 48
```

Nezapomeňte, že caching forwarder neprovádí kryptografické ověření zón, pro které je autoritativní. Proto, pokud je caching forwarder primárním nebo sekundárním serverem pro určitou zónu, obdržíte odpověď vždy, jelikož se předpokládá, že data z disku jsou bezpečná.
Další řešení problému je třeba předvést na serveru konfigurovaném jako ověřovací rekurzivní jmenný server. Níže je uveden příklad výstupu logu ověřujícího jmenného serveru, když jsme se dotazovali na poškozená data.

```
validating @0x1823e00: corrupt.example.net A: starting
validating @0x1823e00: corrupt.example.net A: attempting positive response validation
validating @0x182a400: example.net DNSKEY: starting
validating @0x182a400: example.net DNSKEY: attempting positive response validation
validating @0x182a400: example.net DNSKEY: verify rdataset: success
validating @0x182a400: example.net DNSKEY: signed by trusted key; marking as secure
validator @0x182a400: dns_validator_destroy
validating @0x1823e00: corrupt.example.net A: in fetch_callback_validator
validating @0x1823e00: corrupt.example.net A: keyset with trust 7
validating @0x1823e00: corrupt.example.net A: resuming validate
validating @0x1823e00: corrupt.example.net A: verify rdataset: RRSIG failed to verify
validating @0x1823e00: corrupt.example.net A: failed to verify rdataset
validating @0x1823e00: corrupt.example.net A: verify failure: RRSIG failed to verify
validating @0x1823e00: corrupt.example.net A: no valid signature found   validator @0x1823e00: dns_validator_destroy
```

(Tento výstup byl generován s konfigurací, kdy se do logovacího kanálu . zaznamenávají chyby kategorie dnssec od úrovně závažnosti 3)

2.8 MOŽNÉ PROBLÉMY

Sériové číslo SOA

Pokud zapomenete před opětovným podepsáním vaší zóny zvýšit sériové číslo, sekundární servery nemusí přijmout nové podpisy. To může způsobit časové odpojení některých autoritativních serverů, takže některé resolvery budou schopny váš podpis ověřit, zatímco jiné ne.

Rotace klíče podepisujícího klíče

Pokud administrátor zóny rozlišuje mezi klíči podepisujícími zóny a klíči podepisujícími klíče, pak nevyžaduje rotace klíče podepisujícího zóny žádnou akci ze strany administrátorů validátorů. Pokud se mění klíč podepisující klíče, měla by se věnovat pozornost tomu, aby byl všem resolverům v organizaci zaslán nový důvěryhodný klíč.

Pokud je zóna zabezpečena pouze lokálně (tj. není součástí řetězu důvěry), pak je rotace klíče podepisujícího klíče relativně snadná. Nezapomeňte, že k ověření dat musí existovat aspoň jeden podpis, který může být ověřen důvěryhodným klíčem v resolverech. Po omezenou dobu používáte k podepisování vaší zóny dva klíče podepisující klíče: starý a nový klíč. Během této doby začněte rekonfigurovat resolvery ve vaší organizaci pomocí nových důvěryhodných klíčů. Jakmile mají všechny resolvery nakonfigurovaný ve své instrukci důvěryhodného klíče nový klíč, měly by být zóny podepsány pouze novým klíčem.

Viz také Kapitola 4, „Rotace klíčů".

Problémy podřízených serverů

Pokud není jeden z autoritativních serverů zóny DNSSEC, musí se na podřízených serverech spustit kód, který povoluje DNSSEC. Problémy mohou nastat v případě, že se klient DNS snaží získat data z těchto serverů, které nemají DNSSEC.

Vzroste zatížení všech vašich jmenných serverů. Zvětší se zónové soubory a vzroste využití paměti a šířky pásma. Faktory 2 – 5 nejsou neobvyklé; Některé údaje jsou uvedeny v „Návodech a tipech".

3. DELEGOVÁNÍ PODEPISUJÍCÍ AUTORITY; PŘECHOD NA GLOBÁLNÍ ZABEZPEČENÍ

3.1 ÚVOD

Zabývali jsme se tím, jak nasadit DNSSEC v jednotlivé zóně. Nyní chceme sestavit řetěz důvěry, tedy jakmile klient bezpečně získá veřejný klíč, který je vysoko v hierarchii DNS, může sledovat řetěz, aby ověřil data ve vašich zónách nebo vašich vnořených zónách.

Při procesu ověření začne resolver od nakonfigurovaného trust anchor. Použije ten, který ověřuje sady klíčů v apexu zóny. Jakmile je jednou key-set ověřen, klíče v tomto key-setu mohou být použity k ověření jakýchkoliv jiných dat v zóně, jako **A**, **AAAA** a zdrojové záznamy **PTR**. Aby mohl důvěřovat vnořené zóně, bude validátor sledovat ukazatel uložený ve zdrojovém záznamu DS, který odkazuje na klíč v key-setu vnořené zóny, který bude použit k ověření klíčů v této zóně. Toto **DS RR** je podepsáno klíčem podepisujícím zóny nadřazené zóny a odkazuje na klíč podepisující klíče vnořené zóny (obrázek 4).

3.2 PRAKTICKÉ KROKY

Níže popíšeme, jak nastavit zónu, která je globálně zabezpečená na základě podpisu nadřazené zóny přes záznam DS odkazující na klíč podepisující klíče vnořené zóny.

V příkladu používáme **net** jako nadřazenou zónu a **example.net** jako vnořenou zónu. Na začátku procesu předpokládáme, že už je nadřazená zóna lokálně zabezpečená, ale nemá ještě bezpečnou delegaci. To znamená, že nadřazená zóna nemá DS RR pro example.net., a že resolvery sledující řetěz důvěry přes net. budou považovat zónu example.net. za prokazatelně nezabezpečenou. Předpokládá se, že zóna example.net. je bezpečná.

Většina postupu bude obdobná jako v kapitole 2, „Zabezpečení zóny DNS", ale protože používáme key-sety, liší se některé podrobnosti.

Naším cílem je publikovat nadřazenou zónu s DS RR. DS RR se vztahuje ke klíči podepisujícím klíče generovaným vnořenou zónou (DS RR obsahuje kryptografický hash z dat v DNSKEY RR). Proto musí vnořená zóna odeslat nadřazené zóně některé informace. Pro usnadnění procesu používá BIND key-sety a ds-sety.

key-set je malý soubor se stejnou syntaxí jako zónový soubor, který obsahuje jeden nebo více klíčů podepisující klíče vnořené zóny. ds-set je taktéž podobným souborem, ale tento soubor obsahuje DS RR, které musí být připojeno k nadřazené zóně. Tyto soubory jsou vytvořeny při podepisování zóny, jak bylo popsáno v oddíle 2.4. Například pro zónu example.net budou pojmenovány **dsset-example.net** a **keyset-example.net**.

Můžeme si představit mnoho způsobů, jak přiřadit key-set nadřazené zóně. Například

- vnořená zóna odesílá e-mail (kryptograficky podepsaný, aby byla zaručena integrita a umožněny autentizace) s ds-setem nebo key-setem.
- nadřazená zóna může získat odpovídající klíč z DNS vnořené zóny a sama vytvořit soubor **key-set**. To provede umístěním materiálu klíče do souboru pojmenovaného **keyset-child-domainname**.
- pro získání key-setu se používá systémové rozhraní založené na webové registraci.

V operačním prostředí je nesmírně důležité, aby byly stanoveny autenticita a integrita DNSKEY. Administrátor nadřazené zóny bude muset ověřit, že klíč byl zaslán administrátorem vnořené zóny. Pokud je to možné, mělo by to být potvrzeno nějakým mechanizmem mimo DNS. Nadřazená zóna může využít zákaznickou databázi pro ověření, zda byl klíč skutečně zaslán administrátorem zóny. Pokud je podepsán špatný klíč, bude vnořená zóna náchylná k útokům; podepsání špatného klíče narušuje DNSSEC.

Nadřazená zóna ukládá key-sety v adresáři, kde jsou uloženy zónové soubory, nebo pokud chcete udržovat čistotu souborového systému, pak v adresáři, který je určen příznakem **-d dnssec-signzone**. Nástroj pro podepisování zóny automaticky vygeneruje (nebo připojí) odpovídající

záznamy DS, pokud je zadán příznak -g a nalezen keyset-child-domainname (nebo ds-set). Přestože key-set generovaný vnořenou zónou obsahuje podpisy, není nutné, aby byly v souboru keyset-child-domain nadřazené zóny RRSIG RR, podepisující nástroj neprovede ověření podpisu.
Následuje příklad, jak vyvolat příkaz:

```
dnssec-signzone -r /dev/random -g -d /registry/tld-zone/child-keys/ \
-o tld -f tld.signed db.tld
```

Alternativní metodou připojení DS RR do jednotlivé zóny je zřetězení nebo připojení ds-setů k zónovému souboru.

```
cat /registry/tld-zone/child-dssets/dsset-* >> tld
dnssec-signzone -r /dev/random  -o tld -f tld.signed db.tld
```

Pokud nadřazená zóna podepisuje svou vlastní zónu a používá u **dnssec-signzone** příznak **-d** a své vlastní **ds-** a key-set, bude uložena v určeném adresáři, což může být zavádějící.

3.3 MOŽNÉ PROBLÉMY

Algoritmus veřejného klíče

Pro globální zabezpečení musíte používat minimálně jeden klíč s algoritmem, který je povinně implementován. Povinně jsou implementovány klíče RSA/SHA1 a DSA. Doporučujeme použití pouze klíčů RSA/SHA1.

Nadřazená zóna udává zabezpečení vnořené zóny

Je důležité, aby byl DNSKEY publikován v DNS předtím, než nadřazená zóna připojí pro tento klíč podepsané DS RR.
Pokud nadřazená zóna připojí DS RR, zatímco vnořená zóna ještě nepublikovala klíč, stane se vnořená zóna ‚špatnou'; Tím, že pro vnořenou zónu nemá DS RR, indikuje nadřazená zóna, že vnořená zóna není bezpečná.
Jako nadřazená zóna byste měli vždy ověřit, že vnořená zóna publikuje podepsaný DNSKEY před připojením DS RR.

3.4 PŘIHLÁŠENÍ DO REGISTRU DLV

Pokud vaše nadřazená zóna ještě není zabezpečena, můžete využít registru DLV, a tak mohou třetí strany nadále využívat zabezpečení, které poskytuje vaše doména (konfigurace na straně klienta viz oddíl 1.5)
dnssec-signzone v BINDu umožňuje vytvořit soubor, který obsahuje příslušná data pro registr DLV. Předpokládejte, že je váš oblíbený registr DLV ukotvený na dlv-registry.org, pak podepsání s volbou -l vytvoří soubor dlvset.
Například:

```
/usr/local/sbin/dnssec-signzone  \
-l dlv-registry.org \
-o example.net  \
-k Kexample.net.+005+49656 \
db.example.net \
Kexample.net.+005+17000.key
```

vytvoří soubor pojmenovaný dlvset-example.net., který obsahuje následující informace:

```
example.net.dlv-registry.org. IN DLV 49656 5 1 3850EFB913AEC66275BCA53221587D445702397E
```

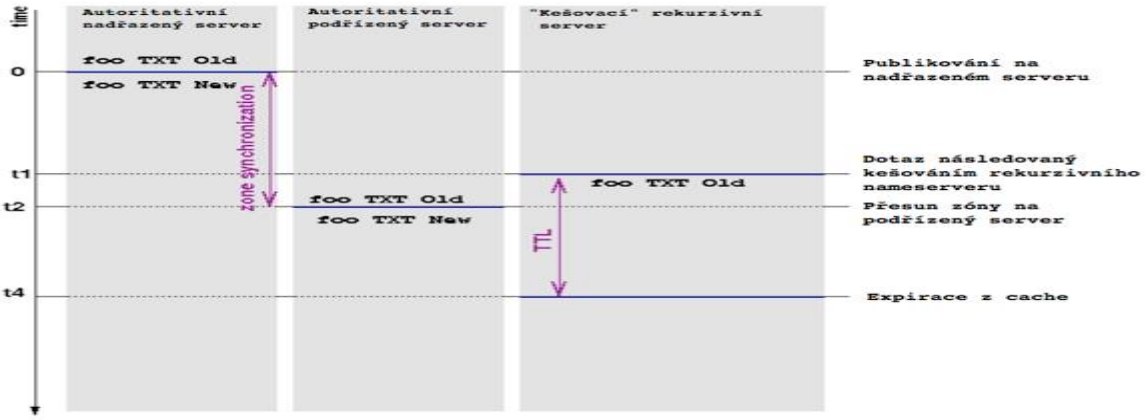
Autor doporučuje využití registrační služby lookaside společnosti ISC.

4. ROTACE KLÍČŮ

Rotace je proces, při kterém je jeden klíč v zóně nahrazen jiným. Jelikož mají klíče omezenou životnost, musí se občas měnit. Během rotace je třeba dbát na to, aby nebyly narušeny existující řetězy důvěry.
Rotace je definována okamžikem, kdy jsou do zóny zapsány klíče generované pomocí „nového" soukromého klíče. Pár klíčů může být generován s dostatečným předstihem a veřejný klíč může být rovněž publikován s dostatečným předstihem.
Pokud je rotace plánovaná, hovoříme o ní jako o plánované rotaci. Pokud je rotace zapříčiněna (předpokládaným) prolomením nebo ztrátou soukromého klíče, hovoříme o neplánované, nebo nouzové rotaci klíče.
Existují dva druhy plánovaných rotací klíčů. Rotace klíčů podepisujících klíče a rotace klíčů podepisujících zóny.
Přestože protokol DNSSEC nerozlišuje klíče podepisující zóny a klíče podepisující klíče, důrazně doporučujeme dodržovat toto rozlišení, jelikož zaručuje jasné oddělení klíčů, které mohou být rotovány bez vnějšího zásahu (klíče podepisující zóny) a klíče, které vyžadují vnější zásah (klíče podepisující klíče). Při vytváření klíčů podepisujících klíče pomocí dnssec-keygen byste měli používat příznak KSK -f tak, abyste vždy oddělovali klíče podepisující klíče a klíče podepisující zóny při pohledu na tzv. příznakové pole ve zdrojovém záznamu DNSKEY. Jeho příznakové pole se bude lišit (většinou 257), pokud pracujete s podepisováním klíčů, nebo klíčem SEP.

4.1 DNS TRAVERSAL (PŘECHOD PŘES DNS)

Kdykoliv jsou data v zónovém souboru nahrazena jinými daty, musí se předtím, než je klienti skutečně uvidí, propagovat pomocí DNS. V prostředí bez DNSSEC to může být stěží zaznamenáno, ale pokud využíváte DNSSEC, je umožnění přechodu dat přes DNS kritické.
Data DNS se svými propojenými podpisy a veřejný klíč, kterým jsou tato data ověřena, prochází přes DNS nezávisle. To znamená také to, že veřejné klíče a podpisy jsou do vyrovnávací paměti ukládány (cached) nezávisle a proto z ní odcházejí v různou dobu. Důsledkem může být, že je RRSIG ověřen pomocí DNSKEY z cache, a že RRSIG a DNYKEY pochází z různých verzí zóny tj., veřejný klíč se vztahuje ke klíči, který je starší než podpis. Dochází také k opačné situaci, kdy jsou podpisy starší než veřejné klíče používané pro ověření.
Jako administrátor zóny musíte mít toto chování na paměti a počítat s tím, že vaše podpisy mohou vyžadovat ověření pomocí jakékoliv budoucí nebo předcházející verze vašeho key-setu. [9] popisuje podrobnosti, které se liší u rotací klíčů podepisujících zóny a klíčů podepisujících klíče.
Existují dva přístupy. „Předem publikovaná" a „dvojitě podepsaná" rotace.
Nejprve se podrobněji podívejme, jak prochází data přes DNS. Viz obrázek 10 níže.
V t0 nahrazují nová data ta z předchozí verze zónového souboru. Data jsou publikována na autoritativním masteru (nebo primárním serveru). Nějakou dobu bude trvat (označujeme ji jako dobu synchronizace), než všechny autoritativní servery akceptují novou verzi zóny. Podle nejhoršího scénáře se neprojeví změna podřízeného serveru na master serveru a zóna vyprší. Takže maximální hodnota doby synchronizace bude hodnotou parametru vypršení SOA.
Předpokládejme, že v určitém čase (t1) mezi publikací nové zóny na master serveru (t0) a dobou, kdy je nová zóna akceptována podřízeným serverem (t2), je dotaz na data zpracován rekurzivním jmenným caching serverem. Tento rekurzivní server vrátí jakémukoliv klientovi v době stanovené hodnotou TTL ve starém RRsetu stará data. Rekurzivní server akceptuje nové záznamy až po čase t4, kdy se vrátí a bude se dotazovat na nová data.
Všimněte si, že t4 nezávisí pouze na t1+TTL, ale je omezen také dobou vypršení podpisu na starém RRsetu.



Obrázek 10: Přenášení DNS dat

4.2 „PŘEDEM PUBLIKOVANÉ" A „DVOJITĚ PODEPSANÉ" ROTACE

Během předem publikované rotace je veřejný klíč představen v DNSKEY RRsetu s dostatečným předstihem, než jsou pomocí soukromé části klíče vytvořeny RRSIG. „Nové" veřejné klíče jsou pak k dispozici v caches jakmile se RRSIG dat objeví na autoritativních jmenných serverech a jmenné caching servery mohou využívat k ověření nových dat DNSKEY RR uložené v cache.

Během dvojitě podepsané rotace je uveden nový pár klíčů a podpisy jsou generovány pro oba – starý i nový – klíče. Oba veřejné klíče jsou publikovány v DNS. Po uplynutí doby potřebné k propagaci těchto dat přes DNS je starý klíč odstraněn a k podpisu je publikován pouze nový klíč.

4.3 NÁSTROJE

Pro řádnou správu ‚statutu' budete potřebovat prováděcí poznámkový blok. Pro každou zónu bude existovat několik KSK a ZSK a všechny tyto klíče budou mít ‚statut'. Situace může být velmi zavádějící. Níže uvádíme přehled operací využívajících „prováděcí poznámkový blok". V RIPE NCC jsme vyvinuli nástroj, který nahrazuje „prováděcí poznámkový blok" a je napojený na podpisové operace. Nástroj je k dispozici na http://www.ripe.net/disi/dnssec_maint_tool/.

4.4 ROTACE ZSK

Během rotace klíče podepisujícího zóny (ZSK) používáme „předem publikované" schéma.

4.4.1 PŘÍPRAVA ZSK (PRODUKČNÍ FÁZE)

Jako příklad použijeme jednoduchou zónu **example.com** (obrázek 11). Zóna je uložená v **db.example.com**.

```
; example.net zone
;
$TTL 100
$ORIGIN example.net.
@               100     IN      SOA     ns.example.net. (
                        olaf.nlnetlabs.nl.
                        2002050501
                        100
                        200
                        604800
                        100
                        )

                NS              ns.example.net.
ns.example.net.         A               192.168.2.203

a               A               192.168.2.1
b               A               192.168.2.2

*               A               192.168.2.10
b.a             A               192.168.2.11
```

Obrázek 11: Jednoduchá example.com

Pokud předpokládáme, že budeme nejprve publikovat example.com, vygenerujeme dva klíče **ZSK** a jeden **KSK**.

```
dnssec-keygen -a RSASHA1 -b 1024 -n ZONE example.com
Kexample.com.+005+63935
dnssec-keygen -a RSASHA1 -b 1024 -n ZONE example.com
Kexample.com.+005+64700
dnssec-keygen -f KSK -a RSASHA1 -b 1024 -n ZONE example.com
Kexample.com.+005+54915
```

Do prováděcího poznámkového bloku zapíšeme, že klíč 63935 bude použit jako aktivní a klíč 64700 jako pasivní ZSK. Oba klíče budou dostupné v key-setu, ale k podpisování bude použitý jen aktivní klíč.

Po vygenerování klíčů jsme je připojili do zóny pomocí zapsání instrukcí do db.example.com

```
;; ZSKs
$include Kexample.com.+005+63935.key
$include Kexample.com.+005+64700.key

;; KSKs
$include Kexample.com.+005+54915.key
```

Pak podepište zónu. Jelikož nechceme použít implicitní nastavení dnssec-signzone (což znamená použití všech dostupných klíčů k podpisu), musíme v příkazové řádce jasně specifikovat, které klíče použít. Protože to budete muset dělat často, bude se hodit prováděcí poznámkový blok.

```
dnssec-signzone -k  Kexample.com.+005+54915.key -o example.com \
db.example.com Kexample.com.+005+63935
```

Všimněte si, že jsme jako argument použili KSK s přepínačem -k a podepisovacím klíčem je pouze aktivní klíč ZSK.

4.4.2 ROTACE ZSK (FÁZE 1)

Poznamenejte si dobu vypršení podpisu DNSKEY RR, kterou nyní naleznete v DNS. Tato hodnota může být použita jako horní hranice trvání této fáze. Je to hodnota t4 na obrázku 10. V níže uvedeném DNSKEY RR setu je doba vypršení podpisu 21. srpna 2004, okolo 11:35 UTC. Pokud není žádný 12 TTL ve vaší zóně vyšší než například 600, neměli byste čekat tak dlouho. Měli byste počkat 10 minut od okamžiku, kdy uvidíte, že nová zóna byla publikována na všech autoritativních serverech. Vypršení podpisu je horní hranice.

```
100     DNSKEY  256 3 5 (
AQPQyhg865V4zkFZN+FICLAZPWWaAf5I43pW
UcuOiejT92AVu0eHOkbH5YiHV97r+QjAdZ7K
W7W+bvbqKBR5P4QMVNm8zCs5Trb9OcOY0+bb
LYZG3aG69wUfF1pjvmFV5zUSRHCLMEzXb5NS
XdazgdhhuM07L2e2EfJGp5qijtRwpQ==
) ; key id = 63935

100     DNSKEY  256 3 5 (
AQPWrMsW0sGSTD7iE9ou+s7886WeSLIq/l/J
CgqwAn7jlECGAAN6cSHV5jWvovcWFthapWdG
DpC1uL48AcWtVWkRABGjU8Q16CAy0EcZ+24V
4cul+VluBt1YjuNfUlye+k5V+lmkjXBQ3Qdf
E8/owjsdx9mTkeQC4qiFjUxWXTl4DQ==
) ; key id = 64700

100     DNSKEY  257 3 5 (
AQPhZQ29Xg60NLgR+qdJENZpklU+WQF0abmp
Ni3CeOYyR+bd01Q/2WDI6BbWCLdIb9YflRaj
hmyb+AmzmjNzhw8VjcY9Sr2zIcG50ctuZ8Og
t7fcGrCbEM9fIDIKdDRlf+SY8OnGEMi6sI4m
bZ4zoh+nWfNrTxQR5hHv074uSAvZyQ==
) ; key id = 54915

100     RRSIG   DNSKEY 5 2 100 20040821114554 (
20040722114554 54915 example.com.
gcnf3rf+D6izv9A//16u+Jx/LDVinLtcpkWR
yxDV5goS2SnoLfyEryqbSAyKbh4redyQCjSW
/HZXFBoPYrAy8fqaY1AfjVP+q9zJPvysUOp+
2T6mm8/9pcZoGXw1wPjPUAz+AF0oJnoaWo7t
764xvZc47kAI1pT0RTizV2BofcU= )

100     RRSIG   DNSKEY 5 2 100 20040821114554 (
20040722114554 63935 example.com.
T7gRcEZkxEl5iGJdCzSu47Og9ydMO5Uggvcz
A9jETiTUrBttyYua7qDZOjNrzT4GVZ6s/UBw
tbGCqyMU/sVvaulP4h8oerX44bw5eP/mluLY
T9rwm2jBI1rZSPDdGDp8lJ2vvrxASYSF2Fxg
```

V okamžiku rotace musíte nastavit svůj současný pasivní klíč (64700) jako aktivní a svůj současný aktivní klíč (63935) jako pasivní. Také si poznamenejte, že tento klíč musí být z keysetu odstraněn v následující fázi rotace.
Zvyšte sériové číslo SOA a znovu podepište zónu pomocí nového aktivního klíče.

```
dnssec-signzone -k  Kexample.com.+005+54915.key -o \
example.com db.example.com \
Kexample.com.+005+64700
```

Publikujte tuto zónu v DNS a ujistěte se, že zůstane publikována dostatečně dlouho, aby byla propagována přes DNS.

4.4.3 VYČIŠTĚNÍ ZSK (FÁZE 2)

Poté, co byla data propagována přes DNS, musíte nahradit pasivní ZSK (63935) novým pasivním klíčem.
Začněte s generováním pasivního ZSK.

```
dnssec-keygen -a RSASHA1 -b 1024 -n ZONE example.com
Kexample.com.+005+01844
```

Zapište nový pasivní klíč (01844) do zónového souboru. Odstraňte starý pasivní klíč (63935) ze zónového souboru.

```
;; ZSKs
$include Kexample.com.+005+64700.key
$include Kexample.com.+005+01844.key

;; KSKs
$include Kexample.com.+005+54915.key
```

Zvyšte sériové číslo SOA a znovu podepište zónu pomocí stejného aktivního klíče, jako ve fázi 1.

```
dnssec-signzone -k  Kexample.com.+005+54915.key -o example.com db.example.com \
Kexample.com.+005+64700
```

Po publikaci vaší zóny jste zpět ve "produkční" fázi. Neměli byste přikročit k nové rotaci, dokud neměl současný DNSKEY RRset dostatek času, aby byl propagován v systému.
Nyní můžete vymazat klíč 63935. Doporučujeme přesunout tento pár klíčů do odděleného adresáře nebo provést zálohu.

4.4.4 ÚPRAVA DAT ZÓNY BĚHEM ROTACE

Kromě dat v key-setu můžete data zóny upravovat kdykoliv. Po dobu, kdy používáte k podpisu odpovídající aktivní ZSK.

4.5 ROTACE KLÍČE PODEPISUJÍCÍHO KLÍČE

Během rotace klíče podepisujícího klíče (KSK) používáme „dvojitě podepsané" schéma.

4.5.1 PŘÍPRAVA KSK (PRODUKČNÍ FÁZE)

Jako příklad znovu použijeme jednoduchou zónu example.com (Obrázek 11). Zóna je uložená v db.example.com. Obsahuje aktivní a pasivní ZSK (63935 a 64700) a KSK (54915). Zapsané instrukce jsou stejné jako v oddíle 4.4.1:

```
;; ZSKs
$include Kexample.com.+005+63935.key
$include Kexample.com.+005+64700.key
```

```
;; KSKs
$include Kexample.com.+005+54915.key
```

a příkaz k podepsání zóny je také stejný.

```
dnssec-signzone -k  Kexample.com.+005+54915.key -o example.com \
db.example.com Kexample.com.+005+63935
```

4.5.2 ROTACE KSK (FÁZE 1)

Rotaci začneme vygenerováním nového KSK

```
dnssec-keygen -f KSK -a RSASHA1 -b 1024 -n ZONE example.com
Kexample.com.+005+06456
```

Vložte nový KSK do zónového souboru:

```
;; ZSKs
$include Kexample.com.+005+63935.key
$include Kexample.com.+005+64700.key

;; KSKs
$include Kexample.com.+005+54915.key
$include Kexample.com.+005+06456.key
```

Podepište zónu oběma KSK a aktivním ZSK.

```
dnssec-signzone -k  Kexample.com.+005+54915.key  \
-k  Kexample.com.+005+06456.key -o example.com \
db.example.com \ Kexample.com.+005+64700
```

Právě jste zapsali nový klíč KSK.
Protože rotujete KSK, budete muset tento klíč nahrát do vaší nadřazené zóny, nebo jej budete muset konfigurovat ve vašich trust-anchors (viz oddíl 1.3).Veřejný klíč, který budete muset nahrát a konfigurovat, je ten nový s key-id 06456.
Pokud DS RR vaší nadřazené zóny odkazuje na váš starý klíč, bude nějakou dobu trvat, než DS RR zmizí z chache. Horní hranice parametru t4 je vypršení podpisu v DS RR odkazujícím na starý KSK (54915).
dnssec-signzone vytváří dva soubory, které vám během tohoto procesu pomohou. dsset-example.com. a keyset-example.com.. Soubor „dsset" obsahuje DS RR, které se vztahují k KSK ve vašem zónovém souboru, a soubor „keyset" obsahuje KSK publikované ve vašem zónovém souboru.
Pamatujte, že jelikož nahrazujete klíče pouze jedním z těchto vstupů (06456), bude je zapotřebí zaslat do vaší nadřízené zóny.

4.5.3 VYČIŠTĚNÍ KSK (FÁZE 2)

Pokud jste se ujistili, že jsou všechny trust anchors aktualizovány a DS RR nadřazené zóny prošlo DNS, můžete odstranit starý klíč ze sady instrukcí:

```
;; ZSKs
$include Kexample.com.+005+63935.key
$include Kexample.com.+005+64700.key

;; KSKs
$include Kexample.com.+005+06456.key
```

Podepište zónu novým KSK a aktivním ZSK.

```
dnssec-signzone  -k  Kexample.com.+005+06456.key \
-o example.com db.example.com \
Kexample.com.+005+64700
```

Od této chvíle jste opět v produkční fázi.

4.5.4 VÍCENÁSOBNÉ KSK

Tento algoritmus se používá také v případě, že používáte vícenásobný KSK. Jednotlivé kroky:

- vygenerujte a připojte nový KSK do zóny;
- podepište zónu všemi KSK; počkejte na propagaci;
- odstraňte jeden KSK a podepište všemi zbylými KSK.

II. ZABEZPEČENÍ KOMUNIKACE MEZI SERVERY

Tato část se zabývá otázkami bezpečnosti transakcí. Zaměřuje se na zabezpečení transakcí mezi autoritativními servery (TSIG) k zajištění autorizace a integrity pro zónové přenosy, stejný postup může být použit i při zabezpečování dynamických aktualizací.

5. ZABEZPEČENÍ ZÓNOVÝCH PŘENOSŮ

5.1 ÚVOD

Komunikace mezi hosty může být zabezpečena (autentizována a šifrována) schématem založeným na symetrickém šifrování. Sdílením klíče se mohou administrátoři dvou serverů ujistit, že k výměně dat DNS dochází mezi těmito dvěma bloky a že data nejsou během přenosu manipulována. Nejznámější mechanismus, který to umožňuje, označujeme jako **TSIG** a je založen na sdílení tajného klíče. Sdílený tajný klíč se používá k podpisu obsahu každého paketu DNS. Podpis může být použit k ověření autenticity i integrity dat. Aby nedocházelo k podvodnému opětovnému zaslání zachycených dat (**replay attack**) je k datům přiřazeno časové razítko (**time stamp**). Mechanismus TSIG můžeme použít také, pokud chceme zabránit neautorizovaným zónovým přenosům; zónový přenos mohou provést pouze vlastníci tajného klíče. Popíšeme si, jak nakonfigurovat primární server **ns.foo.example** a sekundární server**ns.example.com** tak, aby při zónových přenosech používaly TSIG.
Pro konfiguraci TSIG proveďte následující kroky:

- Synchronizujte hodiny.
- Vytvořte a distribuujte sdílený tajný klíč, klíč TSIG.
- Na primárním serveru vytvořte přístupový seznam určující, kterým klíčům je umožněn přenos.
- Na sekundárním serveru určete, které klíče používat při kontaktu s primárními servery.

První položka je pro DNSSEC nezbytnou podmínkou. Pokud používáte DNSSEC, měli byste být v synchronizaci s okolním světem: Používejte **NTP**. Časové zóny mohou být zavádějící. Používejte **date -u** pro ověření, zda vaše zařízení používá správný čas **UTC**.
Konfigurace TSIG je úkolem systémových administrátorů.

5.2 GENEROVÁNÍ KLÍČE TSIG

Existuje několik způsobů, jak vytvořit sdílený tajný klíč.

5.2.1 GENEROVÁNÍ TAJNÉHO KLÍČE TSIG POMOCÍ DNSSEC-KEYGEN

dnssec-keygen je nástroj používaný k vygenerování šifrovaného náhodného čísla base64, které bude použito jako tajný klíč. Argumenty, které musíme do dnssec-keygen zadat pro vytvoření klíče TSIG jsou (viz také obrázek 6):

```
dnssec-signzone -a hmac-md5 -b 256 -n HOST ns.foo.example.ns.example.com.
```

Příkaz vytvoří dva soubory. Jména souborů obsahují příslušné informace:

```
Kdomain_name+algorithm_id+key_id.extension
```

domain_name je jméno určené jako jméno klíče. Jméno, které je zde určeno, nemusí být jménem, kterým se můžete dotazovat na DNS, ale mělo by být jménem, které může být šifrováno jako jméno domény. Obvykle se používá zřetězení jmen DNS dvou serverů.
Pro generování opravdu náhodného tajného klíče můžete použít dnssec-keygen, nebo použít passphrase – obě metody popíšeme v oddílu 5.2. V tomto případě **ns.foo.example.com** a **ns.example.com. algorithm_id** identifikuje použitý algoritmus: 5 pro HMAC-MD5 (1 a 3 jsou pro RSA a DSA, viz příloha A.1. v 5). key_id je identifikátor materiálu klíče, nevztahuje se k symetrickým klíčům. Rozšířením je **key** nebo **private**; první je veřejný klíč a druhý je soukromý klíč.
Formát těchto souborů se trochu liší, ale obsahují zcela stejné informace; šifrované náhodné číslo base64, které bude použito jako sdílený tajný klíč. Nenechte se zmást rozšířeními private a key, oba soubory musí zůstat zabezpečeny. Protože jsou tajné informace zkopírovány do konfiguračních souborů a tyto soubory nejsou využity v produkčním chodu, lze doporučit jejich vymazání.
Všimněte si, že **-n HOST** a jméno nejsou pro generování šifrovaného náhodného čísla base64 použity. Obvykle se používá jedinečná jmenovka domény, jež slouží k identifikaci klíče jako jména.

```
# dnssec-keygen -r /dev/random -a  HMAC-MD5 -b 128  -n HOST \
ns.foo.example.ns.example.com.
Kns.foo.example.ns.example.com.+157+12274

# cat Kns.foo.example.ns.example.com.+157+12274.key
ns.foo.example.ns.example.com. IN DNSKEY 512 3 157 gQOqMJA/LGHwJa8vtD7u6w==

# cat Kns.foo.example.ns.example.com.+157+12274.private
Private-key-format: v1.2
Algorithm: 157 (HMAC_MD5)
Key: gQOqMJA/LGHwJa8vtD7u6w==
```

Náhodné šifrované číslo **base64** je položkou, kterou potřebujete extrahovat z těchto souborů (tj. gQOqMJA/LGHwJa8vtD7u6w==), které určují tajný klíč v instrukci klíče:

```
key  ns.foo.example.ns.example.com.{
algorithm hmac-md5;
secret "gQOqMJA/LGHwJa8vtD7u6w==";
};
```

Tato definice klíče by měla být uložena v konfiguračních souborech primárního i sekundárního serveru a měla by být zcela totožná na obou stranách (v tomto příkladu je na obou jmenných serverech použit ns.foo.example.ns.example.com.). Doporučujeme generovat tajný klíč pro každou jinou stranu, se kterou přicházíte do styku, a bude zapotřebí spravovat tolik tajných klíčů, kolik je zón, pro které máte sekundární servery.

5.2.2 DALŠÍ ZPŮSOBY GENEROVÁNÍ TAJNÝCH KLÍČŮ

Příkaz **dnssec-keygen** vytváří opravdu náhodnou bit sekvenci. Předat tajný klíč vašemu kolegovi, který provozuje sekundární server na druhé straně světa, může být složité. V takových případech pravděpodobně dáte přednost pass-phrase, který může být komunikován po telefonu. Můžete použít jakýkoliv base64 šifrovací nástroj pro konverzi pass-phrase na platné vlákno v key-definition.

```
# echo "Crypto Rules" | mmencode
Q3J5cHRvIFJ1bGVzCg==
```

Pokud nemáte k dispozici mmencode, pomůže vám třeba tento skript perlu.

```
#!/usr/bin/perl
use MIME::Base64;
print encode_base64("@ARGV") ;
```

Ve skutečnosti postačí jakýkoliv řetězec, který může být šifrován base64, například **„ThisIsAValidBase64String“** může být také použit jako tajný klíč.

5.3 KONFIGURACE KLÍČŮ TSIG

Pro zabezpečení zónového přenosu musí administrátoři primárního a sekundárního serveru konfigurovat klíč TSIG v **named.conf.** Klíč TSIG je tvořen tajným klíčem a hash algoritmem, které jsou identifikovány jmény domén. Doporučujeme spravovat seznam tajných klíčů ve zvláštním souboru, který může přečíst pouze root a přidat jej do souboru **named.conf** (např. zadáním **/var/named/shared.keys**).
Instrukce klíče vypadá takto:

```
key ns.foo.example.ns.example.com. {
algorithm hmac-md5
secret "gQOqMJA/LGHwJa8vtD7u6w==";
};
```

Instrukce musí být zcela totožná pro obě zúčastněné strany.

5.4 KONFIGURACE TSIG PRIMÁRNÍCH SERVERŮ

Primární i sekundární server by měly sdílet tajný klíč konfigurovaný pomocí instrukce klíče v souboru **named.conf** (viz výše).
Primární server nyní může používat klíč uložený v seznamu, jenž je v BIND pojmenován **address_match_list**. Tyto seznamy se objeví v instrukcích **allow-notify**, **allow-query**, **allow-transfer** a **allow-recursion**, které řídí přístup k serveru. (Viz také oddíl 6.1.1 a 6.2.14.3 on-line dokumentace BIND).
V tomto bodě je důležitá v instrukci zóny **„allow-transfer“**. Pokud používá primární server pro **foo.example** výše generovaný klíč, bude v **named.conf**následující instrukce:

```
zone "foo.example"  {
type master;
file db.foo.example.signed;
\\ allow transfer only from secondary server that has
\\ key ns.foo.example.ns.example.com.
allow-transfer { key ns.foo.example.ns.example.com.  ; };
notify yes;
};
```

5.5 KONFIGURACE TSIG SEKUNDÁRNÍCH SERVERŮ

Primární i sekundární server by měly sdílet tajný klíč konfigurovaný pomocí instrukce klíče v named.conf (viz výše). Definice serveru v named.conf je použita k předání instrukce, která nařizuje jmennému serveru použít při kontaktu s jiným jmenným serverem určený klíč.

```
\\ secondary for foo.example.
\\ primary server ns.foo.example is on 10.1.1.2
server 10.1.1.2 {
keys { ns.foo.example.ns.example.com.;};
};
```

5.6 ZABEZPEČENÍ KOMUNIKAČNÍHO NÁSTROJE PRO UPOZORNĚNÍ

Výše uvedené nastavení poskytne podpisy pro zónový přenos z primárního na sekundární server. Jelikož je relace spouštěna sekundárním serverem 15, nastavuje zabezpečenou linku sekundární server. Proto má sekundární server v **named.conf** definici serveru. Obdobně můžete zabezpečit přenos na sekundární server, který byl spuštěn primárním serverem. Nezapomeňte zaslat sekundárnímu serveru zprávy **UPOZORNĚNÍ** v případě, že se změní obsah zóny. Tento přenos bude podepsán TSIG, jakmile přidáte server s IP adresou sekundárního serveru do named.conf primárního serveru. Můžete použít stejný klíč jako pro zónový přenos.
Jakmile primární server nakonfiguroval svůj server tak, aby k podepisování zpráv UPOZORNĚNÍ používal TSIG, může sekundární server použít klíč z řídícího přístupového seznamu **„allow-notify".**

5.7 ŘEŠENÍ PROBLÉMŮ S KONFIGURACÍ TSIG

Formát vašeho **named.conf** můžete zkontrolovat použitím programu named-checkconf. Tento program načte konfigurační soubor stejným způsobem jako**named**.
Při odstraňování problémů s konfigurací je vám k dispozici soubor **log** a „**dig**".
Před přidáním *allow-transfer {key ns.foo.example.ns.example.com. ;};* by mělo být možné přenést doménu z jakéhokoli počítače. *dig @ns.foo.example foo.example AXFR* by měl být úspěšný. Po nakonfigurování klíče by měl být stejný příkaz neúspěšný a vytvořit výstup podobný tomuto:

```
; <<>> DiG 9.2.0rc1 <<>> @ns.foo.example foo.example AXFR
;; global options printcmd
; Transfer failed.
```

Správnost konfigurace klíče můžete testovat dvěma způsoby.
Metoda 1: Požádejte administrátora zóny, aby zvýšil sériové číslo SOA a na primárním serveru zónu znovu načetl. Sekundární server by měl převzít změny. Soubor s logem sekundárního serveru bude obsahovat vstupy podobné těmto:

```
... general: info: zone foo.example/IN: transfered serial 2001082801
... xfer-in: info: transfer of 'foo.example/IN' from 10.1.1.2\#53: end of transfer
```

Metoda 2: Použijte „dig" k testování klíče pomocí příznaku –f.

```
dig @ns.foo.example  -k Kns.foo.example.ns.example.com.+157+12274.key \
foo.example AXFR
```

Obdobně můžete využít přepínač -y a určit key-name a tajný klíč přepínačem –y.

```
dig @ns.foo.example  \
-y ns.foo.example.ns.example.com.:gQOqMJA/LGHwJa8vtD7u6w== \
foo.example AXFR
```

Pokud se klíč neshoduje s logovacím souborem primárního serveru, proti kterému jste to zkoušeli, uvidíte záznamy podobné těm následujícím:

```
... security: error: client 10.1.1.6#1379: zone transfer 'foo.example.com/IN' denied
```

5.8 MOŽNÉ PROBLÉMY

Problémy s časováním

Počítače, které se účastní podepsané transakce TSIG, musí mít synchronizovaný čas v rozmezí několika minut. K synchronizaci počítačů použijte ‚**NTP**' a ujistěte se, že jsou časové zóny správně nakonfigurovány. Chybná konfigurace time-zone může vést k těžko odhalitelným problémům; pro ověření nastaveného času ‚UTC' použijte **date -u**.

Vícenásobné příkazy serveru

TSIG je ochranný mechanismus komunikace, který pracuje na principu ochrany jednotlivých počítačů. Použití vícenásobných příkazů serveru pro stejný server, nebo vícenásobných klíčů v jednom příkazu serveru povede k neočekávaným výsledkům.

III. NÁSTROJE PRO ODSTRAŇOVÁNÍ PROBLÉMŮ

Tato část popisuje několik praktických nástrojů pro odstraňování problémů, pomocí kterých lze nalézt příčiny případných chyb.

6. VYUŽITÍ „DRILL" PŘI ODSTRAŇOVÁNÍ PROBLÉMŮ

Při odstraňování problémů s nastaveními DNSSEC můžete použít „dig" i „drill" z distribuce BIND. Více informací o „dig" naleznete v oddíle 7.
Nejprve se budeme zabývat nástrojem „drill".

```
bin/drill version 1.1.0 (ldns version 1.1.1 )
Written by NLnet Labs.

Copyright (c) 2004-2006 NLnet Labs.
Licensed under the revised BSD license.
There is NO warranty; not even for MERCHANTABILITY or FITNESS
FOR A PARTICULAR PURPOSE.

Usage: bin/drill name [@server] [type] [class]
<name>  can be a domain name or an IP address (-x lookups)
<type>  defaults to A
<class> defaults to IN
```

argumenty mohou být seřazeny v libovolném pořadí

```
Options:
-D          enable DNSSEC (DO bit)
-T          trace from the root down to <name>
-S          chase signature(s) from <name> to a know key [*]
-V          verbose mode (once shows question, twice for hexdumps)
-Q          quiet mode (overrules -V)
```

```
-f file          file read packet from file and send it
-i file          file read packet from file and print it
-w file          file write answer packet to file
-q file          file write query packet to file
-h file        show this help
-v file        show version

Query options:
-4             stay on ip4
-6             stay on ip6
-a             only query the first nameserver (default is to try all)
-b <bufsize>      use <bufsize> as the buffer size (defaults to 512 b)
-c <file>             use file for rescursive nameserver configuration (/etc/resolv.conf)
-k <file>     specify a file that contains a trusted DNSSEC key [**]
used to verify any signatures in the current answer
-o <mnemonic>       set flags to: [QR|qr][AA|aa][TC|tc][RD|rd][CD|cd][RA|ra][AD|ad]
lowercase: unset bit, uppercase: set bit
-p <port>     use <port> as remote port number
-s             show the DS RR for each key in a packet
-u             send the query with udp (the default)
-x             do a reverse lookup
when doing a secure trace:
-r <file>           use file as root servers hint file
-t             send the query with tcp (connected)
-d <domain>           use domain as the start point for the trace
-y <name:key[:algo]>     specify named base64 tsig key, and optional
an algorithm (defaults to hmac-md5.sig-alg.reg.int)
-z                     don't randomize the nameservers before use

[*] = enables/implies DNSSEC
[**] = can be given more than once

vldns-team@nlnetlabs.nl | http://www.nlnetlabs.nl/ldns/
```

Obrázek 12: Argumenty „drill"

„drill" je součástí knihovny ldns dostupné na http://www.nlnetlabs.nl/ldns/. Instrukce pro instalaci naleznete rovněž na této stránce. (Je to jednoduché:*./configure ; make ; make install*).

Při odstraňování problému s nastaveními DNSSEC nám pomohou obzvláště přepínače **-T** a **-S** nástroje „drill". Při použití „drill" s –T sleduje řetěz důvěry od rootu až k leaves a ukazuje status zabezpečení (viz obrázek 13). S příznakem -S bude „drill" prohledávat podpisy od leave-node zpět k rootu a hledat příslušné záznamy (viz obrázek 14). Pokud používáte příznak -T nebo -S, budete muset určit soubor, který obsahuje trust anchor ve formátu RR tj., právě takový jako v souborech generovaných pomocí dnssec-keygen.

```
;; Domain: .
[T] . 100 IN DNSKEY 256 3 RSASHA1 ;{id = 63380 (zsk), size = 1024b}
. 100 IN DNSKEY 257 3 RSASHA1 ;{id = 63276 (ksk), size = 1280b}
[T] net. 100 IN DS 13467 RSASHA1 1 de01426e08ddb9186502ccc1081390cd7da0e178

;; Domain: net.
[T] net. 100 IN DNSKEY 256 3 RSASHA1 ;{id = 62972 (zsk), size = 1024b}
net. 100 IN DNSKEY 257 3 RSASHA1 ;{id = 13467 (ksk), size = 1280b}
[T] example.net. 100 IN DS 49656 RSASHA1 1 3850efb913aec66275bca53221587d445702397e

;; Domain: example.net.
[T] example.net. 100 IN DNSKEY 256 3 RSASHA1 ;{id = 17000 (zsk), size = 1024b}
example.net. 100 IN DNSKEY 257 3 RSASHA1 ;{id = 49656 (ksk), size = 1280b}
[T] example.net.      100     IN      SOA     ns.example.net. olaf.nlnetlabs.nl. 2002050501 100 200 604800 100

;;[S] self sig OK; [B] bogus; [T] trusted
```

Obrázek 13: Výstup „drill" -T -k< root.ksk > example.net SOA

```
;; ->>HEADER<<- opcode: QUERY, rcode: NOERROR, id: 49576
;; flags: qr rd cd ra ; QUERY: 1, ANSWER: 2, AUTHORITY: 2, ADDITIONAL: 0
;; QUESTION SECTION:
;; example.net. IN      SOA

;; ANSWER SECTION:
example.net.    100     IN      SOA     ns.example.net. olaf.nlnetlabs.nl. 2002050501 100 200 604800 100
example.net.    100     IN      RRSIG   SOA RSASHA1 2 100 20070429180451 20070330180451 17000 example.net.
T3R/X6o1mFjx+xGv9Nyn0LZJrF8S+bGjDNHcmAhJyvj+YLIELcCp8JdEjeALN8PF/p76KjuEDz/8VTz//fOx
/WAG5aSynDuT43ergR9TDgt3K+VK43LKtCVWS1SFfUPNLtesFE/6zYftGfDqJKX9USfiLGXPyBhPBOUJNPKyL3Q= ;{id = 17000}

;; AUTHORITY SECTION:
example.net.    100     IN      NS      ns.example.net.
example.net.    100     IN      RRSIG   NS RSASHA1 2 100 20070429180451 20070330180451 17000 example.net.
QCMj/ixUqmo6dmLGP3jITkIal9BF9i8xmRoGz
mY2JJWhB5TtpTyGXZcdPsBSFDk58zGRtlJmPRVTaWfxNtj3pQRJq1BO9oTTlh41RGIhhEyY
52kGGW4lpD+MBIuqUgfguTEzT3sJ4BcAfodDS12OMchOu62EvuQRetMb3FFXPA= ;{id = 17000}

;; ADDITIONAL SECTION:

;; Query time: 17 msec
;; EDNS: version 0; flags: do ; udp: 4096
;; SERVER: 192.168.2.204
;; WHEN: Fri Mar 30 21:04:55 2007
;; MSG SIZE  rcvd: 452
;; Chasing: example.net. SOA
```

```
;; Data set: example.net.       IN      SOA
        ;; Signed by: example.net.
    ;; Chasing: example.net. DNSKEY
;; Data set: example.net.       IN      DNSKEY
        ;; Signed by: example.net.
    ;; Chasing: example.net. DS
 ;; Data set: example.net.       IN      DS
        ;; Signed by: net.
    ;; Chasing: net. DNSKEY
 ;; Data set: net.       IN      DNSKEY
        ;; Signed by: net.
    ;; Chasing: net. DS
 ;; Data set: net.       IN      DS
        ;; Signed by: .
    ;; Chasing: . DNSKEY
 ;; Data set: .  IN      DNSKEY
        ;; Signed by: .
;; Key is trusted   ;; Chase successful
```

Obrázek 14: Výstup „drill" -S -k< root.ksk > example.net SOA

Knihovna ldns není spojena pouze s nástrojem „drill". V jejím adresáři s příklady naleznete několik užitečných nástrojů. Mimo jiné tam jsou:

ldns-key2ds - Vytvoří záznam DS ze záznamu DNSKEY

ldns-keyfecther - Vyhledá pro zóny veřejné klíče DNSSEC

ldns-keygen - Vygeneruje pár soukromý/veřejný klíč pro DNSSEC.

ldns-signzone - Podepíše zónový soubor podle DNNSECbis.

ldns-walk - ‚Projde' zónu DNSSEC.

7. VYUŽITÍ „DIG" PŘI ODSTRAŇOVÁNÍ PROBLÉMŮ

Nástroj „dig" má několik přepínačů užitečných při odstraňování problémů s nastaveními DNSSEC.

+multiline uspořádá výstup „dig" tak, že je snadno čitelný. Navíc bude jako komentář za DNSKEY RR vytištěno key_id.

+cd u dotazu nastavuje bit „ověření zakázáno". Obvykle to využijete v případě, že váš ověřovací rekurzivní jmenný server hlásí SERVFAIL a vy potřebujete zjistit, zda je příčinou to, že DNSSEC označí tato data jako „špatná".

+dnssec přinutí dotazovaný server, aby připojil příslušná data DNSSEC. Použití v kombinaci s +cd zjišťuje, zda jsou data zóny vůbec podepsaná, nebo pokud chcete určit, zda jsou intervaly platnosti na podpisech správné.

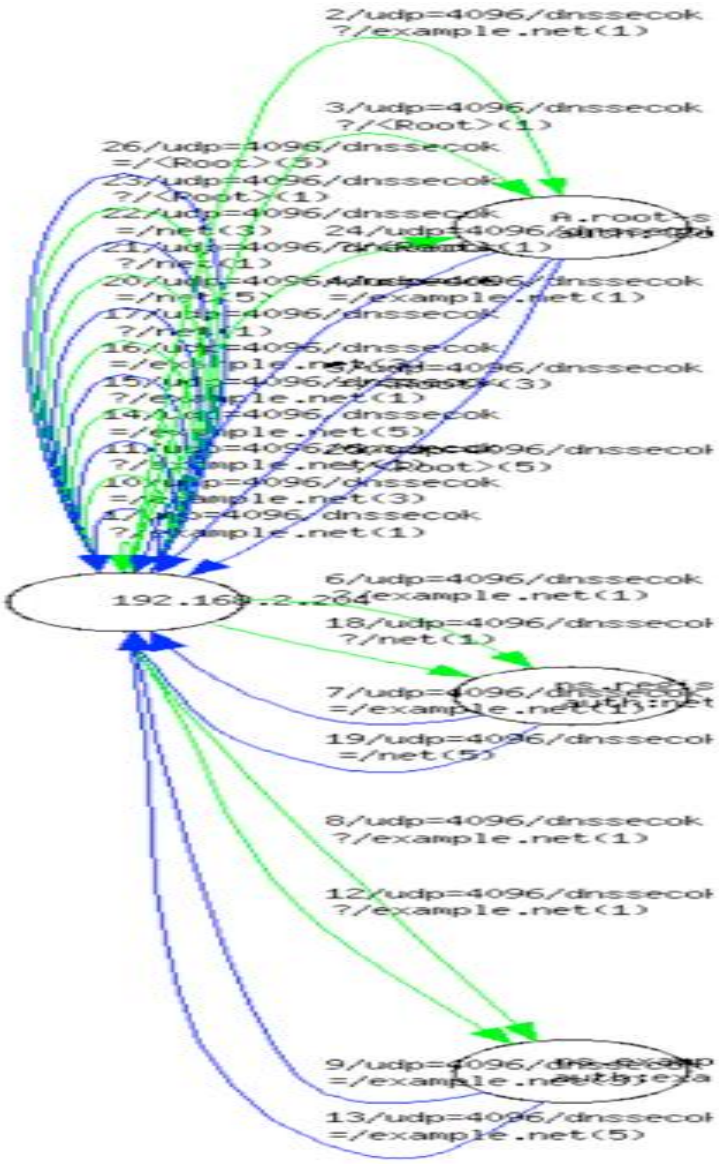
+trace stopuje řetěz delegování. Tato volba může být užitečná, pokud se snažíte zjistit, kde se nachází delegační body.

+sigchase stopuje řetěz podpisů. Budete rovněž potřebovat **./trusted-keys.keys** or **/etc/trusted-keys.keys**, které obsahují vstupy důvěryhodného klíče.

Při odstraňování problémů s nastaveními DNSSEC můžete použít „dig" i „drill" z distribuce BIND. Více informací o „drill" naleznete v oddíle 6.

8. NÁSTROJE DNSSEC

Množství open-source nástrojů DNSSEC naleznete na www.dnssec-tools.org. Stránka obsahuje několik běžných nástrojů údržby pro správu zón a klíčů, některé aplikace DNSSEC (Mozillu a plug-iny odbourávající spam) a několik nástrojů pro odstraňování problémů. Jeden z těchto nástrojů obsahuje nástroj **dnspktflow**, který vizualizuje ‚proudy' DNS v grafu. Tento nástroj napomáhá v kombinaci s výše popsanými nástroji vytvořit náhled na to, co se děje. Na obrázku 15 jsou kupříkladu zobrazeny pakety odeslané a přijaté v rámci trace na obrázku



Obrázek 15: Příklad výstupu dnspktflow

A INSTALACE BIND

Autor má povědomí o dvou open-source referenčních implementacích DNSSEC pro autoritativní servery: BIND a NSD. BIND je v současnosti jediným open-source rekurzivním jmenným serverem, který zvládá ověření DNSSEC.

DNSSEC je k dispozici od BIND 9.3.0. Nejnovější verzi BIND naleznete na ftp serveru společnosti ISC. Podpora DNSSEC je nekompilována, pokud je během kompilace nakonfigurována knihovna openssl.

konfigurujte s příznakem **--with-openssl**.

Pokud chcete zakompilovat oprávnění (viz oddíl 7) **„sigchase"** do **„dig"** budete muset nastavit proměnnou **STF_CDEFINES** na **-DDIG_SIGCHASE=1**

Zkontrolujte výstup config, abyste se ujistili, že bylo nalezeno openssl. Například:

```
cd /usr/local/src
tar -xzf bind-9.3.2.tar.gz
cd bind-9.3.2
./configure --prefix=/usr/local --with-openssl=/sw/ STD_CDEFINES="-DDIG_SIGCHASE=1"

...
Checking whether byte ordering is bigendian... yes
checking for OpenSSL library... using openssl from /sw//lib and /sw//include
checking whether linking with OpenSSL works... yes
...
```

Nezapomeňte prosím, že BIND 9.3.0 nemá implicitně povolen DNSSEC. Proto musíte v oddílu možností named.conf použít příkaz dnssec-enable.

```
options {
// turn on dnssec awareness
dnssec-enable yes;

};
```

B ODHAD ZVĚTŠENÍ ZÓNY

Při plánování podepisování zón musíte brát v úvahu, že podepisování zón zvětší váš zónový soubor a zabere větší množství paměti v autoritativních jmenných serverech. Provedli jsme několik měření, kdy jsme vyjmuli řadu zónových souborů, podepsali je a nahráli na jmenný server.

Začali jsme s 1,8 tisícem zón, které **RIPE NCC** poskytuje na svých autoritativních serverech. Pro řadu těchto zón je RIPE NCC primárním serverem, ale pro větší část těchto zón je RIPE NCC sekundárním serverem. Zóny mohou být rozděleny zhruba do dvou tříd; zóny **„end-**

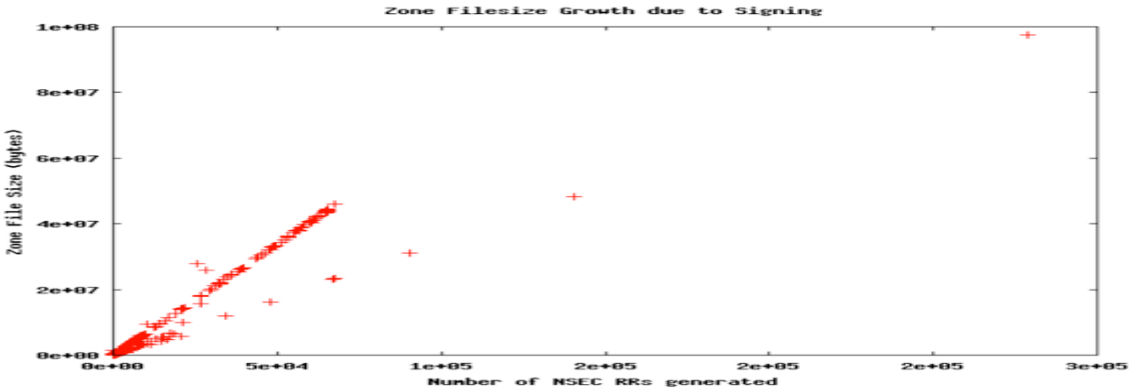
node" a **„delegační" zóny**. V zónách end-node jsou data pro většinu jmen v zóně autoritativní (většina jmen obsahujících např. **A**, **AAAA** nebo **PTR**). Delegační zóny obsahují většinou záznamy (NS) delegování, jedná se obvykle o vrcholové domény a „ reverzní delegační" domény.

Podepsali jsme zónové soubory 1024 bit **RSASHA1** klíčem podepisujícím zóny. Během podepisování byly připojeny **NSEC RR** spolu s odpovídajícími**RRSIG** a podepsány všechny RRsety v zóně. Jelikož delegování NS RR není autoritativní součástí dat, nebyl vytvořen žádný podpis. Obvykle jsou pro zónu **„end-node"** vytvářeny jeden NSEC a dva RRSIG, zatímco pro zónu delegačního typu je vytvářen pouze jeden od každého typu bezpečnostního záznamu. Na obrázku 16 jsme zakreslili zvětšení zónového souboru jako funkci počtu záznamů NSEC. Počet záznamů **NSEC** koreluje s počtem doménových jmen v zóně. Na obrázku můžete jasně vidět bimodální rozdělení. Jedno pro „end-node" typ zón a jedno pro delegační typ zón.

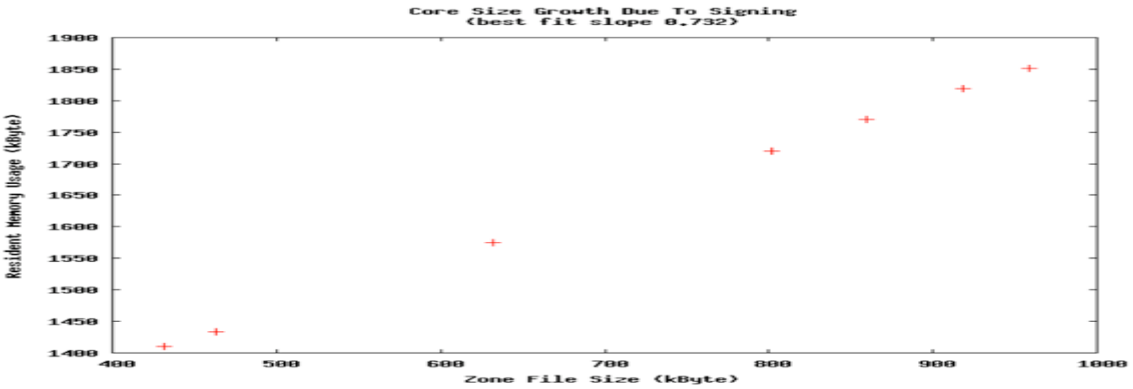
Na těchto datech jsme vyzkoušeli dvě přímé závislosti a zjistili, že u zóny delegačního typu se zvětší velikost o 350 bytů na jméno vlastníka, zatímco u zóny „end-node" se zvětší o 672 bytů na jméno vlastníka.

Na obrázku 17 níže jsme zakreslili poměr mezi zvětšením jádra ve vztahu k zvětšení zónového souboru v důsledku podpisu. Poměr je lineární s koeficientem přibližně 0,73. Zvětšení jádra představuje u zóny delegačního typu asi 200 bytů a u zóny typu „end-node" asi 500 bytů.

Tyto parametry můžete použít pro výpočet přibližné velikosti. Výsledky se mohou lišit v závislosti na velikosti a algoritmu klíče, který používáte, verzi BIND19 a obsahu zóny.



Obrázek 16: Velikost zóny vs počet jmen vlastníků



Obrázek 17: Velikost jádra vs velikost zónového souboru

C GENEROVÁNÍ NÁHODNÝCH ČÍSEL

Generování klíčů a v případě algoritmů DSA generování podpisů vyžaduje náhodná čísla. Měli byste dohlédnout na to, že váš generátor náhodných čísel vytváří „opravdu" náhodná čísla. Kvalita softwarově generovaných náhodných čísel je diskutabilní. To se týká také metod **/dev/random**. Ty získávají „náhodnost" z časů odezvy hardwaru. Měli byste se ujistit, že váš operační systém vytváří proud kvalitních náhodných čísel. U počítače, který nemá žádné vnější zdroje „náhodnosti", to může být komplikované a může to zapříčinit zablokování vašeho generátoru, nebo podepisovacího nástroje při čekání na entropii („náhodnost").

Jedním z relativně snadných nástrojů pro testování „náhodnosti" datových toků je ent od Fourmilab. Nebo můžete použít nástroj institutu NIST. „Dobré" výsledky měření z entu nebo nástrojů NIST by neměly být považovány za důkaz toho, že je váš generátor náhodných čísel dokonalý. Mohou existovat systémové vlivy, které se pomocí tohoto konkrétního nástroje 20 těžko objevují.

Relativně levným zdrojem náhodných dat jsou USB krypto žetony. Více informací o těchto žetonech a generování náhodných čísel naleznete na webové stránce Openfortress.

D KNIHOVNA NET::DNS::SEC V PERLU

Pokud hledáte nástroj pro správu vašich zón DNSSEC, možná vás bude zajímat knihovna Net::DNS::SEC dostupná na CPAN. S použitím tohoto rozšíření knihovny Net::DNS je velice snadné psát skripty, jako je ten níže uvedený, který ověřuje, zda podpisy přes SOA nevyprší během následujících 24 hodin.

```
#!/usr/local/bin/perl -T -Wall
#

# checkexpire.pl

# Příkladový skript, který se dotazuje autoritativního serveru na záznam SOA a ověřuje, že podpisy záznamu jsou stále
# platné a v následujících 24 hodinách nevyprší.

# Tento komentovaný a do jisté míry upovídaný skript je napsán pouze pro demonstrační účely, a proto nejsou testovány
# případné chybové stavy.

use strict;
use Net::DNS::SEC;
use Time::Local;

# Doména a její master server.
my $domain="secret-wg.org";
```

```
my $authoritative_server="ns.secret-wg.org";

# Nastavení resolveru (použijte dokumentaci pro Perl Net::DNS::Resolver této třídy a těchto metod)
my $res = Net::DNS::Resolver->new();

# Dotazujte se implicitního resolveru, abyste zjistili, jaká je adresa autoritativního serveru.

my $answerpacket_auth_server= $res->query($authoritative_server,"A");

# Prostudujte paket viz dokumentace pro Perl Net::DNS::Packet a dokumentace pro Perl Net::DNS::RR::A. Nebereme
# v úvahu ověřování chyb. Předpokládáme, že první RR v oddíle odpovědí je A RR pro ns.secret-wg.org.

my $auth_address=($answerpacket_auth_server->answer)[0]->address;

# Nastavte cílový resolver tak, aby dotazoval autoritativní server.
$res->nameserver( $auth_address );

# Nastavte resolver tak, aby komunikoval DNSSEC
$res->dnssec(1);

# Pošlete dotaz pro SOA autoritativnímu jmennému serveru.
my $packet=$res->send($domain,"SOA");

# Prostudujte oddíl odpovědí a uvědomte si, že může obsahovat více než jeden RRSIG (pro definici existuje
# vždy jeden SOA RR).

my $soa;
my @soasig;
foreach my $rr ( $packet->answer ){
if ($rr->type eq "SOA"){
$soa=$rr;
next;
}
if ($rr->type eq "RRSIG"){
push @soasig,$rr;
next;
}
}


die "NO SOA RR found" unless $soa;
die "NO RRSIGs over the SOA found" unless @soasig;
print @soasig ." signatures found\n";

# Získejte klíče, které patří do této zóny (DNSKEY jsou aktivní jako SOA v apexu).

my @keyrr;

$packet=$res->send($domain,"DNSKEY");
foreach my $rr ( $packet->answer ){
if ($rr->type eq "DNSKEY"){
push @keyrr,$rr;
next;
}
}


die "NO DNSKEYS found for $domain" unless @keyrr;



# Nyní zpracujte všechny podpisy, získejte veřejnou část klíče, pomocí které byl vytvořen,
# ověřte podpis a porovnejte data.
# Přístupové metody k vnitřním položkám RRSIG naleznete v dokumentaci pro Perl Net::DNS::RR::RRSIG

SIGLOOP: foreach my $sig ( @soasig ){
print "Checking signature made with key ".$sig->keytag ."\n";
# verify the signature.
# first select the key with the proper keytag from the key set.
my $keyfound=0;
KEYLOOP: foreach my $key (@keyrr){
next KEYLOOP if ($key->keytag != $sig->keytag);
$keyfound=$key;
last KEYLOOP;
}
print "WARNING: NO public key found to validate:\n " .
$sig->string."\n" unless $keyfound;

# Proveďte reálné ověření.
if (! $sig->verify([ $soa ],$keyfound)){
# Podpis se neověřil. Řekněte proč.
print "WARN: Signature made with " .$sig->keytag . " failed to verify:\n".
$sig->vrfyerrstr;

}else{
```

```
# Podpis se ověřil.
# Ověřme si, zda máme více než 24 hodin před vypršením.

$sig->sigexpiration =~ /(\d{4})(\d{2})(\d{2})(\d{2})(\d{2})(\d{2})/;
my $expiration=timegm ($6, $5, $4, $3, $2-1, $1-1900);
my $hourstogo=($expiration-time())/3600;
print "WARNING: Signature made with ".$sig->tag. "will expire within ".
$hourstogo . " hours\n" if $hourstogo <24;

}

}



####
# $Id: expire.pl 21 2004-10-11 14:52:09Z olaf $
####
```