ISA - Síťové aplikace a správa sítí

Laboratorní manuál

Vysoké učení technické v Brně

`https://github.com/nesfit/ISA/tree/master/manual`

Tento laboratorní manuál slouží jako referenční příručka pro laboratorní cvičení předmětu ISA – Síťové aplikace a správa sítí. Očekává se, že si studenti před účastí na jednotlivých cvičeních přečtou sekce potřebné pro dané cvičení.

Sekce 1, 2, 3, 4 a 5 jsou potřebné pro všechny cvičení. Cvičení *Zabezpečený přenos dat* navíc předpokládá znalosti ze sekce 6, 7 a 8. Ve cvičení *DNS, DNSSEC* se využívají sekce 9 a 10. Cvičení *Konfigurace a analýza přenosů VoIP* předpokládá znalosti uvedené v sekci 14. Cvičení *Správa a monitorování sítě* staví na informacích uvedených v sekcích 11 a 13.

Obsah

1 Adresy v IPv4 síti

1.1 Formát adresy

IPv4 adresa je délky 32 bitů a preferovaný zápis má formát `X.X.X.X`, kde `X` je decimální zápis 8 bitového čísla. Příklad:

```
8.8.8.8
127.0.0.1
```

Adresa dvě části:

- adresa sítě (**prefix**) a
- adresa uzlu.

Délka prefixu se zapisuje v desítkovém tvaru za lomítko. `192.168.0.1/24`

V síti s daným prefixem existují 2 speciální adresy, které nelze použít pro adresování jednotlivých uzlů:

- adresa sítě - adresa uzlu je nulová, např. 192.168.0.0/24
- broadcastová adresa - nejvyšší možná adresa uzlu pro daný prefix, např. 192.168.0.255/24

2 Adresy v IPv6 síti

2.1 Formát adresy

IPv6 adresa je délky 128 bitů a zapsaných ve formátu `X:X:X:X:X:X:X:X`, kde `X` je hexadecimální zápis 16 bitového čísla. Příklad:

```
FEDC:BA98:7654:3210:fedc:ba98:7654:3210
1080:0000:0000:0000:0008:0800:200C:417a
```

Preferovaný formát je dále upraven v RFC 5952[1], které definuje následující pravidla:

- Nuly na začátku každého 16 bitového čísla je potřeba vynechat (např. 80 namísto 0080 a 0 namísto 0000).
- Více nulových bloků lze nejvýše jednou nahradit znakem `::`, a MUSÍ být nahrazena nejdelší možná posloupnost takových nulových bloků. V případě více shodně dlouhých posloupností, nahrazuje se ta nejvíce vlevo.
- Znak `::` nesmí nahrazovat samostatný nulový blok.
- Znaky "a", "b", "c", "d", "e", "f" hexadecimální soustavy se vždy píšou malými písmeny.

Adresy z předchozího příkladu tedy budou vypadat následovně:

```
fedc:ba98:7654:3210:fedc:ba98:7654:3210
1080::8:800:200c:417a
```

Stejně jako v IPv4 má adresa dvě části: *adresa sítě (***prefix***)* a *adresa uzlu*. Délká prefixu se zapisuje v desítkovém tvaru za lomítko (např. `1080::/60`).

[1] https://tools.ietf.org/html/rfc5952

2.2 Rozdělení adres

IPv6 adresy je možno rozdělit podle rozsahu. Typicky může jít o tři možnosti – adresy na lince (neprojdou za router), lokální adresy (ULA, nejsou routovatelné ve veřejné sítí) a veřejné adresy.

Význam	Prefix (bitově)	Prefix
Veřejné	`001`	`2000::/3`
Lokální	`1111 110`	`FC00::/7`
Linkové	`1111 1110 10`	`FE80::/10`
Multicast	`1111 1111`	`FF00::/8`

2.3 Lokální IPv6 (ULA) adresy

Tyto adresy jsou vhodné pro malé sítě zahrnující jedno místo, nebo organizaci. Adresy ULA nejsou globálně směrovatelné (ale nic nebrání jedné, či více organizacím své adresy mezi sebou směrovat). Jedním z využití adres ULA je získání celosvětově unikátního prefix bez formálních požadavků na registraci adres.

Síťová část ULA adresy se stává ze 4 části:

Prefix `fc00::/7`

L bit 1 pokud byl prefix přiřazen lokálně, 0 zatím nebyla definována, začátek adresy je proto typicky `fd00::/8`

Global ID identifikátor sítě, měl by být unikátní, standard popisuje pseudonáhodný algoritmus pro generování. Unikátnost je požadována, aby při spojení více lokálních sítí nebylo třeba žádnou přečíslovat.

Inderface ID Identifikátor rozhraní, existuje několik variant jak jej získat. Původní standard (RFC 3513[2] a RFC 4291[3]) doporučoval použití EUI-64. V současnosti byla tato metoda nahrazena RFC 7217[4]. Pro ochranu soukromí jsou další specifika automatické generace identifikátorů definovány v RFC 4941[5].

7 bits	1	40 bits	16 bits	64 bits
1111 110	L	Global ID	Subnet ID	Interface ID

Lokální síť je tedy složená ze tří části:

- unikátní prefix délky 48 bitů,
- 16 bitový identifikátor podsítě,
- adresa uzlu o délce 64 bitů.

3 Základy konfigurace linuxového serveru

Detailní popis včetně možných voleb a příkladu použití k jednotlivým níže zmíněným příkazům můžete nalézt v manuálových stránkách (`man <příkaz>`).

[2] https://tools.ietf.org/html/rfc3513
[3] https://tools.ietf.org/html/rfc4291
[4] https://tools.ietf.org/html/rfc7217
[5] https://tools.ietf.org/html/rfc4941

3.1 Základní orientace v Linuxu

Většina práce v OS Linux bude probíhat v terminálu. Terminál na školních PC spustíte pomocí `Alt+F2`, zde zadáte "`gnome-terminal`". Případně můžete vybrat aplikaci terminálu z postraní nabídky.

3.1.1 Hierarchie souborového systému

Všechny soubory jsou uloženy v souborovém systému. Ten je organizován jako invertovaný strom adresářů, kde kořenovým adresářem je `root` adresář (označení `"/"`). Ten obsahuje podadresáře, kde každý má svůj standardizovaný účel pro zařazení různých souborů.

Některé podadresáře a jejich význam:

- **/etc** - obsahuje konfigurační soubory systému,
- **/var** - proměnlivá boot perzistentní data, dynamicky se mění, obsahuje databáze, cache adresáře, obsah www stránek (`/var/html/www`), logy (`/var/log`),
- **/dev** - obsahuje speciální soubory pro přístup k hardware zařízením.

3.1.2 Základní příkazy

Některé základní příkazy pro práci v terminálu OS Linux:

- `cd` - posun v adresářové hierarchii,
- `ls` - zobrazení obsahu adresáře,
- `cp` - kopírování souborů,
- `mv` - přesun souborů,
- `mkdir` - vytvoření adresáře,
- `touch` - vytvoření prázdného souboru,
- `head` - zobrazení x řádků od začátku souboru,
- `tail` - zobrazení x řádků z konce souboru,
- `cat` - zobrazení obsahu souboru,
- `grep` - filtrování/hledání v textu,
- `sed` - editace textu v příkazové řádce,
- `awk` - skenování a zpracování textu,
- `ps` - zobrazení informací o běžících procesech.

Mezi command line editory patří například `nano` nebo `vim`. Kromě nich můžete taktéž použít grafické editory, například `gedit`.

3.1.3 Uživatelé

Každý uživatel v OS Linux má své jedinečné `uid`. Každý proces (program) v systému je spuštěn pod nějakým uživatelem. Každý soubor je vlastněn uživatelem a omezen přístupovými právy. Tato přístupová práva se vztahují taktéž na procesy spuštěné daným uživatelem.

- `root` - super uživatel, má veškerá práva a kontrolu nad systémem.
- `user` - pouze základní správa, bez možnosti zasahovat do systémového nastavení.
- `sudo` - delegace některých práv super uživatele na normálního uživatele (nastavení v `/etc/sudoers`).

Při správě OS se pokud možno snažíme vyhnout používání systému jako `root`. K tomuto slouží `sudo`, kde má daný uživatel přesně specifikováno, které operace smí se systémem provádět, a je pak lépe dohledatelné, kdo co nastavil.

Přepínání uživatele: `su [user]`

- `su` - přepnutí na uživatele `root`.
- `su user` - přepnutí na uživatele `user`.

Spouštění příkazů jako `sudo`: `sudo command...`

3.2 Správa systémových služeb

OS Linux poskytuje řadu různých aplikací plnících roli systémových služeb, tyto mohou poskytovat užitečné funkce jiným aplikacím, uživatelům nebo spravovat konfiguraci subsystémů. Systémové služby typicky běží autonomně, na pozadí systému bez přímého rozhraní pro uživatele. Naopak systém poskytuje speciální rozhraní pro jejich manipulaci. Pro Linuxové distribuce se `systemd` máme dostupnou aplikaci `systemctl`, která poskytuje několik základních příkazů pro manipulaci systémových služeb:

- `systemctl start [služba]` spustí službu
- `systemctl stop [služba]` zastaví službu
- `systemctl restart [služba]` restartuje službu
- `systemctl enable [služba]` povolí automatické spuštění služby po startu systému
- `systemctl disable [služba]` zakáže automatické spuštění služby po startu systému
- `systemctl status [služba]` vypíše informace aktuálního stavu služby a několik posledních řádků systémových logů této služby. Tento příkaz můžete použít i bez specifikace služby, dostanete tak informace o všech aktuálně spuštěných službách.

Jméno služby má typicky tvar `<aplikace>.service`, například `sshd.service`.

Pro zobrazení kompletních logů konkrétní systémové služby můžete použít příkaz `journalctl -u [služba]`.

Detailní popis včetně možných voleb a příkladů použití můžete nalézt v manuálových stránkách.

3.3 Konfigurace síťových zařízení

Ke zjišťování síťové konfigurace na daném zařízení můžeme použít několik nástrojů, které jsou na OS Linux k dispozici.

3.3.1 Zobrazení konfigurace

Jedním ze základních příkazů je `ip`. Pomocí tohoto příkazu můžeme zjistit konfiguraci všech síťových zařízení (fyzických i virtuálních), obsah routovací tabulky aj. Manuálové stránky pro dané možnosti zobrazíte jako `ip-<volba>`.

- `ip address` - zobrazí adresu na všech síťových zařízení (příkaz `ifconfig` je dnes již zavržený[6]).
- `ip route` - zobrazí všechna pravidla v routovací tabulce.
- `ip link` - vylistuje všechna síťová zařízení.
- `ip neighbour` - zobrazí obsah ARP záznamů (případně můžete využít starší příkaz `arp`).

Tento nástroj mimo jiné slouží také ke konfiguraci, ne jen k jejímu zobrazení. [7]
Alternativou k zobrazení obsahu routovací a ARP tabulky jsou tyto příkazy:

- `netstat` - zobrazí mimo jiné obsah routovací tabulky (volba `-rn`)

K zobrazení aktuálně běžících spojení včetně protokolu, portu, zdrojové a cílové adresy slouží příkaz `ss` (`sockstat`). Příkaz `lsof -ni` zobrazuje otevřená spojení.

3.3.2 Konfigurační soubory a logy

Na OS Linux je veškeré nastavení systému a služeb uloženo v adresáři `/etc`. Jednotlivé služby zde mají svůj konfigurační soubor s příslušným názvem. Pro lepší přehlednost je možné vytvářet více konfiguračních souborů, pro tyto účely zde pak existují adresáře `"nazevsluzby.d/"` (například `/etc/rsyslog.d/`). Veškeré nastavení v souborech v těchto adresářích je pak aplikováno na danou službu.

Některé ze základních podstatných konfiguračních souborů:

- `/etc/hostname` - obsahuje hostname systému.
- `/etc/hosts` - obsahuje mapování hostname na IP adresu (zde můžete libovolné adrese přiřadit hostname, následné použití tohoto hostname předchází samotnému vyhledání pomocí DNS resolveru).
- `/etc/host.conf` - obsahuje konfiguraci specifickou pro resolver.
- `/etc/resolv.conf` - obsahuje direktivy specifikující výchozí servery, na které je poslán dotaz pro překlad doménového jména na IP adresu.
- `/etc/rsyslog.conf` - obsahuje nastavení syslogu, rozřazování jednotlivých zpráv do příslušných logovacích souborů.

Veškeré podstatné události jsou zaznamenávány do logů. Všechny logovací soubory jsou uloženy v adresáři `/var/log`. Tyto soubory (logy) jsou cyklicky promazávány po určitém čase či po dosažení určité velikosti. Dobu, po kterou jsou logy udržovány, lze nastavit v konfiguračním souboru `/etc/logrotate.conf`.

Některé významné logovací soubory:

- `/var/log/messages` - obsahuje obecné zprávy systému (bez kritických a debugovacích).
- `/var/log/auth.log` - zde jsou uloženy zprávy týkající se autentizace uživatelů.

[6] `https://www.root.cz/clanky/prikaz-ip-ovladnete-linuxova-sitova-rozhrani/`
[7] Obdobou tohoto nástroje je `ifconfig`.

- `/var/log/kern.log` - ukládá zprávy týkající se jádra OS.

Tyto logovací soubory můžeme prohlížet přímo pomocí standardních zobrazovacích příkazů.

Systémové události můžeme také prohlížet pomocí nástroje `journalctl`, ten nám umožní procházet podrobnější informace daných zpráv, filtrovat a vyhledávat.

Příklad použití `journalctl`:

- `journalctl -n <x>` - zobrazí posledních x záznamů.
- `journalctl -p <priority>` - zobrazí pouze záznamy s danou syslog prioritou.
- `journalctl -u <unit|pattern>` - umožní vyfiltrovat pouze záznamy s předanou službou (spravovanou systemd), nebo záznamy odpovídající předanému vzoru.
- `journalctl -t <syslog_id|pattern>` vyhledává v syslogovém záznamu podle názvu služby (tzn. i služby, které nejsou spravovány systemd, např. *sudo*).
- `journalctl -f` - zobrazí kontinuálně posledních 10 událostí.
- `journalctl --since <whenstart> --until <whenend>` - zobrazí záznamy v daném rozmezí.

3.3.3 Aktivní zjišťování

Zjišťování dostupnosti zařízení a konektivity. K tomuto účelu poslouží příkazy:

- `ping ipv4|domain` - zašle ICMP ECHO_REQUEST k danému hostu (pro ipv4).
- `ping6 ipv6|domain` - zašle ICMP ECHO_REQUEST k danému hostu (pro ipv6).
- `traceroute ipv4|domain` - zobrazí cestu, kudy packet prochází skrz síť k danému hostu (pro ipv4).
- `traceroute6 ipv6|domain` - zobrazí cestu, kudy packet prochází skrz síť k danému hostu (pro ipv6).
- `tcptraceroute ipv4/ipv6|domain` - `traceroute` používající TCP.

Mimo základní výše uvedené příkazy `ping` a `traceroute` existuje užitečný nástroj `nmap`.

- `nmap` - umožňuje pomocí různých protokolů aktivně zjišťovat otevřené porty, aktivní hosty na dané síti apod.

Pro přihlášení na vzdálený host můžeme použít dva nástroje:

- `telnet ip` - nešifrované přípojení
- `ssh ip` - šifrované přípojení

4 Analýza síťového provozu programem Wireshark

Wireshark je aplikace sloužící k analýze protokolů a zachytávání paketů. Zachytává síťový provoz z jednotlivých síťových zařízení (jak fyzických, tak virtuálních) a umožňuje inspekci jednotlivých polí v daných paketech.

Alternativou ke grafickému Wireshark je command line nástroj `tcpdump`. Wireshark má oproti `tcpdump` například integrované volby pro řazení, filtrování a jiné.

4.1 Zachytávání provozu

Zachytávat provoz můžeme z libovolného síťového zařízení, dokonce z více zařízení najednou.

Odchytíme pomocí dialogu dostupného pomocí `Capture -> Interfaces...`, kde vybereme zařízení, která chceme snímat. V dialogu je možné nastavit filtrace provozu již při zachytávání. Tato vlastnost může být užitečná při dlouhodobějším zachytávání, především z hlediska redukce množství ukládaných paketů. Tzv. *C*apture filter (zachytávací filtr) můžeme aplikovat před spuštěním zachytávání. Použijeme `Options` a v poli `Capture Filter:` můžeme zvolit jeden z předdefinovaných filtrů, případně specifikovat vlastní (vizte také `man 7 pcap-filter`). Záchyt spustíme pomocí tlačítka `Start` – v tomto nastavení odchytíme veškerý provoz na vybraných zařízeních.

Odchycený provoz můžeme uložit do souboru (přípona `.pcap`). Později je možné tyto soubory znovu analyzovat.

4.2 Zobrazovací filtr

V některých situacích, například pokud zachytáváme veškerý provoz, je velmi vhodné použít filtrování. Jelikož zde pracujeme s velkým množstvím různých paketů, můžeme si tak vyfiltrovat pouze ty relevantní. K tomu slouží tzv. display filtry, jejichž syntax je popsána v Tabulce 1.

Porovnávání	`==, >=, <=, !=, contains`
Logické operátory	`||, or, &&, and, !, not`
Kombinace filtrů	`(ip.src==192.168.0.105 and udp.port==53)`
Filtrování na základě existence pole	`http.cookie or http.set_cookie`
Filtrování specifických bytů	`eth.src[4:2]==22:1b`
Regex filtrování	`http.host && !http.host matches "com$"`

Tabulka 1: Filtrovací operátory

4.3 Flow graph

Wireshark umožňuje zobrazit vybranou komunikaci v `flow graph`. Zde můžeme přehledně vidět, které stroje, rozlišené IP adresou, spolu komunikují a kdo komu, zasílá kterou zprávu v jakém pořadí. Flow graph si zobrazíme pomocí `Statistics -> Flow Graph..`, zde si vybereme, co chceme zobrazit.

4.4 Zobrazení streamu

Zachycenou komunikaci vidíme v podobě jednotlivých paketů. Pokud například zachytíme HTTP komunikaci, jeden paket nese většinou pouze část obsahu HTTP zprávy. V tomto případě můžeme pro zobrazení celé zprávy (celého streamu) použít volbu `Flow <protokol> Stream`. Zobrazení provedeme tak, že vybereme jeden zachycený paket, označíme ho a následně `Analyze -> Follow <protokol> Stream`.

4.5 Čas

V rámci zachytávání paketů nese každý záznam informaci o čase. Defaultně je zde čas zobrazen v sekundách, od počátku zachytávání komunikace. Nicméně v `View -> Time Display Format` můžeme vybrat konkrétní zobrazení času a zjistit tak například, ve který den byl daný paket zachycen.

5 Konfigurace síťování koncových zařízení

5.1 Manuální konfigurace IP Adres

Pro dočasnou konfiguraci IP adres na OS Linux slouží příkaz `ip`. Pro dnešní cvičení budeme potřebovat následující příkazy:

- `ip link set [rozhraní] up/down` pro zapnutí/vypnutí rozhraní
- `ip addr add/del [IP adresa]/[prefix] dev [rozhraní]` pro přidání/odstranění adresy z rozhraní
- `ip addr flush dev [rozhraní]` pro odstranění všech adres z rozhraní
- `ip route add default via [IP adresa]` pro nastavení výchozí brány
- `ip link`, `ip addr`, `ip route` zobrazí aktuální konfiguraci

Pro trvalou konfiguraci použijeme grafické rozhraní NetworkManageru, dostupné přes aplikaci *Settings* v sekci *Network*; nebo v panelu vpravo nahoře.

5.2 Dynamická konfigurace IPv4 - DHCP

Pro dynamickou konfiguraci IP adres je potřeba aby byl na síti přístupný nakonfigurovaný DHCP server a klienti, kteří si o IP adresu požádají. Kromě přiřazení IP adres má DHCP server na starosti i šíření jiných informací důležitých pro bezproblémovou funkčnost sítě. Jednou takovou informací je IP adresa doporučeného DNS serveru pro klienty na síti.

Na Vašich počítačích máte nainstalovanou serverovou aplikaci ISC DHCP, která poskytuje služby DHCP serveru. Aplikace se konfiguruje pomocí souboru `/etc/dhcp/dhcpd.conf`. Systémová služba se jmenuje `isc-dhcp-server.service`.

Příklad obsahu konfiguračního souboru:

```
option domain-name-servers [IP adresy DNS serverů];
subnet [IP adresa sítě] netmask [maska] {
    range [první přiřaditelná IP adresa] [poslední přiřaditelná IP adresa];
}
```

Dodatečné informace o konfiguračních možnostech najdete v manuálových stránkách `man dhcpd.conf` případně `man dhcpd`.

DHCP klient je implementován v aplikaci `dhclient`. Může být spuštěn bez parametrů pro všechna aktivní síťová rozhraní nebo lépe s jménem konkrétního síťového rozhraní, které má být konfigurováno:

```
dhclient -v [rozhraní]
```

Argument `-v` zajistí, že aplikace vypíše detailní informace. Chybu týkající se `smbd.service` někdy zobrazovanou v laboratoři můžete ignorovat.

5.3 Dynamická konfigurace IPv6

Dynamická konfigurace IPv6 se dělí na **bezstavovou** a **stavovou** konfiguraci. Na cvičeních se budeme zabývat jen bezstavovou. Bezstavová konfigurace byla navržena tak, aby stačilo připojit zařízení do sítě a automaticky si klient vygeneroval nějakou adresu a ihned mohl komunikovat se světem. K tomu klient potřebuje znát prefix sítě do které byl připojen. K tomuto účelu se používají zprávy označované jako Routing Advertisement (RA). Tyto a ještě další zprávy jsou součásti procesu Neighbor Discovery [16].

Na počítačích v laboratoři je nainstalovaná systémová služba `radvd.service`, která je schopna vysílat zprávy RA. Aplikaci je možno nakonfigurovat pomocí souboru `/etc/radvd.conf`.

Příklad konfigurace:

```
interface [rozhraní]
{
    AdvSendAdvert on;
    MaxRtrAdvInterval [Max pocet sekund mezi zpravami RA, min 4];
    prefix [prefix]/[delka prefixu]
    {
        AdvOnLink on;
        AdvAutonomous on;
        AdvRouterAddr on;
    };
};
```

Přehled všech možností konfigurace poskytnou manualové stránky `man radvd` a `man radvd.conf`.

Pro správné šíření RA zpráv je potřebné povolit směrování IPv6 provozu:

```
sysctl net.ipv6.conf.all.forwarding=1
```

Součásti balíčku radvd je i aplikace `radvdump`, kterou možno použít pro analýzu – naslouchá na síťových rozhraních a tiskne na obrazovku obsah zachycených RA zpráv.

6 Network Time Protocol

NTP [13] umožňuje synchronizovat čas mezi uzly v síti. Tento protokol se dokáže vypořádat s proměnlivou dobou přenášení paketu po síti. NTP organizuje servery hierarchicky do úrovní. Tyto úrovně se nazývají *stratum*. Nejnižší hodnota 0 označuje samotný zdroj přesného času (např. GPS). Stratum 1 pak označuje servery, které jsou synchronizovány právě s referenčním zdrojem. Stratum 2 jsou servery synchronizovány se servery stratum 1, atd. [14].

Implementaci protokolu NTP zajišťuje balík aplikací *ntp*. Tento balík se skládá z několika aplikací. V rámci laboratorního cvičení se použijí aplikace `ntpd`, `ntpq` a případně `ntpdc`. Chrony[8] nabízí alternativní implementaci NTP a v bezpečnostním auditu [5] překonal balík *NTP*.

6.1 ntpd

NTPD je aplikace, která běží na pozadí a neustále provádí kontrolu času s nastavenými servery a případně upravuje lokální čas. Úprava lokálního času se provádí úpravou rychlosti běhu lokálního času. Pokud je lokální čas pozadu, resp. se předbíhá, tak se *zrychlují*, resp. *zpomalují* systémové hodiny. Tento způsob úpravy času znamená, že pokud je čas odchýlen o několik minut, bude nějakou dobu trvat, než dojde k jeho srovnání. Na druhou stranu se tak zabrání skokové změně a navíc čas se nikdy neposune do minulosti. Pokud je lokální čas odchýlen o více než 1000 sekund (necelých 17 minut) aplikace to vyhodnotí jako chybu a skončí. Pokud se tak stane, objeví se zpráva v systémovém logu. Zda aplikace běží, lze zjistit například příkazem `ntpq -p` (aplikace skončí s hláškou *ntpq: read: Connection refused* pokud není ntpd spuštěn).

Démona je možné spustit příkazem:

```
systemctl start ntp.service
```

Aby se předešlo problému v případě, kdy např. hardwarové hodiny jdou špatně a při vypnutí může dojít k odchýlení lokálního času o více než 17 minut, je možné vynutit okamžité nastavení času příkazem

```
ntpd -qg
```

(`-q` pro vynucení okamžitého nastavení času a `-g` pro vypnutí kontroly kdy je rozdíl větší než 1000 sekund).

[8] `https://chrony.tuxfamily.org/`

6.1.1 Konfigurace

Základním konfiguračním souborem je `/etc/ntp.conf`. Tento konfigurační soubor obsahuje mnoho konfiguračních voleb. Nastavení serverů NTP zajišťuje konfigurační volba *server*, která má jeden parametr (adresu nebo jméno NTP serveru). Tato volba se může vyskytovat opakovaně, klient pak využívá větší počet serverů a tím lze dosáhnout vyšší přesnosti. Volba server mimo jiné znamená, že lokální čas se může nastavit podle uvedeného serveru, ale nemůže tomu být naopak. V jiných případech může být volba *server* nahrazena volbou *peer*, která umožňuje, aby se i server synchronizoval podle lokálních hodin. Tato volba je užitečná, pokud je více ekvivalentních serverů, k zajištění, že se budou synchronizovat navzájem mezi sebou. Dalšími možnostmi jsou *broadcast* a *manycastclient*. Tyto možnosti využívají broadcastového nebo skupinového vysílání (pokud je nutné synchronizovat velký počet uzlů může tato možnost šetřit síťové zdroje). Server v této konfiguraci vysílá na broadcastovou nebo multicastovou adresu informace o správném čase v pravidelných intervalech a klienti zpracovávají tyto informace.

Jinou užitečnou volbou je *restrict*, která slouží pro řízení přístupu. Aplikací `ntpd` mohou být zaslány různé požadavky přes síť (viz aplikace `ntpq`/`ntpdc`). Je vhodné dovolit některé dotazy jen z určitého uzlu nebo podsítě. Využití této volby může být také užitečné v případě, že se pro nastavování času využívá broadcastu nebo multicastu a není žádoucí, aby takto vyslanou informaci klienti akceptovali z libovolného zdroje. Základní tvar tohoto příkazu je:

```
restrict <adresa> [mask <maska>] [<jeden či více příznaků>]
```

Příznak definuje omezení pro danou adresu/síť:

- ignore – zahazovat všechny pakety
- nomodify – povolí pouze dotazy, požadavky měnící stav serveru jsou zahazovány
- noquery – zakáže dotazy pomocí `ntpq` a `ntpdc`, synchronizace času není ovlivněna
- notrust – zahazovat neautentizované pakety

Další příznaky a jejich popis lze nalézt v manuálových stránkách `man ntp.conf`.

Pokud budou aplikace `ntpq` a `ntpdc` používané i pro změnu konfigurace, pak je nezbytné nastavit autentizaci. Protokol nabízí možnost využití symetrické i asymetrické kryptografie. Pro použití symetrické kryptografie jsou k dispozici tyto volby:

- keys – tato volba má jeden parametr, který udává název souboru, který obsahuje používané klíče (obvykle `/etc/ntp.keys`,
- trustedkey – výčet klíčů, kterým se bude důvěřovat,
- requestkey – seznam klíčů, které mohou být použity aplikací `ntpdc`,
- controlkey – seznam klíčů, které mohou být použity aplikací `ntpq`.

Formát souboru `/etc/ntp.keys` má následující tvar:

```
<číslo klíče> <typ> <heslo>
```

Typ může nabývat čtyř hodnot. Hodnoty `S` a `N` používají bitový formát a běžně se neužívají. Hodnoty `A` a `M` používají textový řetězce délky 1 až 8 znaků a určují způsob zašifrování při přenosu. Nejčastěji se užívá možnost `M`, která značí použití DES nebo MD5. Příklad souboru pak může vypadat následovně:

```
1 M heslo1
2 M secret
3 M passwd
```

Konfigurace v `/etc/ntp.conf` potom vypadá:

```
keys /etc/ntp.keys
trustedkey 1 2 3
requestkey 2
controlkey 1 3
```

Toto značí, že důvěryhodné jsou všechny tři klíče. Aplikace `ntpdc` se může autentizovat pouze klíčem 3 a aplikace `ntpq` klíči 1 a 3.

6.2 Aplikace ntpdc a ntpq

Oba dva programy nabízejí v podstatě podobné možnosti konfigurace ntp serveru. Rozdílů je mezi těmito programy několik. První, který byl již uveden výše, je v tom, že každá z těchto aplikací může používat jinou množinu klíčů. Podstatnější rozdíl z uživatelského hlediska je v tom, že aplikace `ntpdc` má přesně definovaný výčet příkazů, které lze aplikovat, a proto může měnit jen to, co je v aplikaci definováno. Naproti tomu `ntpq` disponuje příkazem `:config`, kterému se jako parametry předávají konfigurační volby, které se používají v souboru `/etc/ntp.conf`. Další rozdíl je ve formátu zpráv, pomocí kterých komunikují se serverem.

Obě aplikace používají pro komunikaci s aplikaci `ntpd` zasílání zpráv přes síťové rozhraní, bez ohledu na to, zda běží lokálně nebo vzdáleně. Hlavní výhoda však spočívá v tom, že tyto aplikace dovolují měnit konfiguraci `ntpd` za běhu. Možnost změny parametru přes síť může být nežádoucí a proto, je-li tato možnost povolena, je vhodné omezit pomocí volby *restrict* přístup pro změnu pouze přes loopback rozhraní.

Příklad výstupu volání `ntpq -p`:

```
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
+rhino.cis.vutbr 248.205.243.78   3 u  425 1024  377   10.070   -4.280  10.575
+tik.cesnet.cz   195.113.144.238  2 u  622 1024  377    4.796   -3.464  16.528
*tak.cesnet.cz   .GPS.            1 u  364 1024  377    5.008   -3.625   6.228
```

Výpis obsahuje následující hodnoty:

remote Klient se synchronizuje vůči třem serverům: rhino.cis.vutbr.cz, tik.cesnet.cz a tak.cesnet.cz. Hvězdičkou je označený primární zdroj času, plusem sekundární zdroje času.

refid Primární zdroj času pro vzdálený server NTP.

stratum Počet skoků od vzdáleného serveru k přesnému zdroji času (1 znamená, že zdroj přesného času je přímo připojen, 16 znamená, že vzdálený server je nedosažitelný).

when Počet sekund od posledního kontaktu se serverem.

poll Počet sekund mezi jednotlivými dotazy protokolem NTP.

reachability Úspěšnost posledním 8 pokusů o kontakt vzdáleného serveru v osmičkové soustavě. (1 znamená pouze poslední pokus uspěl, 377 znamená všech 8 posledních pokusů uspělo).

delay, offset, jitter Zpoždění, posun a jitter – charakteristiky vzdáleného zdroje času, podle kterého se volí primární a sekundární zdroje času.

7 Secure Shell

Základní funkcí protokolu Secure Shell (SSH) [21] je umožnění bezpečného přístupu ke vzdálenému počítači přes nezabezpečenou síť. Díky tomu, že je protokol SSH navržen obecně, lze pomocí něj zabezpečovat i další služby, jako je např. X Window, přístup ke vzdálenému souborovému systému (SFTP, sshfs, scp), tunelování portů TCP apod. Protokol SSH zajišťuje šifrování dat, autentizaci, integritu dat a volitelně také kompresi přenášených dat.

V rámci předmětu ISA se zaměříme pouze na malou část možností, které protokol SSH přináší. V rámci cvičení si vyzkoušíme protokol SSH pro terminálový přístup ke vzdálenému stroji a ukážeme si využití přihlašování ke vzdálenému počítači pomocí klíčů. Dále si vyzkoušíme, jak využít SSH k vytvoření jednoduché HTTP proxy.

Jednou z nejčastěji používaných aplikací pro využití protokolu SSH je sada programů OpenSSH. Balíček programů obsahuje kromě klienta `ssh` i serverovou aplikaci `sshd`, program pro generování SSH klíčů `ssh-keygen`, agenta pro usnadnění práce s SSH klíči `ssh-agent` a další. My budeme předpokládat, že na počítačích, na které se budeme snažit připojit již běží SSH server `sshd`. Konfigurace tohoto programu je nad rámec tohoto manuálu. Zájemci mohou nalézt podrobnější informace v manuálové stránce `sshd_config(5)`, či v jiných návodech.

7.1 Připojení ke vzdálenému počítači

Pro připojení se k počítači pojmenovaném *h01* a otevření příkazového řádku na vzdáleném stroji je možné použít příkaz:

```
ssh h01
```

Příkaz ssh má celou řadu parametrů, které jsou detailně popsány v manuálové stránce `ssh(1)`. Z těch nejčastěji používaných zmíníme alespoň změnu uživatelského jména (`-l`), specifikování TCP portu vzdáleného serveru (`-p`), zvýšení výřečnosti programu (`-v`, tento parametr lze použít i vícekrát), zapnutí tunelování protokolu X Windows (`-X`, `-Y`) a přesměrování portů (`-L`). Často zadávané parametry se specifikují v konfiguračním souboru `~/.ssh/config`. Popis tohoto souboru obsahuje manuálová stránka `ssh_config(5)`.

Po připojení ke vzdálenému serveru jsou informace o použitém spojení dostupné např. v rámci proměnných prostředí. Proměnné prostředí související s protokolem SSH je možné zobrazit příkazem `env | grep SSH`. Následující výpis ukazuje příklad proměnných po připojí k serveru `merlin` protokolem IPv6:

```
local $ ssh merlin6.fit.vutbr.cz
merlin $ env | grep SSH
SSH_CLIENT=2001:67c:1220:80c:e138:4d11:c04c:c675 54514 22
SSH_TTY=/dev/pts/30
SSH_CONNECTION=2001:67c:1220:80c:e138:4d11:c04c:c675 \
  54514 2001:67c:1220:8b0::93e5:b013 22
```

Z výpisu vidíme, že připojení bylo realizováno na IPv6 adresu `2001:67c:1220:8b0::93e5:b013` z počítače s adresou `2001:67c:1220:80c:e138:4d11:c04c:c675`. Byl použit zdrojový port č. 54514, na straně serveru byl použit standardní protokol 22. Po připojení využíval vzdálený uživatel terminál č. 30. Odhlášení ze vzdáleného počítače probíhá standardními prostředky pro ukončení shellu, např. příkaz `exit`, nebo vložení konce souboru klávesovou zkratkou `Ctrl-D`.

7.2 Kopírování souborů mezi počítači

Pro kopírování souborů protokolem SSH je často využívaná utilita `scp`. Cesta na vzdáleném serveru je specifikována v následujícím formátu: `uživatel@jménoserveru:cesta`. Následující příkaz zkopíruje lokální soubor isa na vzdálený počítač h01 do složky fit umístěné v domovské složce uživatele student.

```
scp isa student@h01:fit/
```

7.3 SSH jako proxy a tunelování portů

Díky velmi obecné implementaci lze protokolem SSH tunelovat jakýkoli typ provozu. Je tedy možné např. zajistit šifrování provozu po určitou část komunikace, konkretně po stanici, ke které se připojujeme SSH protokolem, nebo přesměrovat provoz z některého portu na jiný port. Následujcí příkaz nám otevře SSH spojení, které pak můžeme použít jako proxy ve webovém prohlížeči.

```
ssh -D 12345 -N student@sshserver
```

Po autentizaci stačí pak v prohlížeči zadat jako nastavení proxy server localhost s portem 12345 a provoz bude směrován nejprve na localhost, poté půjde ssh spojením na sshserver, kde bude převeden na běžný HTTP/HTTPS provoz a poslán dále.

7.4 Klíče protokolu SSH

Klíče protokolu SSH mají několik výhod. Díky nim není nutné posílat heslo pro přístup ke vzdálenému stroji přes síť, byť v šifrované podobě. Délka používaných klíčů znesnadňuje potenciálnímu útočníkovi útok hrubou silou, protože síla klíče bývá typicky vyšší než síla běžně používaných hesel. Ve spojení s agentem pro správu klíče může uživatel přistupovat ke vzdálenému serveru bez nutnosti opakovaného zadávání hesla. Agent může být nastaven, aby pro použitý klíč nevyžadoval heslo po zbytek sezení, či po určitý počet minut.

SSH klíč se obvykle generují utilitou `ssh-keygen`. Po spuštění bez parametrů je vytvořen pár klíčů algoritmem RSA o délce 2048 B. Uživateli je nabídnuto umístění souboru, které může změnit. Dále je uživatel požádán o zadání passfráze, od které se očekává, že bude silnější než heslo. Parametr `-t` nastavuje jiný typ šifrovacího algoritmu, parametr `-b` mění délku klíče, parametrem `-f` specifikuje umístění souboru, parametr `-C` upravuje popis klíče, další parametry popisuje manuálová stránka `ssh-keygen(1)`. Následující příklad vygeneruje klíč o délce 4096 B, který nebude chráněn passfrázi:

```
ssh-keygen -t rsa -b 4096 -N "" -f ~/.ssh/nopass -C nopass
```

Po vytvoření klíčů vzniknou dva soubory. Ten který je bez přípony `.pub` je soukromý klíč, který by měl zůstat tajný a uživatel, který jej vytvořil by jej neměl dále distribuovat. Soubor s příponou `.pub` je určen pro další distribuci, protože data zašifrovaná tímto klíčem dešifruje pouze tajný soukromý klíč.

7.5 Konfigurace použití klíčů

Nejdříve je potřeba distribuovat soubor s veřejným klíčem na vzdálený počítač. K tomu slouží např. program `scp`. Každý z uživatelů si může specifikovat sadu klíčů pro přístup k danému stroji v souboru `~/.ssh/authorized_keys`. Nový klíč do tohoto souboru přidáme např. takto (všimněte si, že se klíč přidává na konec souboru pomocí `>>`):

```
cat id_rsa.pub >> ~/.ssh/authorized_keys
```

Nyní se již můžeme připojit ke vzdálenému počítači pomocí vytvořených klíčů. Pokud jsme klíč na lokálním počítači umístili do výchozího umístění, je klíč použit automaticky. Pokud jsme zvolili jiné umístění, je potřeba program `ssh` informovat o umístění klíče parametrem `-i`, nebo volbou `IdentityFile` v konfiguračním souboru. Informace o hledaných klíčích jsou zobrazeny po použití parametru `-v`.

8 Transport Layer Security

Původní protokoly, na kterých vznikaly původní sítě a počátky Internetu byly nešifrované. Požadavek na zabezpečení dat se objevil až po vzniku nejdůležitějších protokolů jako je DNS, Telnet, HTTP, SMTP, FTP apod. Zatímco protokol SSH popisované v sekci 7 zavedlo nový protokol kompletně nahrazující protokol Telnet, většina ostatních protokolů založených na TCP (např. SMTP, HTTP, SIP) funguje v šifrované variantě stejně jako v nešifrované, jen mezi protokol aplikační vrstvy a TCP přidala mezi vrstvu Secure Sockets Layer (SSL) a Transport Layer Security (TLS), viz obr. 1, zajišťující šifrování. Výhodou tohoto přístupu je, že přidání podpory šifrované varianty do existující aplikace je v případě správného návrhu velice snadné a je potřeba upravit jen funkce zajišťující navazování spojení a zasílání zpráv; části programu vytvářející a zpracovávající zprávy aplikačního protokolu a samotný program není potřeba měnit.

Obrázek 1: Vrstvy TCP/IP modelu s využitím SSL/TLS.

Protokol TLS standardizuje IETF na základě původního protokolu SSL, který je dnes již zastaralý [3]. Poslední verze protokolu TLS je 1.3 [17][9].

TLS vytváří obousměrný šifrovaný kanál mezi dvěma počítači, který zajišťuje:

- Autentizaci: vždy se ověřuje identita serveru, identita klienta může být také ověřená. Pro ověřování se používají certifikáty X.509, které mohou být podepsány certifikační autoritou.
- Důvěrnost (utajení): po navázání spojení je obsah přenášené komunikace známý pouze komunikujícím stranám. Délka zpráv může být pozorovateli komunikace známá, TLS podporuje vycpávkový provoz.
- Integritu: Obsah zprávy není možné bez odhalení při přenosu upravit.

Samotný protokol TLS se skládá ze dvou podprotokolů:

- *Handshake protocol* se používá při navazování spojení a zajišťuje autentizaci komunikujících stran, domluvu kryptografických parametrů a sdílených klíčů.
- *Record protocol* přenáší data mezi komunikujícími stranami.

[9] Převyprávěná verze RFC 8446 je dostupná na `https://www.davidwong.fr/tls13/`

8.1 Certifikační autority

Certifikační autorita (CA)[10] je v asymetrické kryptografii subjekt, který vydává digitální certifikáty (elektronicky podepsané veřejné šifrovací klíče), čímž usnadňuje využívání Public Key Infrastructure (PKI) tak, že svojí autoritou potvrzuje pravdivost údajů, které jsou ve volně dostupném veřejném klíči uvedeny. Na základě principu přenosu důvěry tak můžeme důvěřovat údajům uvedeným v digitálním certifikátu za předpokladu, že důvěřujeme samotné certifikační autoritě.

Na Internetu působí mnoho komerčních certifikačních autorit, které obvykle mají své veřejné klíče umístěny přímo ve webových prohlížečích a dalších programech, čímž mohou uživateli zjednodušit rozhodování o míře důvěry webových serverů, ke kterým se připojuje (ale též digitálně podepsaných e-mailů i jiných dat). Existují též bezplatné certifikační autority nebo takové, které se řídí zákony daného státu, vnitřními předpisy organizace a podobně.

Hodnota digitálního certifikátu je úměrná míře důvěry, kterou máme k údajům v něm uvedených. Proto je pro certifikační autoritu nejdůležitější důvěra, kterou vůči svému okolí vzbuzuje (tj. že nevydá digitální certifikát s nepravdivými údaji). Certifikační autorita proto musí adekvátním způsobem pečovat o svoji důvěryhodnost, jinak by nebylo možné využít principu přenosu důvěry.

8.2 Transparentnost certifikátů

Aby bylo možné ověřit, že CA vydávají certifikáty poctivě, byl experimentálně zaveden mechanismus transparentnosti certifikátů. Zapojené CA veřejně ohlašují každý vystavený certifikát (případně pre-certifikát). Vydané certifikáty se shromažďují v lozích (cerificate transparency log), kde je monitorovat nově přidávané certifikáty a auditovat přítomnost certifikátu nalezených při prohlížení webu.

9 Domain Name System

Cílem DNS je zajistit překlad mezi doménovým jménem a IP adresou. Dříve se jednalo především o překlad hostname na IP adresu a naopak, tedy záznamy typu `A`, `AAAA`, `PTR` [15, 20]. Dalším známým typem je `MX`, který deleguje zodpovědnost za příjem e-mailové pošty dané domény. Tento typ se od předchozích tří odlišuje, protože se nejedná o adresaci hosta, ale služby. Podobný význam mají i záznamy typu `SRV`, které lze v současnosti využít pro služby SIP a XMPP [9].

Mimo tyto typy záznamů existují i další. Některé jsou definovány již v původním návrhu, jiné přidané později nebo význam původních typů je využit pro další služby (např. `TXT` se používá pro distribuci veřejného klíče podepisování emailu DKIM [1]).

9.1 Klient

Aby klient věděl, na který server se obrátit s požadavkem na přeložení doménového jména na IP adresu či opačně, je součástí systému tzv. resolver. Tento resolver má několik konfiguračních souborů, z nichž jmenujme `/etc/hosts`, `/etc/resolv.conf` a `/etc/host.conf`.

První z těchto souborů se používá pro statický překlad doménového jména na IP adresu. Formát souboru a další popis lze nalézt v manuálových stránkách `man hosts`.

Dynamický překlad, neboli překlad pomocí DNS serveru, vyžaduje existenci konfiguračního souboru `/etc/resolv.conf`, který obvykle obsahuje jedenkrát volbu `search` a jednu nebo více voleb `nameserver`.

search definuje tzv. searchlist, neboli seznam domén (max. 6), které se budou prohledávat, pokud je požadavek na překlad neúplného doménového jména. Tedy má-li tato volba podobu `search fit.vutbr.cz`, pak při pokusu přeložit jméno *merlin* se při neúspěchu pokusí systém také přeložit jméno *merlin.fit.vutbr.cz*.

[10] Text popisující CA je převzatý z `https://cs.wikipedia.org/wiki/Certifika%C4%8Dn%C3%AD_autorita`.

nameserver se uvádí právě s jedním parametrem, který definuje IP adresu DNS serveru, který se systém pokusí kontaktovat. Může se jednat jak o IPv4 tak o IPv6 adresu.

Další možnosti, které může soubor obsahovat, lze nalézt v manuálových stránkách `man resolv.conf`.

V souboru `/etc/host.conf` lze definovat, v jakém pořadí se předchozí dvě volby využijí. Standardně se nejprve zkouší statický překlad a poté dynamický. Toto výchozí nastavení umožňuje lokální předefinování překladu z doménového jména na IP adresu.

Pokud se síť na klientovi konfiguruje dynamicky, pak je soubor `/etc/resolv.conf` upravován automaticky. Při statické konfiguraci musí být obsah souboru upraven ručně.

9.2 Konfigurace serveru

Konfigurace DNS se stává ze dvou částí – konfigurace vlastností samotné aplikace a konfigurace obsluhovaných zón. Existuje mnoho různých variant implementací. V laboratoři se bude pracovat s implementací od ISC – BIND.

Konfigurační soubor **named.conf** se na počítačích v laboratořích nachází ve složce `/etc/`.

Další možnosti konfiguračního souboru naleznete v manuálových stránkách `man named.conf`.

9.2.1 Zóna typu hint

Asociuje se s nejvyšší doménou v rámci celé hierarchie. Pokud přijde na server požadavek, prohledá seznam spravovaných zón ostatních typů (*master*, *slave*, *forward*). Pokud záznam nenajde, vybere jeden z kořenových serverů definovaný právě v tomto typu zóny.

Soubor, který je vyžadován pro konfiguraci této domény lze získat například programem dig:

```
dig +norec NS . @a.root-servers.net > /var/named/db.root
```

9.2.2 Lokálně obsluhovane zóny

Tyto zóny lze rozdělit do dvou skupin – **loopback adresy** a **„prázdné zóny"**. RFC 6303 [2] je z kategorie *best practice* a definuje, které zóny by měl být schopen DNS server obsloužit sám a tím redukovat dotazy přeposílané na další servery. Ve většině případů jsou to zóny pro reverzní záznamy privátních IP adres.

9.2.3 Vlastní zóna

Aby server začal překládat vlastní zónu, je třeba přidat definici zóny do souboru `/etc/named.conf` a vytvořit zónový soubor. Pokud se přidává zónový soubor pro doménu, je vhodné přidat i zónu pro reverzní záznamy.

```
zone "moje.domena.cz" {
  type master;
  file "/var/named/moje.domena.cz.zone";
};

zone "0.168.192.in-addr.arpa" {
  type master;
  file "/var/named/192.168.0.rev";
};
```

Pro každou z těchto zón se tvoří samostatný zónový soubor, jehož základní tvar má podobu:

```
$TTL 5m

@ IN  SOA <fqdn autoritativniho serveru>. <email správce>. (
  1   ; seriové číslo
  10h ; obnovovací interval pro sekundární servery
  10m ; prodleva, po které se sekundární server pokusí znova kontaktovat
      ; autoritativní server, pokud se předchozi spojení nezdařilo
  1w  ; jestliže se sekundárnímu serveru nepodaří kontaktovat autoritativní
      ; server během této doby, přestane vracet záznamy pro tuto doménu
  1h  ; původně výchozí TTL (nahrazeno $TTL), nově určuje
      ; NEGATIVNÍ TTL -- pro chybové odpovědi (např. NXDOMAIN)
  )

@ IN  NS <fqdn autoritativního serveru>
@ IN  NS <fqdn sekundárního serveru>

následuje seznam dalších záznamů
```

Typ záznamu **SOA** definuje parametry zóny – jméno autoritativního serveru a email správce domény. Všimněte si, že znak '`@`' má zvláštní význam (zastupuje název domény). Z tohoto důvodu se v emailu správce nahrazuje znak '`@`' za znak '`.`' (tečka).

Záznam typu **NS** by měl být vždy obsažen alespoň jednou a to právě pro autoritativní server. Počet sekundárních serverů je libovolný. Tyto záznamy by také měly byt obsaženy v nadřazené doméně, aby bylo možné nalézt server zodpovědný za danou poddoménu. Tedy kořenová doména obsahuje `NS` záznamy pro domény nejvyšší úrovně *.cz, .com, .eu, ...* a ty zase pro jednotlivé poddomény *google.com, seznam.cz, ...*

V zónovém souboru se rozlišuje mezi relativním a absolutním jménem. O jaký typ jména jde, se rozlišuje podle zakončení. Končí-li tečkou, je to jméno absolutní (plně kvalifikované). V opačném případě jde o jméno relativní a za to se vždy doplní obsah definován v `$ORIGIN`. Jestliže není definován, použije se název, který je definován u názvu zóny v souboru `named.conf`.

Další záznamy mají obecný tvar:

```
jméno  ttl  třída  typ  <na typu závislá data>
```

Jméno běžně bývá ve tvaru relativním. Pokud není hodnota TTL jiná než je výchozí pro celou zónu, pak se může vynechat. Třída záznamu se běžně používá pouze `IN` (Internet). Kromě typu SOA a NS, se v laboratorním cvičení použijí záznamy typu `A`, `PTR` a volitelně `AAAA`. Typ `AAAA` se chová stejně jako záznam typu `A`, který se používá pro IPv4. Typově závislými daty je v tomto případě adresa IPv6, resp. IPv4. Příklad:

```
host  IN A  192.168.0.1

host  IN AAAA  fd12:1234::1
```

Typ `PTR` nerozlišuje, zda se jedná o IPv6 nebo IPv4 adresu. Pro tento případ se typově závislá data chovají stejně jako jméno na začátku záznamu. V tomto případě se ale vždy zapisuje v plném tvaru. Další zvláštností PTR záznamu, který je vidět i v názvu zóny, je, že se hodnoty zapisují v obráceném pořadí a každá hodnota je oddělená tečkou. Obrácené pořadí je kvůli vyhodnocování záznamů od konce a tečka mezi každým číslem dovoluje snadnější delegaci podsítě na jiný DNS server. Příklad:

```
1 IN PTR host.moje.domena.cz.
```

Pro IPv6 můžete předefinovat proměnnou `$ORIGIN` např. následovně:

```
$ORIGIN 0.0.0.0.0.0.0.0.4.3.2.1.2.1.d.f.ip6.arpa.
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0   IN PTR host.moje.domena.cz.
```

Pozor však na to, aby expanzí nedocházelo k vytváření neplatných záznamů jako jsou

```
$ORIGIN 0.0.0.0.0.0.0.0.4.3.2.1.2.1.d.f.ip6.arpa.
2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0   IN PTR host
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0   IN PTR host.moje.domena.cz
```

protože po expanzi se z nich stanou následující záznamy:

```
2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.4.3.2.1.2.1.d.f.ip6.arpa. IN PTR
                                                    host.0.0.0.0.0.0.0.4.3.2.1.2.1.d.f.ip6.arpa.
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.4.3.2.1.2.1.d.f.ip6.arpa. IN PTR
                                host.moje.domena.cz.0.0.0.0.0.0.0.4.3.2.1.2.1.d.f.ip6.arpa.
```

Příklad PTR záznamu pro IPv4 je:

```
19.176.229.147.in-addr.arpa. IN PTR merlin.fit.vutbr.cz.
```

Po každé změně zónového souboru by se vždy mělo upravit i sériové číslo zóny a to tak, aby bylo vždy větší než předchozí hodnota. Také je nutné aplikaci načíst změněnou konfiguraci.

```
systemctl restart named.service
```

10 DNS Security Extensions (DNSSec)

DNS je službou, která si z minulosti nese různé problémy. Jedním z nich je nezabezpečení této služby vůči útokům s podvržením záznamu. DNSSEC rozšiřuje službu DNS o bezpečnostní prvky, jejichž cílem je zabránit útokům využívající podvržené záznamy v DNS. Obrana spočívá v zavedení asymetrické kryptografie pro podepisování zón. Přesněji řečeno se podepíše každý záznam v zóně a tyto podpisy se zveřejní jako speciální typ záznamu. Protože není možné, aby resolver měl uložený veřejný klíč od každé zóny, používá se metoda sestavení důvěryhodného řetězce s tím, že resolver v tomto případě potřebuje mít uložen pouze veřejný klíč nejvyšší domény. Aby tento systém fungoval, musí nadřazená doména obsahovat nejen záznamy typu `NS` ale i záznam typu `DS` – speciální typ záznamu obsahující hash klíče a název domény. Principiálně samozřejmě může mít resolver uložen veřejné klíče více zón.

Pro vytvoření klíčů a podepsaní zóny není třeba zpracovávat každý záznam samostatně. Součástí balíčku *bind* jsou aplikace, kterým stačí zadat vhodné parametry, a ty zajistí vytvoření všech potřebných souborů. Pak již jen stačí správně nakonfigurovat aplikaci `named`, aby tyto soubory používala.

10.1 Klíče

Není to nezbytně nutné, ale obecně se doporučuje používat v DNSSec dva typy klíčů:

- Key Signing Key (KSK) a
- Zone Signing Key (ZSK).

Ve své podstatě se tyto klíče liší jen v jediném bitu (SEP – Secure Entry Point), který říká, že od tohoto místa může byt zahájeno sestavovaní důvěryhodného řetězce. V praxi se tyto dva klíče liší ještě další vlastností, která se určuje při jejích vytváření – délkou. Důvodem je vztah mezi délkou klíče, bezpečností a výpočetními nároky. Delší klíč je sice bezpečnější, ale zároveň má vyšší nároky na výkon a naopak. Protože k překladu DNS dochází velice často, není žádoucí, aby tato operace byla náročná na výpočetní zdroje. Proto byl zvolen postup, kdy klíč, který je často používán, je sice slabší, ale na druhou stranu jej lze měnit relativně často bez větších obtíží, a je-li čas nutný k jeho prolomení

hrubou sílou delší, než délka jeho platnosti, pak jeho prolomení nepředstavuje problém – tímto klíčem je klíč pro podepisování zóny (ZSK). Druhý klíč (KSK) se použije pouze pro podepsání ZSK, což znamená, že se nepoužívá tak často. V důsledku tento klíč může být delší a proto i odolnější vůči prolomení. Výměna tohoto klíče je obvykle trošku složitější, neboť vyžaduje spolupráci se správcem nadřazené domény. Čas, po kterém je nutné klíče obměnit, není explicitně definován a stejně tak klíče neobsahují časové vymezení platnosti. Po jaké době provést výměnu klíčů je tedy v režii správce domény a víceméně to závisí i na frekvenci používaní klíče k podpisu. Dochází-li v doméně k častým změnám a tedy i k častému podepisování, je vhodné měnit klíče častěji.

Pro vygenerování klíčů je v balíku *bind* k dispozici aplikace `dnssec-keygen`, které stačí jediný parametr[11] – název domény. Další užitečné parametry jsou:

`-f KSK` pokud má mít vygenerovaný klíč nastaven SEP bit,

`-b <číslo>` pro nastavení délky klíče,

`-a <algoritmus>` pro výběr algoritmu a

`-r` kterým se nastaví zdroj náhodných dat (standardně se použije `/dev/random`, který je blokující, pokud nemá dostatek entropie a může být vhodné nahradit ho `/dev/urandom`).

Pro získaní seznamu dalších možností stačí spustit bez parametru. Po spuštění aplikace se správnými parametry je vypsán na výstup řetězec, který se skládá z názvu domény, číselného kódu použitého algoritmu a identifikačního čísla klíče. Kromě toho vytvoří dva soubory, jejichž jména začínají vypsaným řetězcem a končí příponami `.key` a `.private`. První z nich obsahuje veřejný klíč a ten druhý privátní. Podle toho by se také s těmito klíči mělo zacházet (omezit přístup k privátnímu klíči, apod.). Příkazy:

```
dnssec-keygen -r /dev/urandom -a RSASHA1 -b 2048 -f KSK example.com
dnssec-keygen -r /dev/urandom -a RSASHA1 -b 1024 example.com
```

vytvoří čtyři soubory, kde první pár je KSK a druhý ZSK. Oba klíče je třeba zveřejnit v zóně. Vypsáním obsahu souborů s příponou `.key` zjistíte, že záznamy již mají tvar požadovaný pro zápis do zóny. Obsah těchto souboru se může buďto zkopírovat do zónového souboru a nebo je připojit pomocí direktivy `$INCLUDE`, např:

```
cat Kexample.com.+005+06487.key >> /var/named/example.com
cat Kexample.com.+005+32883.key >> /var/named/example.com
```

Alternativně přidejte do souboru `/var/named/example.com` řádek:

```
$INCLUDE Kexample.com.+005+06487.key;
$INCLUDE Kexample.com.+005+32883.key;
```

Nyní už jen stačí zónový soubor podepsat. To se provede příkazem `dnssec-signzone`, kterému se pouze předá parametrem název souboru, který obsahuje podepisovanou zónu. Pokud se název souboru neshoduje s názvem zóny, pak je třeba použít ještě parametr `-o`, kterým se určí název podepisované zóny. Pro podpis zóny se použijí automaticky ty klíče, jejichž veřejné části byly vloženy do zónového souboru (viz odstavec výše).

```
dnssec-signzone -o example.com labXX.zone
```

[11] Závisí na implementaci, v některých systémech může požadovat více parametrů.

Parametr `-o` specifikuje origin a `labXX.zone` je jméno zónového souboru (vstupního souboru). Výstupem tohoto příkazu je soubor se stejným názvem jako vstupní soubor rozšířený o příponu *.signed*. Pokud je požadován jiný název výstupního souboru, lze toho docílit volbou `-f`. Takto vzniklý soubor má požadovaný tvar zónového souboru. Stačí jen upravit `/etc/namedb/named.conf`, aby místo původního zónového souboru (bez podpisů) začal používat nově vzniklý soubor, který obsahuje i podpisy záznamů.

V souboru podepsané zóny se objevily nové typy záznamů `DNSKEY`, `RRSIG` a `NSEC`. První z nich je typ, který nese data o veřejné části klíče. `RRSIG` obsahuje samotný podpis plus další informace – typ podepsaného záznamu, jeho TTL, časy od kdy a do kdy je podpis platný a ID klíče, kterým byl podpis proveden. Poslední ze záznamu (`NSEC`) obsahuje informaci o následujícím záznamu v abecedně seřazené zóně. Tento záznam se využívá při odpovědi pro dotaz na neexistující záznam. Protože součásti podpisu je i časové vymezeni jeho platnosti, je nutné provádět znovupodepsání zóny. To by se mělo provést dříve, než vyprší platnost starých záznamů (nutné počítat se zkrácením hodnotou TTL, po kterou mohou být staré záznamy uloženy v cache paměti).

V tuto chvíli je již zóna podepsána a je možné ji začít používat. Ovšem resolver, který bude ověřovat, že je zóna dobře podepsaná, nemá k dispozici veřejný klíč (z jiného bezpečného zdroje), a nadřazená zóna neobsahuje informaci o použitém klíči a není proto možné sestavit důvěryhodný řetězec. Zbývá požádat správce nadřazené zóny o zveřejnění `DS` záznamu. Tento DS záznam se vytváří pro KSK klíč. Při podpisu zóny vznikne také soubor začínající řetězcem *dsset-* a končící názvem zóny. Tento soubor obsahuje `DS` záznamy, které je potřeba zveřejnit. Tyto záznamy lze také získat příkazem `dnssec-dsfromkey`. Pokud se nepoužijí žádné volby, pak tento program standardně vypíše na výstup dva DS záznamy ve stejné formě, jako je ve výše uvedeném souboru (jeden s použitím algoritmu SHA-1 a druhý s SHA-256). První z nich by měl být zveřejněn povinně.

```
dnssec-dsfromkey <název souboru s KSK klíčem>
```

Předání DS záznamů ke správci nadřazené domény se běžně provádí "out-of-band". Samotné záznamy jsou tedy přidávány ručně. Tento proces je nutné provádět i v případech výměny klíčů, což může vést k chybám při přidání nového klíče. Proces lze však automatizovat pomocí `CDS` a `CDNSKEY` záznamů [10], [8]. Tyto záznamy obsahují stejná data jako příslušné `DS`/`DNSKEY` záznamy, avšak je možné tato data předat standardní důvěryhodnou cestou. Servery nadřazené domény tyto záznamy využijí k synchronizaci uložených DS/DNSKEY dat. Poté je možné CDS a CDNSKEY záznamy odebrat.

Výměna klíčů

Jak bylo uvedeno výše, používají se obvykle dva typy klíčů, kdy jeden je slabší a druhy silnější. Oba tyto klíče je třeba obměňovat, aby nedošlo ke kompromitování zóny. Slabší z klíčů, ZSK, je třeba měnit častěji, ale vzhledem k tomu, že se změnou tohoto klíče není svázaná nutnost změny v nadřazené zóně, je tento krok relativně jednodušší (alespoň po administrativní stránce).

Při vyměně klíčů je nutné zajistit, aby klienti měli vždy k dispozici klíč, kterým jsou data podepsána. Způsoby, jak toho dosáhnout jdou dva – buďto podepsat data starým i novým klíčem, nebo nejprve zveřejnit nový klíč, počkat až se klíč rozšíří, a pak jej začít používat k podepisování zóny. Varianta, kdy se data podepíší starým i novým klíčem, je sice rychlejší, ale rovněž znamená, že zónový soubor zvětší svou velikost (a to téměř dvojnásobně). Z tohoto důvodu se tento způsob používá pouze pro výměnu KSK klíčů. Druhý způsob vyžaduje více časů, ale předchází zvětšování zónového souboru – běžně se používá pro výměnu ZSK.

10.2 Ověřování podepsané zóny

Bind od verze 9.5 má validaci zapnutou implicitně. Liší se ovšem způsob zpracování veřejného klíče. V současné době již je podepsána zóna `root`, a není proto nezbytně nutné zabývat se distribucí klíčů pomocí DLV registrů.