České vysoké učení technické v Praze

Fakulta strojního inženýrství

Ú 12110 Ústav přístrojové a řídící techniky

Vytvoření návodu pro instalaci lokální počítačové sítě

Bakalářská diplomová práce

Karel Landa

2009

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně s tím, že její výsledky mohou být dále použity podle uvážení vedoucího diplomové práce jako jejího spoluautora. Souhlasím také s případnou publikací výsledků diplomové práce nebo její podstatné části, pokud budu uveden jako její spoluautor.

Dne 26. 6. 2009

Podpis

Poděkování

Tato práce by nevznikla bez pomoci a podnětů vedoucího bakalářské diplomové práce Ing. Václava Hlaváče. Taktéž chci poděkovat Bc. Ondřeji Caletkovi a Ing. Radce Pernicové za cenné rady poskytnuté v nejtěžších chvílích zpracování bakalářské diplomové práce a Mgr. Monice Kipeťové za pomoc při opravě pravopisných a stylistických chyb.

Dále bych rád poděkoval své rodině a přátelům za morální podporu v době studia na FS ČVUT v Praze.

Vysoká škola: ČVUT v Praze
Pracoviště:
Ústav přístrojové a řídicí techniky

Fakulta: strojní

Školní rok: 2008/2009

ZADÁNÍ BAKALÁŘSKÉ DIPLOMOVÉ PRÁCE

pro: **Karla LANDU**

obor: Informační a automatizační technika

Název tématu: **Vytvoření návodu pro instalaci lokální počítačové sítě**

Zásady pro vypracování:

1. *Seznamte se s problematikou oddělení a zabezpečení lokální počítačové sítě od internetu a jejími principy.*
2. *Popište jednotlivé použité služby a jejich realizaci na serveru s operačním systémem Linux.*
3. *Nainstalujte a nakonfigurujte jednoduchou síť připojenou k internetu přes firewall.*
   Na jednom z počítačů oddělené sítě nainstalujte lokální http server a připojte k němu USB kameru.
4. *Vytvořte návod na instalaci a konfiguraci jednotlivých použitých služeb.*

Rozsah grafických prací: Dle pokynů vedoucího DP.

Rozsah průvodní zprávy: Dle pokynů vedoucího DP.

Seznam odborné literatury: *http://www.ietf.org/rfc.html*
http://www.w3.org/

Vedoucí diplomové práce: Ing. Vladimír Hlaváč

Recenzent diplomové práce: Mgr. Peter Sabolčák

Datum zadání diplomové práce: 15. 5. 2009

Termín odevzdání diplomové práce: 24. 6. 2009

Vedoucí ústavu: doc. Ing. Jan Chyský, CSc. *Děkan:* prof. Ing. František Hrdlička, CSc.

V Praze dne : 14. 5. 2009

Anotace

Cílem této práce je vytvoření výukového materiálu, který bude použit jako výuková pomůcka při výuce předmětu zabývající se instalací počítačové sítě a nastavením Linux serveru. Součástí práce je seznámení s prvky ovládání příkazové řádky i vybranými základními příkazy. Další část je věnována samotné instalaci překladu adres (NAT) a s tím úzce související služby DHCP pro automatické přidělování IP adres. Důležitou kapitolou je i nastavení zabezpečení serveru pomocí iptables. Nebylo zapomenuto ani na web server a jeho praktické použití za pomocí kamerky, kdy výstup z kamerky je publikován na internetových stránkách pomocí web serveru.

Klíčová slova

[Linux, DHCP, NAT, iptables, firewall, Apache]

Abstract

The main objective of this paper is focus on creation of education tutorial to be used as teaching aids in subject at university. Subject deal with installation of computer network and Linux´s server setting. One of significant part of this work is familiarization with controls command prompt and selected basic commands. Another part is focus on installation network address translation (NAT) itself and the closely related services DHCP for automatic assignment of IP addresses. An important chapter is the security settings of the server by means of iptables. At the end of my theses is also shown a web server and its practical application for camera, when the output of camera is published on the website through the web server.

Key word

[Linux, DHCP, NAT, iptables, firewall, Apache]

Obsah

1 Úvod

Tato bakalářská práce řeší softwarové nastavení serveru a počítačové sítě od úplných základů, tj. instalace serveru a nastavení základních služeb, které na tomto serveru poběží. Nedílnou součástí řešení je také zabezpečení a základní princip ochrany proti různým útokům na server.

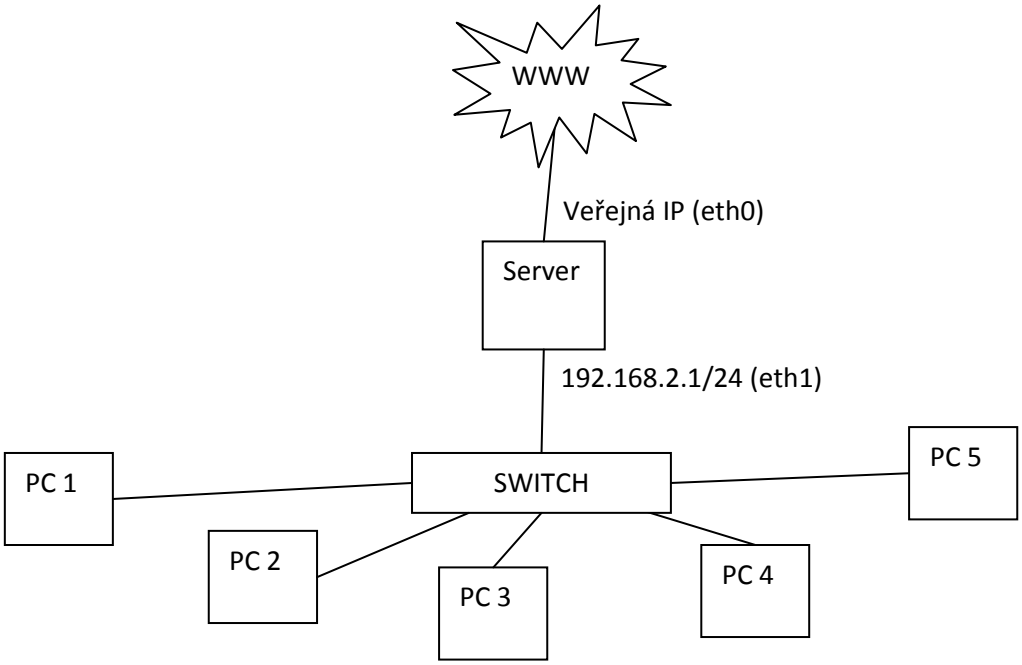
Cílem práce je vytvořit výukový materiál, pro použití při výuce předmětu zabývající se operačním systémem Linux a praktickými principy síťování. Přepokládají se základní znalosti sítí, které se probírají v předmětu Inženýrská informatika. Úkolem je seznámení studentů vysoké školy s alternativním operačním systémem. Tímto student získá přehled o základních principech práce s operačním systémem Linux a rozšíření tak svých obzorů v tomto oboru. Díky volnému použití tohoto operačního systému pod licencí GNU/GPL (General Public License) získají studenti nový legální prostředek ke své práci, což v době stále vysoké [12] softwarové kriminality a stále se zvyšujícími finančními nároky na software je velkou výhodou a přidanou hodnotou k dalšímu rozvoji ve vzdělání.

Nedílnou součástí je seznámení s příkazovou řádkou a s tím spojenými základními příkazy pro práci v tomto operačním systému. Obsahem je taktéž konfigurace důležitých a nejvíce využívaných funkcionalit serveru, především v oblasti zabezpečení, sdílení internetového připojení, konfigurace web serveru a s tím spojených nejen databázových služeb.

Zajímavostí je i konfigurace web kamery a následné publikování výstupu na webovém serveru. Tyto dnes moderní a často využívané technologie ukazují praktické využití získaných teoretických znalostí.

2 Síťová topologie

V této práci se vycházelo z aktuálně zadaného stavu sítě, který je znázorněn na obrázku 1. Základem aktuálního návrhu je tzv. stromová struktura, kde veškeré počítače jsou připojeny ke switchi. Na serveru jsou služby jako DHCP, web server, firewall, překlad adres (NAT) a možnost streamovat kamerku, která je připojena k jednomu z klientských počítačů.



Obrázek 1 – síťová topologie

Pro vnitřní síť byly navrženy IP adresy v rozsahu 192.168.2.0/24. Jedná se privátní rozsah adres dle RFC1918 [1]. Pro stolní počítače bylo vyhrazeno prvních 30 IP adres. Ostatní počítače a zařízení, které do sítě budou vstupovat „mimořádně“, budou dostávat automaticky adresy ze stejného rozsahu, avšak adresy vyšší, než 192.168.2.31.

Hlavním úkolem je tedy nastavení serveru a jeho služeb. Práce se zabývá pouze softwarovým řešením, a to operačního systému i jednotlivých služeb, které je nutné na server nainstalovat a svědomitě nastavit.

3 Výběr vhodného operačního systému

Jako první byl vybrán vhodný operační systém. Jelikož se předpokládá, že operační systémy, které se musí zakoupit, jsou obyčejným studentům jen těžko finančně dostupné, zvolil se operační systém GNU/Linux.

Operační systém GNU/Linux se skládá z jádra Linux, pomocných programů z projektu GNU dalších programů. Ty se společně nazývají distribuce GNU/Linuxu. Těchto distribucí je nepřeberné množství [3] od těch nejznámějších jako je např. Debian, Gentoo, SUSE, Ubuntu až po méně známé jako Openwall, Resulinux, Untangle. Budu se zabývat především distribucí Debian, která je jednou z nejrozšířenějších a na serverech nejvyužívanější distribucí na světě[4]. Používají ho i velké společnosti jako např. seznam.cz[5][6] a další.

Pro instalaci tohoto systému lze využít některou z instalačních příruček, které nalezneme v české jazykové mutaci například na oficiálních internetových stránkách [2] v sekci „instalační příručky“. Je potřeba vybrat si správnou verzi pro daný server. Naše vybrané verze jsou: 32-bitové počítače verze Intel x86, 64-bitové počítače verzi AMD64. V době vzniku této práce byla poslední známá stabilní verze Debian 5.0, proto se budu věnovat právě této verzi. Debian je systém, který obsahuje software s licencí GNU/GPL[7] a má vlastní systém správy balíčků „deb“. Jeho filosofií je stabilita. V Debianu především nalezneme software stabilní, řádně odladěný a ověřený uživateli. Z toho však plyne, že na nejnovější verze softwaru většinou v Debian Stable nenarazíme. Debian se rozděluje na tři základní vývojové verze: verze, která se distribuuje, se nazývá Stabilní, jako Nestabilní se označuje verze, která se neustále mění a testuje, a verze Experimentální, což znamená nejnovější verze programů s minimálním ohledem na kompatibilitu. Má velmi dobrou lokalizaci do českého jazyka.

Je důležité vědět, že operační systém Linux rozlišuje velká a malá písmena, proto je nutné dodržovat velikosti písmen.

4 Práva a příkazy v GNU/Linux

Pro lepší orientaci v prostředí Příkazové řádky je nutné znát určité základní příkazy a klávesové zkratky. Tyto zkratky nám mnohdy usnadní a zefektivní naši práci v prostředí Bash, které je převážně u operačního systému GNU/Linux základním prostředím.

Bash (Bourne again shell) je unixový shell naprogramovaný v rámci projektu GNU. Není to však pouze výborný příkazový řádek, jak by se na první pohled mohlo zdát, ale především plnohodnotný skriptovací jazyk. Mnoho programů, které budeme spouštět, jsou právě skripty příkazové řádky. Zde však budou vysvětleny především základní klávesové zkratky a příkazy pro práci v příkazovém řádku.

4.1 Základní klávesy

- *tab* : doplňuje příkazy, případně vypisuje příkazy, které lze použít
- *šipka nahoru*: vypisuje příkazy, které byly použity (historie příkazů)
- *ctrl+r* : vyhledávání v historii příkazů, následující příkazy vyhledává opětovným stiskem *ctrl+r*
- *shift+šipka nahoru*: roluje nahoru obrazovku o řádek nahoru
- *shift+PageUP* : roluje o obrazovku nahoru

4.2 Práva v GNU/LINUX

Linux je víceuživatelský systém a proto je nutné zajistit, aby ostatní nemohli měnit, číst či mazat osobní dokumenty, pokud jim to není danou osobou dovoleno. Každý soubor i adresář má přidělena práva, která upravují práci s daným souborem. Pro lepší orientaci a pochopení je nutné znát základní orientaci v přidělování práv a jejich význam.

Příkaz `ls` vypisuje obsah zadaného adresáře. Pokud přidáme parametr `-l` zobrazíme podrobnější informace, včetně práv:

```
      # ls
.bash_logout   profile   .bashrc   centericq
      # ls - l
-rwx-w--w- 1     charlie users    240   čen   28 12:53    .bash_logout
```

```
-rwxrw----  1     charlie users    308   čen   28 12:53    profile
-rwxr--r--  1     charlie users    1306  čec   15 02:06    .bashrc
drwx--x--x  170   charlie users    5184  čen   11 21:02    centericq
```

Práva jsou uvedená v prvním sloupci. První znak označuje typ souboru, pomlčka '**-**' označuje normální soubor, '**d**' adresář, dále '**l**' je symbolický odkaz, '**c**' znakové zařízení, '**b**' blokové zařízení a '**p**' je pojmenovaná roura. Za typem je devět znaků, které symbolizují práva. Vždy jsou vypsány v pořadí právo pro čtení '**r**', právo pro zápis '**w**' a právo pro spuštění '**x**'. Na úrovni adresářů '**r**' značí, že je možné vypsat seznam souborů, '**w**', že je možné do adresáře soubory přidávat nebo je z něj odebírat a '**x**', že adresář smí být součástí cesty. První trojice je nastavení platné pro vlastníka souboru, ten je uveden ve třetím sloupci a v našem případě je to 'charlie', druhá trojice pro skupinu users, kterou najdeme ve čtvrtém sloupci a poslední třetí trojice pro ostatní. Vlastník souboru je uživatel, který jej vytvořil nebo bylo na něj vlastnictví převedeno příkazem `chown`, a obvykle má největší pravomoc. Skupina usnadňuje sdílení souborů mezi uživateli. Práva může měnit pouze vlastník souboru a superuživatel root. Za těmito znaky práv je obvykle číslo, které nám značí počet linků (odkazů), které nám na daný soubor (adresář) ukazují. Pokud mají soubory před svým názvem tečku, je tento soubor skrytý. Kořenový adresář značíme na začátku lomítkem '/'. Aktuální adresář je označen jednou tečkou ' **.** ', nadřazený adresář dvěma tečkami '**..**' a domovský adresář vlnovou '**~**'.

Pro práci s právy se používají především následující příkazy:

`chmod` – tento příkaz slouží ke změně přístupových práv k souboru. Práva se udávají nejčastěji číslem, kdy první číslice udává práva vlastníka, druhá práva skupiny a třetí práva ostatním. Operace spuštění souboru přispívá do celkového součtu vahou 1, zápis vahou 2 a čtení vahou 4. Tím vznikají různé kombinace přístupových práv:

hodnota	0	1	2	3	4	5	6	7
práva	`---`	`--x`	`-w-`	`-wx`	`r--`	`r-x`	`rw-`	`rwx`

Tabulka 1 – práva příkazu chmod

`chown` – příkaz sloužící ke změně vlastníka nebo přiřazení ke skupině. Tento příkaz má následující syntaxi: `chown` [*vlastnik*]:[*skupina*] [*měněný soubor*]

4.3 Adresářová struktura

Při bližším pohledu na adresářovou strukturu je zde vidět několik adresářů, z nichž každý má svůj význam.

`/` - kořen souborového systému, začátek stromové struktury

`/bin` - základní spustitelné soubory pro použití všemi uživateli

`/boot` - zde je umístěno jádro (kernel) systému a jeho mapa, initrd, zavaděč (boot loader) GRUB

`/dev` - soubory v tomto adresáři reprezentují jednotlivá zařízení systému.

`/etc` - globální konfigurační soubory systému a dalších programů

`/home` - domovské adresáře uživatelů

`/lib` - základní sdílené knihovny systému, mapování klávesnice a konzolové fonty, moduly pro jádro systému (kernel)

`/mnt` – do tohoto adresáře se většinou mapují souborová zařízení

`/opt` - zde bývají SW aplikace, které nejsou standardní součástí distribuce

`/root` - domovský adresář superuživatele (root)

`/sbin` - systémové privilegované spustitelné soubory, používané uživatelem root

`/tmp` - adresář pro odkládací a pomocné soubory

`/usr` - obsahuje velké množství informací, jako knihovny, zdrojové kódy, spustitelné soubory, konfigurační soubory a další

`/var` - soubory, jejichž obsah se během chodu systému většinou mění, obsahuje též logy

4.4 Základní příkazy

Pro práci se systémem je žádoucí znát určité základní příkazy a principy, které práci v příkazové řádce zefektivní. Pro názornost bylo vybráno několik příkazů a jejich částečný popis. Kompletní manuálový popis příkazů a jejich parametrů je možné získat příkazem `man` [*název příkazu*].

`mkdir` – tento příkaz vytvoří adresář (např. „`mkdir /mnt/pokus`“ vytvoří adresář pokus v adresáři `mnt`)

`rmdir` – tento příkaz vymaže adresář, adresář musí být ale prázdný (např. `rmdir /mnt/pokus`)

`rm` – tento příkaz vymaže daný souboru

parametry:

`-r` (recursive – Bude mazat zadané adresáře i s jejich obsahem)

`-i` (interactive) - Bude vyžadovat potvrzení před smazáním jednotlivých souborů.

`df` – vypíše velikost místa na disku

parametry:

`-m` (velikost místa v MB)

`-h` (velikost místa bude vypsána v tzv. inteligentním módu v jednotkách Kb, Mb a Gb)

`cp` – kopíruje soubor (první se zadává soubor, který chci kopírovat, potom cestu kam má být soubor nakopírován)

`cp` [*soubor*] [*kam ho kopírovat*]

`nano` – textový editor pro úpravu souborů (`nano` [*soubor*])

`vim` – textový editor pro úpravu (`vim` [*soubor*]) souborů s mnoha funkcemi. Uvedu zde některé základní:

- *i* – režim pro zapisování
- *y y* – kopírování řádku
- *p* – vložení řádku
- *esc esc : q !* – ukončení programu vim bez uložení
- *esc esc : x* – ukončení programu vim s uložením souboru
- *:help* – vypsání dalších specifikací

`mount` – příkaz pro tzv. připojení disků. Př.: `mount /dev/hda1 /mnt/d` připojí svazek `/dev/hda1` do adresáře `/mnt/d`. Svazky vypíši příkazem `cat /proc/partition`

Abychom nemuseli vždy při spuštění systému připojovat ručně disky, nastavíme si toto do souboru `/etc/fstab`. Tím se bude připojovat při spuštění automaticky.

`umount` – příkaz odpojí daný svazek (`umount` [*adresář*])

Soubor `/etc/fstab` je textový soubor. Jedná se o databázi dostupných svazků a přípojných míst v adresářové struktuře, do kterých se dané svazky připojí. Jako do všech důležitých souborů má i do něj pro zápis přístup pouze superuživatel *root*. *Fstab* může popisovat cokoliv, co lze připojit příkazem `mount`. Příklad: Chceme připojit disketu do adresáře `/mnt/floppy`. První disketová jednotka je označená v systému jako fd0. Pro připojení tohoto zařízení se použije příkaz:

```
# mount /dev/fd0 /mnt/floppy -t vfat
```

Tímto příkazem dojde k připojení zařízení `/dev/fd0` do adresáře `/mnt/floppy` a že je na něm souborový systém typu *vfat*. Syntaxe *fstabu* je prostá: Co řádek, to

svazek. Na každém řádku je několik položek oddělených standardními oddělovači, mezery nebo tabulátory. Pořadí je podstatné a nezaměnitelné a musí se dodržovat. Příklad takového řádku:

```
   1            2                  3       4             5     6

/dev/fd0    /mnt/floppy            vfat  noauto,user      0     0
```

- `1` zařízení, které budeme připojovat.

Může ji tvořit jak běžný oddíl disku `/dev/hda2`, tak i označení vzdáleného svazku, třeba přes SMB `//pokoj/pokus`.

- `2` připojovací bod (adresář), do kterého se svazek připojí

Adresář připojení. Obvykle je to některý z podadresářů `/mnt`. Pro swap se zadává `none`, protože ten se nikam nepřipojuje.

- `3` typ souborového systému

Kompletní seznam souborových systémů, které podporuje vaše jádro naleznete v `/proc/filesystems`. Jejich množství a typ záleží na verzi jádra a zavedených modulech. Mohou to být: `proc`, `tmpfs`, `ext2`, `ramfs`, `iso9660`, `devpts`, `ext3`, `usbdevfs`, `usbfs` a další. Můžeme také použít výraz auto a jádro se pokusí obsah samo detekovat.

- `4` parametry pro připojení

Do té je možno vyplnit celou řadu různých parametrů pro různé souborové systémy. Parametry se od sebe oddělují čárkou. Mezi hlavní patří

`noauto` - nepřipojuje svazek automaticky při startu

`users` - s tímto svazkem mohou pracovat i běžní uživatelé

`codepage` - znaková sada, ve které jsou názvy souborů

`iocharset` - znaková sada, do které se budou převádět názvy souborů

`noexec` - nespouštěj soubory na tomto médiu

`umask` - nastavení práv u souborů

`ro` - read only - pouze ke čtení

`rw` - read write - čtení i zápis

Kompletní popis naleznete v manuálové stránce k *mount*.

- `5` určuje, jestli bude svazek zálohován

Tuto položku používá program *dump*, který podle ní pozná, zda má provádět zálohu svazku. Jednička znamená zálohovat, nula nezálohovat.

- `6` nastavuje, jako kolikátý bude svazek kontrolován při startu

Poslední položku používá program `fsck` aby zjistil, jako kolikátý bude svazek kontrolován. Jednička označuje kořenový svazek, který se připojuje do ' / ', dvojka pak ostatní a nula ty, které se kontrolovat nebudou. Každý řádek, tedy i poslední, musí být ukončen enterem tj. odřádkován. Podle něj pak programy poznají, že informace o svazku končí. Častou chybou je nezapsání enteru na konec posledního řádku. Pokus o připojení pak končí chybou. Oddíly označené jako *swap*, jsou připojovány při startu systému pomocí příkazu `swapon -a`. Stejně tak je lze odpojit pomocí `swapoff -a`.

5 Nastavení sítě a routování

5.1 Nastavení sítě

Předpokládá se, že v našem serveru se nachází dvě síťové karty. Jedna z těchto síťových karet bude sloužit pro komunikaci do/z internetu, druhá pro komunikaci s vnitřní sítí. Pro aktuální zobrazení IP adres na interfacech *eth0* a *eth1* použijeme příkaz, který pro daný parametr `-a` vypíše veškerá síťová rozhraní:

```
# ifconfig -a
```

Výpis by měl vypadat např. nějak takto:

```
eth0   Link encap:Ethernet  HWaddr 00:50:56:00:00:10
       inet addr:147.32.30.144  Bcast:147.32.30.255  Mask:255.255.255.128
       UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
       RX packets:215458 errors:0 dropped:0 overruns:0 frame:0
       TX packets:217758 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:1000
       RX bytes:22707316 (21.6 MiB)  TX bytes:37307449 (35.5 MiB)
       Interrupt:18 Base address:0x2000

eth1   Link encap:Ethernet  HWaddr 00:50:56:00:00:11
       inet addr:192.168.2.1  Bcast: 192.168.2.255  Mask:255.255.255.0
       UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
       RX packets:234554 errors:0 dropped:0 overruns:0 frame:0
       TX packets:233754 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:1000
       RX bytes:25204216 (25.6 MiB)  TX bytes:33307449 (31.5 MiB)
       Interrupt:18 Base address:0x2000

lo        Link encap:Local Loopback
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:62 errors:0 dropped:0 overruns:0 frame:0
          TX packets:62 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:6572 (6.4 KiB)  TX bytes:6572 (6.4 KiB)
```

Z tohoto výpisu se dozvídáme několik důležitých informací. MAC adresu, aktuální IPv4 adresu i případné IPv6 adresy, masku, rozsah a mnoho dalších informací. Nejdůležitější v této chvíli je nastavení IP adresy, masky, brány i DNS. V souboru `/etc/network/interfaces` je možné editovat nastavení síťového rozhraní s následující syntaxí:

```
auto eth0

  iface eth0 inet dhcp             / přijímat informace z DHCP


iface eth1 inet static       /statické nastavení daného interface
```

```
address 192.168.2.1           / IP adresa daného interface
netmask 255.255.255.0         / maska
broadcast 192.168.2.255  / broadcastová adresa
dns-server 147.32.127.240     / DNS server
```

U této konfigurace se předpokládá, že *eth1* vede do vnitřní sítě a *eth*0 vede do internetu. A dále, že na *eth0* je nastaveno získávání IP adresy z DHCP. Na server však není vhodné nastavovat získávání IP adres z DHCP kvůli možným komplikacím při přidělování IP adres serveru. U serveru je nutné mít stále stejnou IP adresu, jelikož se na ni často odkazujeme. Proto se důrazně doporučuje nastavit na obě rozhraní statické adresy.

5.2 Směrování

Směrování neboli routing se používá k propojení jednotlivých sítí nebo spíše přesněji podsítí (subnetů). Podle toho, zda routery přejímají informace od ostatních sousedních routerů nebo od administrátora, dělíme routování na dva typy: dynamické nebo statické routování.

U dynamického routování je routovací tabulka nastavována automaticky podle toho, jaké informace dostává od ostatních routerů v síti. Tento způsob je vhodný u rozsáhlých sítí, kde tak můžeme „obejít“ případný výpadek některého z routerů. Při výpadku tak dojde k aktualizaci routovací tabulky a pakety mohou putovat jinou cestou.

Statické routování je založeno na manuálním nastavení záznamů v routovací tabulce administrátorem. Toto nastavení je vhodné u menších sítí.

Po příchodu paketu se jeho cílová adresa porovná s údaji v routovací tabulce a vybere se ten záznam, který adrese nejlépe odpovídá. Na tuto adresu je poté paket odeslán. V našem případě bude probíhat routování mezi naší vnitřní sítí 192.168.2.1/24 a vnější veřejnou sítí. K tomu postačí statická směrovací tabulka, která se automaticky vytvoří nakonfigurováním IP adres na příslušných síťových rozhraních.

Pro zobrazení aktuální routovací tabulky použijeme příkaz

```
# route –n
```

Parametr `-n` zobrazuje pouze IP adresy bez doménových jmenných záznamů.

6 SSH

Server je počítač, který se instaluje lokálně, ale pro rychlou a efektivní správu potřebuje také bezpečný vzdálený přístup. Správa serveru s sebou nese určitá rizika. S tím jsou spojené i požadavky na bezpečnost pro komunikaci mezi námi a serverem. Nejčastěji se pro tuto komunikaci využívá SSH - *Secure Shell* [16] neboli zabezpečený vzdálený přístup. Tento program byl vytvořen jako náhrada za *telnet* a další nebezpečné vzdálené *shelly*, které posílají heslo v nezabezpečené textové formě, čímž může dojít k vyzrazení hesla. SSH šifruje nejen počáteční autentizaci, ale i veškerou komunikaci se serverem. Pro šifrování je možné vytvořit vlastní klíče různých délek. Doporučuje se minimálně 1024-bitové šifrování, aby vzájemná komunikace byla bezpečná. SSH by měl být program, který se na server instaluje jako jeden z prvních a dále je možno pracovat vzdáleně. Program lze nainstalovat následujícím příkazem:

```
# aptitude install openssh-server
```

Po instalaci je možné k našemu server vzdáleně přistupovat z operačního systému Windows např. programem putty. Který je volně ke stažení na internetových stránkách[8]. Pokud se přihlašuje z operačního systímu GNU/Linux, je nutné použít příkaz:

```
# ssh [uživatel]@[adresa serveru]
```

6.1 Autentizace pomocí klíče

Jednou z metod pro autentizaci je použití veřejného klíče. Nejprve je nutné vygenerovat šifrovací klíče, soukromý (privátní) a veřejný klíč. Veřejný klíč můžeme volně zveřejňovat. Pomocí tohoto klíče je možné zprávu či komunikaci zašifrovat. Nelze však tuto komunikaci tímto klíčem rozšifrovat. Pro rozšifrování je potřeba klíče privátního. Přihlášení pak funguje způsobem, že veřejný klíč máme uložen na serveru, na který se chceme přihlásit. Tímto veřejným klíčem server zašifruje určitá náhodná data tzv. výzvu, a tu pošle klientovi. Klient tato data dešifruje a dešifrovanou ji zašle serveru. Pokud se data shodují, má server ověřeno, že klient má k dispozici privátní klíč, který odpovídá veřejnému klíči, který má server. Server poté přístup schválí. V opačném případě je výzva o komunikaci zamítnuta.

Pro vygenerování klíčů pod operačním systémem Windows můžeme využít např. program *puttygen*, který nalzeneme stejně jako program *putty* na internetových stránkách [8]. Jeho ovládání je velice jednoduché a intuitivní. V případě, že využijeme pro generování klíčů systém Linux, je nutné znát příkaz. Klíče vygenerujeme příkazem:

```
# ssh-keygen -t rsa -b 2048
```

Kde `-t rsa` říká, jaký je použit šifrovací algoritmus, zda RSA nebo DSA , parametr `-b`, jak silný má klíč být, v našem případě 2048bitů. Po zadání tohoto příkazu se systém ptá, kam má klíče uložit a pravděpodobně nabídne standardní úložiště `/root/.ssh/id_rsa`. Stačí tedy pouze potvrdit umístění uživatelem. Poté se systém zeptá na heslo k těmto klíčům tzv. *passphrase.* Pokud chce mít administrátor vysokou bezpečnost, je rozumné heslo zvolit. Po potvrzení hesla jsou klíče vygenerovány. Klíče jsou uloženy v adresáři `.ssh`, případě v jiném adresáři, který jsme si zvolili. Privátní klíč se nazývá `id_rsa` a veřejný klíč `id_rsa.pub`.

6.2 Ukládání klíče

Při používání klíče je nutné veřejný klíč bezpečnou cestou dopravit na server do adresáře .ssh a souboru `authorized_keys`. Do tohoto souboru je možné uložit více klíčů s důrazem na dodržení nutné zásady, že na jeden řádek je možné uložit pouze jeden klíč. Po úspěšném uložení klíče si může uživatel vyzkoušet přihlášení.

Je nutné, aby právo do souboru `authorized_keys` měl pouze majitel souboru. V opačném případě nebude ssh server tento soubor akceptovat.

7 Nastavení času

Pro správnou orientaci v systému a v záznamech, je nutné mít na serveru přesný čas. Přesným časem si zajistíme i přesné informace potřebné pro případné další analýzy. Jako první je nutné nastavit odpovídající časové pásmo a to příkazem:

```
# dpkg-reconfigure tzdata
```

Po spuštění tohoto příkazu se ukáže nabídka, kde je možnost vybrat časové pásmo *Europe*, po stisknutí enter i město Prague. Tím se nastavila správná časová zóna. To však neznamená, že správně nastaven. Pro tento případ existuje protokol NTP (network time protokol) [14]. Jedná se o protokol, kterým se hlídá a synchronizuje čas dle časových serverů, kdy chyba synchronizace se pohybuje pod 10 milisekund. Počítač, který chce být synchronizován, odešle několika serverům dotazy, které mu odešlou přesný čas. Po přijetí časů klient nejprve vyřadí servery se zřejmě nesmyslným časem a ponechá skupinu serverů s největším společným průnikem. S těmito časy se poté náš server sesynchronizuje. Program nainstalujeme příkazem

```
# aptitude install ntp
```

Po instalaci je rozumné přidat místní časové servery do konfiguračního souboru:

```
# vim /etc/ntp.conf
```

Do konfiguračního souboru přidáme následující místní časové servery:

```
server pool.netp.org iburst dynamic
server tak.cesnet.cz iburst dynamic
server time.sh.cvut.cz iburst dynamic
```

Nyní již stačí NTP restartovat, aby se načetla aktuální konfigurace:

```
# /etc/init.d/ntp restart
```

Tímto se zajistí přesný čas v našem PC, který překontrolujeme příkazem `date`.

8 DHCP (Dynamic Host Configuration Protocol)

DHCP [13] umožňuje pomocí DHCP serveru předávat jednotlivým klientským stanicím informace potřebné pro komunikaci v síti. Můžeme tak centralizovat i spravovat větší počítačové sítě. DHCP může zasílat několik parametrů. Pokud klient některým parametrům nerozumí, ignoruje je. Většinou jsou předávány parametry jako: IP adresa, maska sítě, brána (gateway), DNS servery a případně další údaje jako např. WINS, NTP, atd.… V závislosti na provedení DHCP serveru může DHCP přidělovat IP adresy třemi různými metodami:

• Dynamické přidělování:

Administrátor nastaví rozsah přidělovaných adres, ze kterého klienti dostávají IP adresy. Každý klient, který požádá o IP adresu, tuto adresu dostane na určitou dobu.

• Automatické přidělování:

Podobný princip fungování jako u dynamického přidělování, ale zde se vede tabulka přidělovaných adres. Při znovu připojení klienta dostane klient naprosto stejnou adresu, jako dostal poprvé.

• Statické přidělování:

DHCP server přiděluje adresy dle jasně definované dvojice MAC adresa a IP adresa. Tuto dvojici musí vytvořit sám administrátor. DHCP přidělí adresu pouze těm klientům, kteří mají svou MAC adresu v tabulce. Ostatním klientům IP adresa přidělena nebude.

Pozn.: Je možné využívat též kombinace jednotlivých metod (např. statického přidělování a Automatického přidělování)

8.1 Princip fungování

Klienti komunikují na portu UDP 68 a server naslouchá na UDP portu 67. Komunikace probíhá následujícím způsobem: Klient vyšle požadavek – (discover) na broadcastovou adresu 255.255.255.255. Vzhledem k tomu, že se jedná o broadcast, je tento požadavek distribuován po celém segmentu sítě, tj. přijde tento požadavek každému počítači na segmentu. Server tomuto klientovi odpoví (offer) a nabídne mu volnou IP adresu, pokud je k dispozici, případě že je na segmentu více DHCP serverů, dostane klient nabídku volné IP adresy od každého serveru. Klient si z dané nabídky vybere (request) adresu a server mu ji potvrdí (odpoví ack).

8.2 Konfigurace DHCP

Dnes se nejčastěji využívá program DHCPd, který nainstalujeme v OS Linux, distribuci Debian příkazem:

```
# aptitude install dhcp3-server
```

Tím je program nainstalován. Po instalaci je však nutné zkontrolovat, zda máme v jádru zakompilovanou podporu pro multicást adresy. To zjistíme příkazem

```
# ifconfig –a
```

Pokud ve výpisu uvidíme slovo *broadcast*, na systému v české mutaci *všesměrové vysílání*, je vše v naprostém pořádku.

Po krátké instalaci lze nalézt veškeré konfigurační soubory v adresáři, který se edituje:

```
# vim /etc/dhcp3/dhcpd.conf
```

Výpis souboru *dhcpd.conf* , kde za lomítkem jsou komentáře k jednotlivým řádkům:

```
option domain-name „fs.cvut.cz“; / název domény

option routers 192.168.2.254 192.168.2.253;
option domain-name-servers 192.168.1.1, 192.168.1.2; / DNS servery
option subnet-mask 255.255.255.0; / maska sítě
option broadcast-address 192.168.2.255;
default-lease-time 3600; / čas propůjčení adresy v sekundách
max-lease-time 7200; / maximální čas propůjčení adresy v sekundách

subnet 192.168.2.0 netmask 255.255.255.0 { / rozsah, ze kterého se mají
adresy přidělovat současně s maskou
range 192.168.2.31 192.168.1.254; / rozsah pro přidělování adres

 }

host PC1 { /název počítače, kterému chceme přidělit statickou adresu
hardware ethernet 08:00:4b:2c:22:23; / MAC adresa počítače
fixed-address 192.168.2.2; / IP adresa počítače

 }
```

Po prvním startu se vytvoří soubor `/var/lib/dhcp/dhcpd.leases`, kde si ukládá server informace o poskytnutých adresách i MAC adresách klientů. Jedná se o obyčejný textový soubor, proto, jej lze bez problému prohlížet. Po dokončení konfiguračního souboru je nutné DHCP server restartovat, aby se projevila nová konfigurace:

```
# /etc/init.d/dhcp3-server restart
```

Aby tento DHCP server byl bezpečný, je nutné zabezpečit, aby server naslouchal pouze požadavkům z vnitřní sítě. Nikdy z Internetu! V opačném případě, se

vystavujeme nebezpečí *DoS* útoků, což v lepším případě může vést k znefunkčnění DHCP.

9 NAT

NAT neboli *Network Address Translation* [15] je technika vyvinutá především kvůli omezené velikosti IPv4 adresního prostoru. Jeho princip spočívá ve schování určitých privátních IP adres za adresu veřejnou, pomocí které se pak počítače z lokální sítě prokazují za jejími hranicemi. Nemalou výhodou NATování je vyšší bezpečnost zařízení za NATem. Rozlišujeme DNAT, SNAT a speciální případ SNATu – *Masquerade*.

Pro vytváření NATu i v následující kapitole firewallu zde bude použit program iptables. Jedná se o velice mocný nástroj, který umožňuje systému plně pracovat se síťovou komunikací. Pomocí něj je možné vytvořit firewall, překlad adres (NAT), sledovat provoz i provoz na síti řídit.

Na následujícím obrázku 2. je vidět, jak procházejí pakety skrz jádro. Za zmínku stojí zejména řetězce PREROUTING a POSTROUTING, neboť ty jsou nyní nejpodstatnější. Všimněme si také řetězce FORWARD, kde pakety také protékají, proto nezapomeňme na povolení předávání paketů mezi síťovými rozhraními:

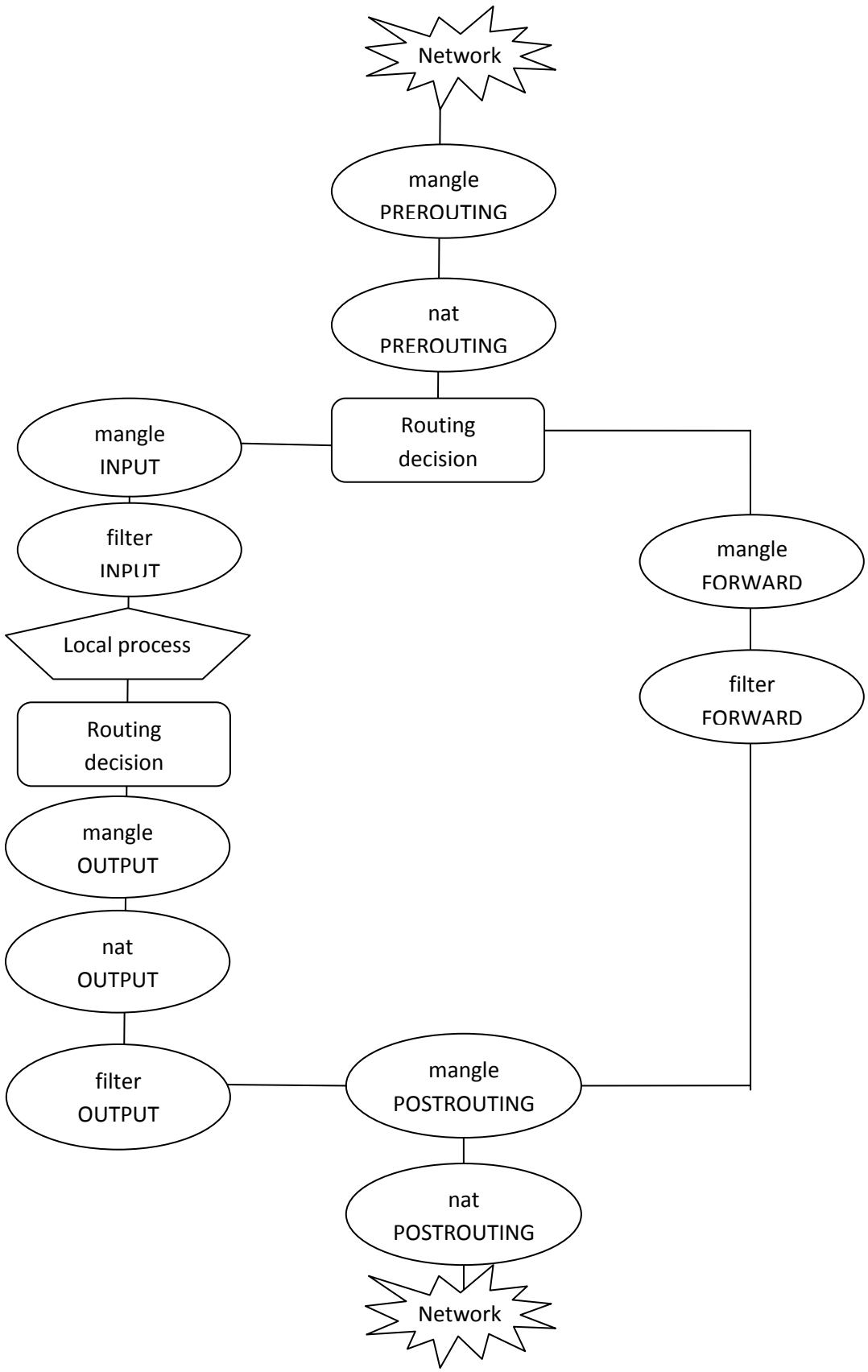
```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Pro povolení předávání paketů mezi síťovými rozhraními i po restartu PC, je nutné v souboru `sysctl.conf` provést následující příkaz:

```
# vim /etc/sysctl.conf
```

Odkomentovat řádku znamená vymazat pouze ' `#` ' před tímto řádkem:

```
net.ipv4.ip_forward=1
```



Obrázek 2 – cesta paketu z tutoriálu Andreassona

9.1 SNAT

SNAT neboli Source NAT je technika, při které se mění zdrojová (source) IP adresa. Máme-li např. PC s adresou 192.168.1.5 za routerem, který dělá SNAT na adresu 147.32.105.30, pak přistupuje-li se do internetu, prokazuje se právě adresou 147.32.105.30. Důležitá informace pro sestavování pravidel je, že SNAT se provádí po routování. Samotné nastavení SNATu je jednoduché, pro názornost jsou zde uvedeny tři příklady:

Změna adresy na jednu konkrétní:

```
# iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 1.2.3.4
```

Změna adresy na jednu z rozsahu:

```
# iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 1.2.3.4-1.2.3.9
```

Změna adresy a portu:

```
# iptables -t nat -A POSTROUTING -p tcp -o eth0 -j SNAT --to 1.2.3.4:1-
1023
```

9.2 Masquerade neboli maškaráda

Maškaráda je zvláštní a zároveň asi nejpoužívanější případ SNATu. Všechny zařízení za NATem se schovávají za jednu jedinou IP adresu, které je přiřazena routeru. Tento případ NATu se nejčastěji používá, pokud si např. klient bude chtít doma připojit do sítě našeho providera více počítačů. Jeho hlavní výhoda je použití dynamicky přiřazené adresy, ovšem používá se i při pevně přidělené adrese. Nastavení maškarády je jednoduché:

```
# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Pozn.: Masquarade pokud možno nepoužíváme, jelikož se špatně kombinuje např. s DNAT.

9.3 DNAT

DNAT neboli Destination NAT je naopak technika, při níž se mění cílová (destination) adresa IP zařízení. Můžete tak realizovat port forwarding, transparentní proxování atd. Důležitá je jasná připomínka, že DNAT se provádí před routováním. Několik příkladů DNATu:

Změna cílové adresy na jinou:

```
# iptables -t nat -A PREROUTING -i eth0 -j DNAT --to 5.6.7.8
```

Změna cílové adresy na jinou ze zadaného rozsahu:

```
# iptables -t nat -A PREROUTING -i eth0 -j DNAT --to 5.6.7.8-5.6.7.10
```

Přesměrování všeho co přijde na port 80 na jiný stroj na port 8080:

```
# iptables -t nat -A PREROUTING -p tcp --dport 80 -i eth0 -j DNAT --to
5.6.7.8:8080
```

Přesměrování (redirect) na transparentní proxy:

```
# iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 80 -j REDIRECT --
to-port 3128
```

9.4 Závěrem k NATu

NAT se snaží měnit spojení minimálním možným způsobem, pokud to jde, zachovává čísla portů. Porty jsou rozděleny do 3 tříd (do 512, 512 – 1023, 1024 a výše) [10]. Přemapovávání portů, pokud na něj dojde, se provádí v rámci jedné třídy. Některé protokoly jako např. *FTP*, které se nemají s NATem „rády“ a tudíž nemusí správně pracovat. NAT tabulka je použita pouze pro první paket spojení, proto nemá smysl v PREROUTING a POSTROUTING řetězcích používat filtrovací pravidla. Na to je určen především řetězec FORWARD.

Hlavní výhoda DNATu spočívá v tom, že můžeme schovat své servery a směrovat na ně pouze konkrétní povolená spojení. NAT jako takový se dnes velmi často využívá, především se stále rozšiřující potřebou připojovat do internetu více zařízení.

10 Zabezpečení serveru

Pro zabezpečení počítače používáme firewall. Jedná se o síťové, většinou softwarové, zařízení, které slouží k řízení a zabezpečování síťového provozu. Úkolem firewallu je ochránit počítač (případně lokální síť) před případnými útočníky.

Problematika firewallů je vždy poměrně komplexní a na první pohled neprůhledná záležitost. Jejich konstrukce vyžaduje určité, většinou nemalé množství času věnovaného pochopení nejen principu vytváření pravidel konkrétního firewallu, ale i principů síťování. Dá se prakticky říci, že žádný firewall nepracuje přesně tak, jak se očekává od první chvíle jeho nasazení, leda by byl opravdu velmi jednoduchý. Mít svůj firewall není dnes móda, ale nutnost.

V dnešní době, kdy rozličné viry každodenně útočí na naše počítače a servery, jsou kladeny velké nároky nejen na zabezpečení počítačů jako takových, ale i celých počítačových sítí. Každá společnost, která je dnes připojena na internet, bez ohledu na její velikost, vyžaduje určitou míru bezpečnostních opatření (Firewall) proti útočníkům. Všechny Firewally pracují na principu brány mezi jednotlivými síťovými rozhraními, kde se pomocí pravidel jasně definuje přístupová práva k dané službě.

Existují 4 základní typy firewallů:

- Paketové filtry

  Jedná se o nejstarší typ firewallu, který pracuje na principu přesného zadání adresy a portu příchozí komunikace. Hlavní výhodou je nenáročnost na zatížení procesoru.

- Aplikační brány

  Pracují na principu oddělení sítí, kdy veškerá komunikace probíhá přes tuto bránu. Výhodou tohoto řešení je relativně vysoké zabezpečení. Nevýhodou především vysoká náročnost na hardware.

- Stavové paketové filtry

  Provádějí kontrolu podobně jako paketové filtry, avšak ukládají také informaci o povolených spojeních. Tímto principem lze rozhodnout, zda příchozí pakety patří do povolených spojení či nikoli. U tohoto řešení je možné povolovat nejen porty, ale uvádět i směr navazovaných spojení. Mezi hlavní výhody patří vysoká rychlost a snazší konfigurace. Nevýhodou je nižší bezpečnost oproti aplikačním branám.

- Stavové paketové filtry s kontrolou protokolů a IDS

Tato ochranná metoda je nejvyspělejší a kromě informací o stavu spojení a dynamického otevírání portů je schopna pracovat formou *Deep inspection* a *Application Inteligence*. Tento systém progresivního scanování funguje v podstatě na bázi antivirové heuristické analýzy, kdy firewall podle známých vzorců kontroluje příchozí i odchozí komunikaci a v případě, že narazí na vzorec, který je zaevidovaný jako škodlivý, okamžitě jej zablokuje.

Mezi nejznámější firewally patří např. Kerio Personal Firewall, Norton Internet Security, iptables a další.

10.1 Řetězce aneb chainy

Jak pakety procházejí jednotlivými řetězci je patrné z minulého obrázku 2.

- INPUT řetězec nám filtruje pakety určené pro lokální stroj, OUTPUT pakety odchozí.

- FORWARD se aplikuje na pakety, které počítačem jen protékají (např. jde o router).

- MANGLE řetězec nám umožňuje pracovat s TTL a TOS hodnotami paketů a pakety označovat. NATovací řetězce provádějí překlad adres (Network Address Translation) a prochází jimi pouze první paket, na ostatní se pak aplikují stejná pravidla. Aktuálně nás zajímají především první dva řetězce - INPUT a OUTPUT.

10.2 Stavy spojení

Tak, jak probíhá připojování klienta k počítači, prochází spojení na obou stranách několika stavy, které budu nyní popisovat:

- NEW

Tento termín označuje nové spojení, které zatím v tabulce nebylo. Toto spojení by u TCP protokolu mělo být iniciováno paketem SYN (synchronize). Může však nastat i jiný případ a právě ten by náš firewall měl ošetřit. Když se navazuje spojení paketem FIN, je něco v nepořádku, většinou jde o sken portů.

- ESTABLISHED

Tento termín označuje navázané spojení. V případě TCP protokolu jde o spojení, u něhož proběhly 2/3 z procesu *3–way handshake*. Klient vyslal SYN, druhá strana odpověděla paketem SYN ACK (ACK = acknowledge), po jehož přijetí je spojení z hlediska klienta považováno za navázané, načež klient odeslal ACK, po jehož přijetí je spojení i na straně serveru prohlášeno za navázané. Tento mechanizmus se nevztahuje na protokoly, které nenavazují spojení (UDP, ICMP). U těchto protokolů je funkce trošku jiná, stačí přijatý a odeslaný paket.

• RELATED

Zvláštní typ spojení, které se vztahuje k jinému spojení, které je ESTABLISHED. Příkladem je např. protokol FTP (File transfer protokol), který nejčastěji na portu 21 přenáší řídící (control) data a na jiném portu potom vyžádaná data (soubory). Jiným příkladem je IRC. Tyto spojení s sebou nesou problém v podobě nutnosti odhalit iniciaci RELATED spojení a otevřít tak příslušný port. V praxi to dělají tzv. helper moduly např. *nf_conntrack_ftp*, jež sledují komunikaci na daném portu a pokud zaregistrují iniciaci spojení na jiném portu, otevřou jej. V případě FTP protokolu, když modul zaregistruje příkaz PASV nebo PORT, otevře zmíněný port a komunikace může probíhat.

• INVALID

Jedná se o stav, kdy paket nelze identifikovat nebo přiřadit k žádnému jinému stavu, např. ICMP error, který se ovšem nevztahuje k žádnému jinému spojení.

10.3 Co s paketem?

Paket patřící některému z pravidel je možno buď tiše zahodit (DROP), přijmout (ACCEPT), zalogovat (LOG) nebo zdvořile odmítnout (REJECT). Většina pravidel buď zahazuje, nebo přijímá. Často je výhodné paket odmítnout REJECTem, neboť pak nevchází v platnost čekání druhé strany na vypršení spojení např. při připojování na port služby auth (TCP 113), pokud na serveru auth server neběží. Je potřeba mít na paměti, že první platné pravidlo, které paket přijme, zahodí, nebo odmítne je zároveň pravidlem pro daný paket posledním.

10.4 Pravidla firewallu

Pravidla *iptables* se zapisují ve formě příkazů. Zde vedou dvě cesty, jak *iptables* nakonfigurovat. Pravidla se mohou zadávat v příkazovém řádku a to v okamžiku, kdy se zdá, že fungují, je uložit příkazem `iptables-save`. Takto se dají konstruovat jen jednoduché firewally a navíc je potřeba si v případě modulů zajistit jejich natažení jinou cestou. Druhá možnost je napsat vlastní skript, ve kterém se nadefinují nejen pravidla, ale i natáhnou moduly a upraví konfigurace v `/proc`. Dalším způsobem je napsat skript, kde si administrátor definuje pouze pravidla, která po spuštění skriptu uloží pomocí `iptables-save` a `/proc` a moduly si opět zajistí jinou cestou.

10.5 Základní příkazy iptables

Základní příkazy iptables:

`iptables` [`-t` *tabulka*] `-A` *chain pravidlo* – přidá na konec daného řetězce (chainu) pravidlo. Je–li uvedena tabulka, bude pravidlo přidáno do této konkrétní tabulky.

`iptables` [`-t` *tabulka*] `-I` *chain řádek pravidlo* – vloží do daného chainu na daný řádek pravidlo. Je–li uvedena tabulka, bude pravidlo přidáno do ní. Tento příkaz má využití při zadávání pravidel v konzoli, ve skriptu pro něj není uplatnění.

`iptables` [`-t` *tabulka*] `-D` chain *řádek pravidlo* – smaže pravidlo zadané číslem řádku nebo definicí.

`iptables` [`-t` *tabulka*] `-L` [*chain*] [*parametry*] – vypíše pravidla, tabulka a `chain` jsou nepovinné. Parametry jsou:

- – –v — verbose, „ukecaný“ výpis
- – –n — nepřekládá IP adresy a porty
- – – –line-numbers – vypíše i pořadí pravidla (hodí se pro mazání pravidel z konzole)

`iptables` [`-t` *tabulka*] `-F` *chain* – úplné vymazání (flush). Je–li zadán chain, jsou smazána pouze jeho pravidla.

`iptables` [`-t` *tabulka*] `-N` *chain* – založí nový chain.

`iptables` [`-t` *tabulka*] `-X` *chain* – smaže zadaný uživatelem vytvořený chain.

`iptables` [`-t` *tabulka*] `-P` *chain politika* – definuje danému chainu výchozí politiku (pravidlo) – `DROP`, `ACCEPT`, `REJECT`.

10.6 Nastavení našeho firewallu

Nastavení NATu:

```
-A POSTROUTING -s 192.168.2.0/255.255.255.0 -d !
192.168.2.0/255.255.255.0 -j SNAT --to-source [X.X.X.X]
```

Povolení loopbacku:

```
-A INPUT -i lo -j ACCEPT
```

Povolení příchozích spojní z vnitřní sítě:

```
-A INPUT -i eth1 -j ACCEPT
```

Povolení 10 ping odezev za sekundu z vnější sítě (orchrana před zahlcením flat pingem):

```
-A INPUT -i eth0 -p icmp -m icmp --icmp-type 8 -m limit --limit 1/sec --
limit-burst 10 -j ACCEPT
```

Veškeré příchozí spojení, které je navázáno z vnitřní sítě, povolovat:

```
-A INPUT -i eth0 -m state --state RELATED,ESTABLISHED -j ACCEPT
```

Povolení příchozí komunikace na porty TCP 25 (SSH), TCP a UDP 53 (DNS), TCP 80 (web):

```
-A INPUT -i eth0 -p tcp -m state --state NEW -m tcp --dport 25 -j ACCEPT
-A INPUT -i eth0 -p tcp -m state --state NEW -m tcp --dport 53 -j ACCEPT
-A INPUT -i eth0 -p udp -m state --state NEW -m udp --dport 53 -j ACCEPT
-A INPUT -i eth0 -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT
-A INPUT -s 195.113.20.119 -i eth0 -p tcp -m state --state NEW -m tcp --
dport 22 -j ACCEPT
```

Povolení směrování spojení, které je požadováno z vnitřní sítě:

```
-A FORWARD -i eth0 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -s 192.168.2.0/255.255.255.0 -d ! 192.168.2.0/255.255.255.0 -j
ACCEPT
```

Jakmile je firewall nastavený, nastane doba pro změnu základní politiky. Výchozí politiky jsou ACCEPT, kdy je povolen veškerý provoz. DROP znamená zakázáno vše, co není povoleno. To uděláme následujícími příkazy:

```
# iptables –P INPUT DROP
# iptables –P FORWARD DROP
```

11 Web server Apache

Http server je služba, která umožňuje publikovat internetové stránky na serveru. Různých typů web serverů je celá řada (Apache, Lighttpd, Thttpd), v naší práci se budeme zbývat především nejrozšířenějším web serverem., což je Apache. Tento program má dlouhou historii a je používán na většině web serverů. Tato kapitola bude věnována především instalaci a základnímu nastavení tohoto serveru. Pro instalaci použijeme příkaz:

```
# aptitude install apache2
```

Tímto se nainstaluje základní web server. Nyní nastává otázka, co od web serveru očekávat. Budou dostačovat stránky psané v html, nebo je požadováno používat také PHP a databáze (MySQL, Postgres)? V případě, že nebude dostačovat obyčejné html, musí se doinstalovat podporu PHP a např. MySQL:

```
# aptitude install mysql-server-5.0 php5
```

Jak PHP tak MySQL jsou dostupné pouze ve verzi 5. Starší verze již nejsou podporovány. Při instalaci systém požádá o nastavení hesla pro administrační (root) přístup do databáze a po zvolení a znovu potvrzení hesla se dokončí instalace. Poté je nutné vše pečlivě nastavit. Celý program je složen z modulů, které se dle potřeby a možností mohou rozšiřovat o další funkcionality. Jednou z těchto funkcionalit je např. podpora PHP, MySQL, nebo např. podpora SSL.

Jednotlivé moduly jsou umístěny v operačním systému v adresáři `/etc/apache2/mods-available/`. Tyto moduly se mohou používat vytvořením tzv. symbolických odkazů potřebných modulů do složky `/apache2/mods-enabled/`.

```
# ln –s /etc/apache2/sites-available/[zdroj] /etc/apache2/sites-
enabled/[cíl]
```

Je potřeba být u přidávání modulů obezřetný a používat pouze takové moduly, které jsou nutně potřeba. Některé moduly se mohou navzájem vylučovat či působit konflikty. Po úspěšné instalaci a případném nakopírování modulů je nutné Apache server restartovat, aby se projevily veškeré změny:

```
# /etc/init.d/apache2 restart
```

Nyní se načetla nová konfigurace a nyní je možné začít vytvářet vlastní webové stránky, které se v základní konfiguraci budou ukládat do adresáře `/var/www/`. V případě, že by tento adresář nevyhovoval, je možné tuto základní složku změnit na jinou v souboru *default* při změně položky *DocumentRoot*:

```
# vim /etc/apache2/sites-available/default
```

V případě, že bude Apache server dělat věci, které nejsou žádané, veškerá chybová hlášení se dají nalézt v adresáři `/var/log/apache2/`. Celá problematika webového serveru a jeho nastavení je velice obsáhlá a složitá. Pro tuto práci však bude tento základní funkční model naprosto dostačovat.

12 Instalace kamery

V kapitole 11 byl nainstalován a nastaven web server Apache. K dispozici je i kamerka, kterou se mohou publikovat obrázky z tohoto média na web server. Je nutné zjistit, zda je kamerka pod operačním systémem Linux podporována. Na to slouží příkaz,

```
# lsusb
```

který vypíše veškerá zařízení připojena k portu USB. V našem případě je výpis relevantní části následující:

```
Bus 002 Device 002: ID 05a9:a511 OmniVision Technologies, Inc. OV511+
WebCam
```

Za písmeny ID se nalézá identifikátor výrobce (`05a9`) a identifikátor produktu (`a511`) Zda je zařízení podporováno je možné zjistit pouze z internetových stránek, doporučené vyhledávání přes www.google.com. Z tohoto informačního zdroje bylo zjištěno, že zařízení je podporováno, ale je potřeba mít v jádře modul ov511[9] . Primárně je nutné zjistit, zda tento modul máme v jádře zaveden. Většinou jsou potřebné moduly v jádře již připraveny a při připojení zařízení je automaticky `udev`, mechanismus pro zavádění modulů do jádra, který v sobě taktéž obsahuje tzv. hot plug, nadetekuje a moduly do jádra automaticky zavede.

```
# lsmod | grep ov511
```

Tento příkaz vypíše veškeré moduly, které jsou v jádře (`lsmod`) a zároveň vypíše pouze ty řádky výpisu, kde se nachází ov511, což je náš hledaný modul (`grep ov511`). Pokud se na obrazovce ukáže následující výpis, znamená to, že v jádře se daný modul nachází.

```
ov511                  83360  0
videodev               41600  1 ov511
```

V opačném případě je možné pokusit se modul do jádra zavést ručně, což učiníme příkazem:

```
# modprobe ov511
```

Je však pravděpodobné, že takový příkaz selže, neboť v jádře není patřičný ovladač. V takovém případě je třeba ovladač doinstalovat, což může být složitý proces, který již není v požadovaném rámci této práce.

Nyní máme podporu v jádře a je nutné nainstalovat program, který bude ze zařízení číst záznam a dále tento záznam ukládat. Pro tyto účely je vytvořen např. program webcam, který si nainstalujeme.

```
# aptitude install webcam
```

Po instalaci je třeba vytvořit konfigurační soubor dle manuálových stránek.

```
# man webcam
```

Aktuálně nastávají tři možnosti, jak obrázky ukládat. První možností je na stanici, ke které je připojena kamera, nainstalovat dle již zmíněného postupu web server a výstup z kamery ukládat lokálně. Druhá, koncepčnější a také složitější metoda, je soubory z kamery posílat lokální sítí na server, kde je poté zobrazovat. Oba tyto způsoby vyžadují mírně rozdílnou konfiguraci dalšího nastavení. Třetí způsob je přímé napojení kamerky k serveru.

• 1. metoda

Jak bylo již naznačeno, princip prvního způsobu spočívá v připojení kamerky ke klientovi, za předpokladu, že klient má operační systém Linux. Obrázky jsou ukládány na klientský počítač lokálně a jsou na tomto klientském počítači zobrazeny pomocí místního web serveru. Aby bylo možné obrázek zobrazovat i z veřejné sítě, je nutné přesměrovat na serveru jiný port, který je aktuálně volný. Zvolíme si např. port 8080.

Na serveru přes příkaz:

```
# iptables -t nat -A PREROUTING -d [veřejná IP adresa] -p tcp -m tcp --
dport 8080 -j DNAT --to-destination [IP adresa klienta].
```

Je nutné také povolit tuto komunikaci ve firewallu:

```
# iptables -A FORWARD -d [IP adresa klienta] -p tcp -m tcp --dport 8080 -
j ACCEPT
```

Je nutné vytvořit konfigurační soubor pro kameru:

```
# vim .webcamrc

[grab]
 device = /dev/video0              /cesta k zařízení kamerky
 text = "webcam %Y-%m-%d %H:%M:%S"/text, který se zobrazí
 fg_red = 255                           /nastavení červené barvy
 fg_green = 255                    /nastavení zelené barvy
 fg_blue = 255                          /nastavení modré barvy
 width = 320                            /výška obrázkového výstupu
 height = 240                           /šířka obrázkového výstupu
 delay = 3                  /čas mezi jednotlivým snímáním snímků
 wait = 0                               /mezičas čekání
 input = Camera                    /typ vstupu
 norm = pal                        /typ výstupu
 rotate = 0                        /rotace snímku
 top = 0                                /umístění textu (nahoře)
 left = 0                               /umístění textu (vlevo)
```

```
 bottom = -1                              /umístění textu (dole)
 right = -1                             /umístění textu (vpravo)
 quality = 75                             /kvalita snímku v DPI
 trigger = 0
 once = 0




[archive]
 host =                                   /adresa počítače, kam snímek poslat
 user =                                   /uživatelské jméno
 pass =                                   /heslo
 dir = /var/www                     /adresář pro uložení
 file = webcam.jpeg                       /název souboru
 tmp = uploading.jpeg               /název dočasného souboru
 passive = 0                              /mód přenášení
 debug = 0                          /ladící mód (1=ano,0=ne)
 auto = 0
 local = 1                          /lokální ukládání (1=ano,0=ne)
 ssh = 0                                  /SSH ukládání (1=ano,0=ne)
```

Po vytvoření konfiguračního souboru pouze program spustíme:

```
# nohup webcam .webcamrc
```

Je možné si všimnout příkazu `nohup`. Tento příkaz udává, že proces, který je spuštěn bude běžet na pozadí tzn. že i po ukončení příkazové řádky proces stále běží. Nyní se již může do internetového prohlížeče napsat internetovou adresu:

```
http://[veřejná IP adresa]/webcam.jpeg
```

a prohlížet si grafický výstup z kamerky.

• 2. metoda

Kamerka jako taková je připojena ke klientskému počítači a výstup z kamery je přímo přenášen pomocí SSH na server, kde je dále zobrazován přes web server.

Konfigurace je podobná jako u první metody s mírně modifikovaným konfiguračním souborem:

```
# vim .webcamrc

[grab]
 device = /dev/video0
 text = "webcam %Y-%m-%d %H:%M:%S"
 fg_red = 255
 fg_green = 255
 fg_blue = 255
 width = 320
 height = 240
 delay = 3
```

```
 wait = 0
 input = Camera
 norm = pal
 rotate = 0
 top = 0
 left = 0
 bottom = -1
 right = -1
 quality = 75
 trigger = 0
 once = 0
[archive]
 host = [IP adresa serveru]
 user = root
 pass = [heslo]
 dir = /var/www
 file = webcam.jpeg
 tmp = uploading.jpeg
 passive = 0
 debug = 0
 auto = 0
 local = 0
 ssh = 1
```

Je důležité, aby heslo pro přístup k serveru nebylo vyplněno. Dále je nutné použít vytvořený certifikát, kterým se budeme k serveru hlásit (viz kapitola 6) a tento certifikát zadat bez hesla. Z toho důvodu, že program vždy navazuje nová spojení a při každém vkládání obrázku (v našem případě každé 3 sekundy) by požadoval vložení hesla. Doporučuje se nejprve vyzkoušet SSH přístup na server, zda je v naprosto funkčním stavu. Předejde se tak případným nedostatkům.

Nyní je možné spustit streamovaní kamerky stejným způsobem, jako u první metody:

```
# nohup webcam .webcamrc
```

a stejně tak se podívat na internetové stránky:

```
http://[veřejná IP adresa]/webcam.jpeg
```

• 3. metoda

Poslední metodou je možnost připojení kamerky přímo na server, kde daný výstup z kamerky budeme lokálně ukládat. Na tomto serveru je již nakonfigurovaný web server, přes který se může zobrazovat výstup kamerky. Tento způsob je téměř shodný s 1. metodou, pouze s rozdílem, že snímací zařízení připojujeme k serveru.

Nejvýhodnější se zdá být 3. metoda. A to především z hlediska jednoduchosti a dlouhodobého streamování kamerky, za předpokladu, že se server většinou nevypíná z hlediska stálé dostupnosti. Metoda 1 a 2 se může využít i na značné fyzické vzdálenosti server X client, což u třetí metody není z technického hlediska realizovatelné [11].

13 Závěr

Celá tato práce je založena na vytvoření materiálu pro výuku. Podařilo se tak vytvořit lehce srozumitelný výukový materiál a osvětlit v něm základní principy instalace operačního systému, služeb i jejich funkcí. Nepřímo se navazuje předmět Inženýrská informatika, a tudíž se předpokládají určité základní znalosti z oblasti síťových technologií. Použil jsem především podporované, aktuálně dostupné a v běžné praxi nejvíce využívané a známé balíky z operačního systému GNU/Linux, distribuce Debian 5 Stable. Provedl jsem praktickou instalaci, nastavení i samotné vyzkoušení funkčnosti celého systému a jednotlivých služeb.

V této práci se podařilo čtenáře seznámit se základními principy práce s operačním systémem Linux i nejčastěji využívanými příkazy pro práci v příkazovém řádku. Veškeré aplikace a služby jsou v dnešním světě směřovány na internetové stránky a služby. Je proto zařazena a vysvětlena instalace webového serveru, na kterou následně navazuje praktické využití technologie přenosu vizuálního snímaného obrázku z kamerky připojené z různých míst lokální počítačové sítě.

Předpokládá se, že do zadaného návrhu síťové infrastruktury budou mít přístup i další externí počítače. Z tohoto důvodu bylo myšleno i na návrh překladu adres spolu s automatickým přidělováním adres novým počítačům pomocí DHCP serveru. Nedílnou součástí je také zabezpečení serveru a tím i celé lokální sítě pomocí firewallu před případnými útoky. Podrobně je vysvětlena nejen funkce a typy jednotlivých firewallů, ale i samotná konfigurace přes firewall založený na iptables.

Je nutné podotknout, že přístup a konfigurace se mohou částečně lišit na typu zvolené distribuce, ale teoretické principy a postupy zůstávají stejné.

14 Seznam použitých zkratek a termínů

- označení shellu uživatele root

$ - označení shellu ostatních uživatelů

bash – volně šiřitelný Unixový shell z projektu GNU

DoS útok – útok na počítač nebo síť, který má způsobit nedostupnost dané služby

GNU/GPL – všeobecná veřejná licence pro svobodný software

GNU/Linux – volně šiřitelný operační systém

grub – zavaděč operačního systému

hot plug – schopnost připojovat a odpojovat komponenty počítače za běhu

HTML – jazyk pro vytváření jednoduchých www stránek

IP adresa – číslo, které jednoznačně identifikuje síťové rozhraní v počítačové síti

kernel – jádro operačního systému

logování – softwarové zaznamenávání důležitých hlášení systému a aplikací

paket – blok přenášených informací počítačovou sítí

PHP – skriptovací programovací jazyk určený především pro programování dynamických internetových stránek

provider – poskytovatel internetového připojení

RFC – standardy a další dokumenty popisující internetové protokoly, systémy apod.

root – uživatel s maximálními možnými právy neboli superuživatel

shell – program, který interpretuje příkazy uživatele (též příkazový řádek)

streaming - technologie přenosu audiovizuálního materiálu mezi zdrojem a koncovým uživatelem

switch – aktivní síťový prvek, který propojuje jednotlivé segmenty počítačové sítě

USB – univerzální sériová sběrnice

15 Literatura

[1] http://tools.ietf.org/html/rfc1918

[2] http://www.debian.org

[3] http://distrowatch.com

[4] http://www.abclinuxu.cz/clanky/ruzne/vysledky-ankety-o-nejoblibenejsi-distribuci-2008

[5] http://cs.wikipedia.org/wiki/Seznam.cz

[6] http://www.root.cz/clanky/seznam-cz-na-vetsine-serveru-mame-debian/

[7] http://debian.org/intro/free

[8] http://www.chiark.greenend.org.uk/~sgtatham/putty/

[9] http://www.abclinuxu.cz/hardware/vstupni-zarizeni/digitalni-kamery/safeway-usb-320k

[10] http://www.netfilter.org/documentation/HOWTO/NAT-HOWTO-6.html#ss6.3

[11] http://www.techdot.eu/index.php/2008/01/06/teorie-usb-na-30m/

[12] http://global.bsa.org/idcglobalstudy2007/

[13] http://tools.ietf.org/html/rfc1531

[14] http://tools.ietf.org/html/rfc1305

[15] http://tools.ietf.org/html/rfc3022

[16] http://tools.ietf.org/html/rfc4252