ČESKÁ ZEMĚDĚLSKÁ UNIVERZUTA V PRAZE
Provozně ekonomická fakulta
Katedra informačního inženýrství

Bakalářská práce

Router a firewall na Linuxu

Kateřina Bubeníčková

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci *Router a firewall na Linuxu* jsem vypracovala samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autorka uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušila autorská práva třetích osob.

V Praze dne 14. března 2010

Poděkování

Ráda bych touto cestou poděkovala doc. Ing. Arnoštu Veselému, CSc. za cenné připomínky k práci. Chtěla bych také vyjádřit díky svým dětem a manželovi Jirkovi za trpělivost a neustálou podporu nejen při psaní této práce, ale i při celém mém studiu.

Router a firewall na Linuxu
Router and firewall on Linux

Souhrn:
Práce se skládá z teoretické a praktické části. V teoretické části jsou vymezeny základní pojmy jako router a firewall, popsány protokoly rodiny TCP/IP, nastíněny základní principy fungování firewallu a zmíníme se také o bezpečnosti sítě. Dále práce popisuje specifika implementace routeru a firewallu na operačním systému Linux.

V praktické části jsme prozkoumali konfigurační soubory linuxového firewallu, který byl v provozu v konkrétní organizaci v letech 2005 až 2008 s ohledem na teoretické poznatky uvedené v předchozí části. Cílem práce bylo najít slabá místa a chyby konfigurace a navrhnout úpravy konfigurace, které nalezené problémy odstraňují.

Při zkoumání konfiguračních souborů se ukázalo, že v nastavení bylo slabé místo, způsobené méně bezpečným nastavením politiky přístupu do DMZ. Také byly v konfiguračním souboru neúplné ingresní a egresní kontroly. Navrhli jsme příslušné úpravy, upravený konfigurační soubor je uveden v příloze práce.

Tento upravený konfigurační soubor by bylo však ještě třeba otestovat v produkčním prostředí.

Summary:
The work consists of theoretical and practical part. In the theoretical part main terms as router and firewall are defined and protocols of TCP/IP stack are described. Esentials of firewall functionality and security are mentioned and the specifics of implementation of router and firewall on Linux are discussed.

In the practical part configuration files of a Linux firewall are examined. This firewall was used in production during 2005 – 2008. The goal of the work was to find weak points and errors in the configuration and to propose some changes for elimination of these problems.

The most critical of the discovered issues was less secure setting of access policy to DMZ. Also Egress and Ingress controls were partialy missing. The modified and improved configuration file is in the appendix of the work.

This new configuration file should be tested in a production enviroment.

Klíčová slova: router, firewall, linux, konfigurační soubor, iptables, netfilter, bezpečnost, TCP/IP, síť

Keywords: router, firewall, linux, configuration file, iptables, netfilter, security, TCP/IP, network

Obsah

Jeruzalém bude městem nehrazeným,
tolik v něm bude lidí i dobytka.
Já sám, je výrok Hospodinův,
budu ohnivou hradbou kolem něho
a slávou uprostřed něho.

Zacharjáš 2:8b - 9

1 Úvod

Téma práce *Router a firewall na Linuxu* jsem si vybrala proto, že již téměř deset let pracuji s operačním systémem Linux a jeho filozofie je mi blízká. V rámci svých pracovních povinností jsem také nainstalovala linuxový firewall a asi tři roky jej spravovala. Tento server již není v provozu, místo něj byl na podzim r. 2008 zakoupen komerční firewall od firmy Cisco. Ukázalo se ale, že nový Cisco firewall má kromě výhod i nevýhody a jeho funkcionalita je v některých ohledech ne zcela dostačující. Proto je možné, že v budoucnu bude třeba jej nahradit nějakým jiným, což nevylučuje možnost návratu k linuxovému firewallu.

Doufala jsem, že při zpracovávání tématu se seznámím do hloubky s teoretickými principy fungování firewallu, na jejichž podrobné zkoumání za provozu, často ve spěchu a metodou pokus-omyl, nebylo dost času.

2 Cíl práce a metodika

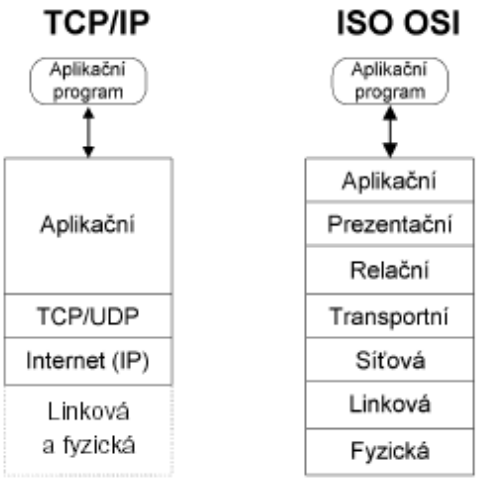
Práce má dvě části, teoretickou a praktickou.

V teoretické části se budeme zabývat nejprve vymezením důležitých pojmů router, firewall, popisem protokolů, a vysvětlením obecných principů fungování a zmíníme se také o bezpečnosti sítě. V oddíle nazvaném *Router a firewall na Linuxu* se zaměříme na specifika implementace na operačním systému Linux.

Praktická část bude spočívat v prozkoumání konfiguračních souborů z firewallu, který byl v provozu v konkrétní organizaci v letech 2005 až 2008, s ohledem na teoretické poznatky uvedené v předchozí části. Cílem je najít slabá místa a chyby konfigurace a navrhnout úpravu této konfigurace, která nalezené problémy odstraňuje.

3 TCP/IP, router a firewall

V této části si chceme ujasnit základní terminologii, vysvětlit, co to je router a firewall. Ale abychom mohli pochopit jejich funkci, je třeba se zabývat komunikačními protokoly, se kterými router a firewall pracují. V následujícím textu objasníme, co to vlastně komunikační protokoly jsou, jaká je jejich struktura a k čemu slouží.



Obrázek 1: Porovnání vrstev modelu OSI a TCP/IP

3.1 TCP/IP: seznámení s protokoly

Mezinárodní normalizační organizace (ISO – International Standards Organization) vyvinula model OSI (Open Systems Interconnect) pro srovnávání komunikačních protokolů.[1] Tento model má sedm vrstev (viz obr. 1), každá vrstva řeší vlastní úkol a komunikuje pouze s vrstvou nad sebou a pod sebou.[2]

V praxi se však prosadila architektura TCP/IP, která obsahuje pouze čtyři vrstvy.[3] Přestože mnoho autorů v literatuře uvádí srovnání modelu OSI a vrstev architektury TCP/IP, je možno říci, že model OSI vznikl na zelené louce a realita se ukázala jiná. Porovnání modelu OSI a modelu TCP/IP ukazuje obrázek 1.[4] Je však třeba si uvědomit, že jednotlivé vrstvy v modelech OSI a TCP/IP si neodpovídají zcela přesně tak, jak to znázorňuje obrázek.

Zkratka TCP/IP znamená *Transmition Comunication Protocol / Internet Protocol* a označuje skupinu protokolů používaných ke komunikaci na internetu. Všem těmto protokolům dohromady se také říká zásobník TCP/IP nebo rodina protokolů TCP/IP (stack TCP/IP). Tyto protokoly se prosadily, protože disponují celou řadou výhod, které jsou stručně shrnuty v tabulce 1.[5]

Hlavní nevýhodou zásobníku TCP/IP je v tom, že ačkoli byl vyvíjen pro vojenské účely, nijak se při jeho vzniku nebral ohled na bezpečnost. Uživatelé byli hlídáni jinak, spíše se aplikovala fyzická bezpečnost. Data nejsou žádným způsobem chráněna proti odposlechu, nejsou šifrována ani kódována, chybí tzv. důvěrnost. Nejsou chráněna proti ztrátě či změně při nespolehlivém

[1] Viz Perkins. *Firewally*, s. 56–57.

[2] Viz Habrman. *Síťové protokoly (I. část) – Model OSI*

[3] Danou skutečnost hezky vyjadřuje motto v prezentaci *Peterka: Rodina protokolů TCP/IP*
Víš-li, jak na to, čtyři vrstvy ti plně postačí ...
... nevíš-li, ani sedm ti jich nepomůže

[4] Obrázek převzat Krasek. *Peer to peer sítě od A do Z*.

[5] Viz Perkins. *Firewally*, s. 54–56.

[6] Přenosem paketů poštovními holuby, letadly apod. se zabývá RFC 1149, aktualizováno v RFC 2549, IP Over Avian Carrier.

TCP/IP	
Založený na paketech	Přes jedno síťové spojení může komunikovat více počítačů
Umožňuje decentralizované řízení	Každý, kdo dostane přidělen určitý rozsah IP adres, může tyto adresy dále přidělovat
Peer to peer komunikace	Každý počítač může iniciovat nebo přijímat spojení
Směrovatelný	Směrování usnadňuje předávání dat přes více LAN síťí pomocí směrovačů, není třeba se spoléhat na protokolové brány
Nezávislý na konkrétním přenosovém médiu	Funguje na Ethernetu, Token Ring, ArcNet, FDDI, dokonce by službu mohli provádět i poštovní holubi [6]
Otevřený standard	Můžete si všechnu dokumentaci stáhnout zdarma na internetu
Nic nestojí	Vyvinuly jej univerzity v USA za peníze ministerstva obrany, nemusí se platit žádné poplatky za používání
Robustní	Umí se elegantně vzpamatovat při přerušení nebo výpadku části sítě
Pružný	Pro jednoduchá zařízení stačí jednoduché protokoly, pro složité úkoly je možno implementovat dostatečně komplexní potokoly
Pragmatický	Protokoly a jejich vlastnosti se na rozdíl např. od zásobníku OSI zaváděly na základě potřeby.
Není dokonalý	Hlavními problémy je adresace a zabezpečení

Tabulka 1: Vlastnosti zásobníku TCP/IP

přenosu, chybí integrita.[7] Zabudované kontroly se dokáží vypořádat spíše se selháním technické stránky než s cílenou snahou útočníků poškodit, změnit nebo číst data.[8]

Mechanismus zapouzdření dat v síti TCP/IP ukazuje obrázek 2. Protokoly TCP/IP jsou specifikovány v RFC dokumentech. Seznam nejdůležitějších RFC dokumentů týkajících se TCP/IP je v tabulce 2.[9]

Problematika TCP/IP je velmi zajímavá a značně komplexní, pro potřeby této práce se omezíme pouze na základní charakteristiky protokolů, které je nutno znát v souvislosti s konfigurací routeru a firewallu.[10]

3.1.1 Protokoly síťové vrstvy: IP, ICMP

IP protokol je jediný protokol z rodiny TCP/IP na síťové vrstvě. Jeho charakteristikou je, že je nespolehlivý a nespojovaný. Hlavička protokolu IP je na obrázku 3.

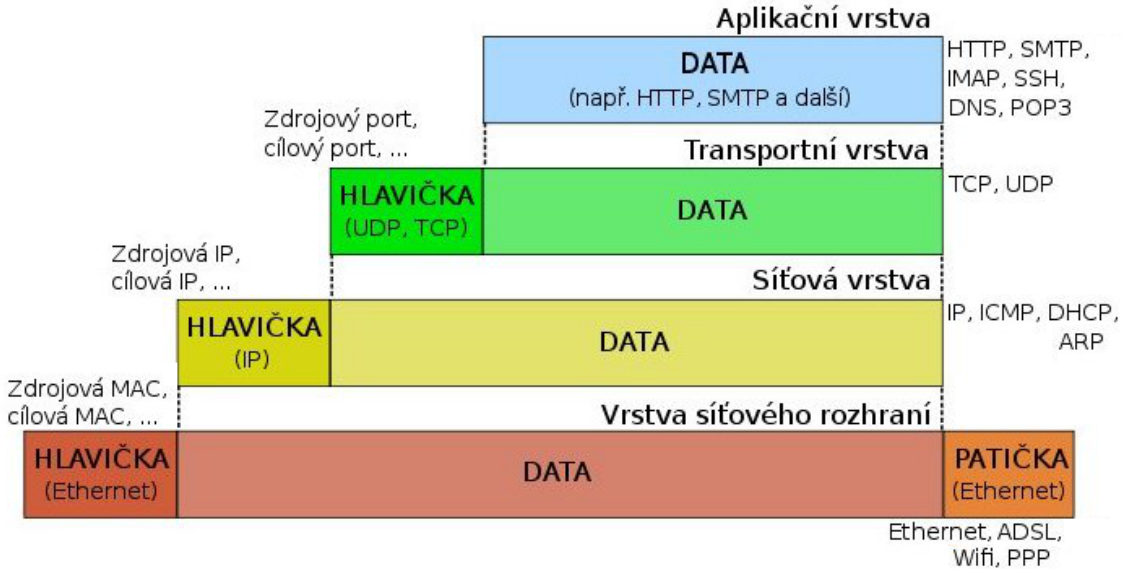
K oznamování problémů IP protokolu se používá protokol ICMP. ICMP je

[7] Viz Peterka. *Rodina protokolů TCP/IP*, část 2.

[8] V současné době se vyvíjejí nové protokoly, které by měly některé nedostatky TCP/IP řešit, např. SCTP (Stream Control Transmission Protocol), ale nejsou ještě příliš rozšířené a jejich popis přesahuje rámec této práce.

[9] Viz Pumanová. *Jak se orientovat v RFC*.

[10] Podrobný popis problematiky je např. v přednáškách Peterka. *Rodina protokolů TCP/IP*.



Obrázek 2: Zapouzdření dat v síti TCP/IP

RFC 760	DoD Standard Internet Protocol (IP) – první návrh IP (nahrazený RFC 791, ale stále zajímavý);
RFC 768	User Datagram Protocol (UDP) [STD6] – transportní protokol
RFC 791	Internet Protocol (IP) [STD5] – síťový protokol IP
RFC 792	Internet Control Message Protocol (ICMP) [STD5] – protokol síťové vrstvy (chybová a informační hlášení)
RFC 793	Transmission Control Protocol (TCP) [STD7] – transportní protokol
RFC 826	Ethernet Address Resolution Protocol (ARP) [STD37] – mapování hardwarových a IP adres
RFC 854	Telnet Protocol Specification [STD8] – aplikační protokol
RFC 959	File Transfer Protocol (FTP) [STD9] – aplikační protokol
RFC 1631	NAT - Network Address Translation

Tabulka 2: Seznam nejdůležitějších RFC dokumentů týkajících se TCP/IP

zvláštní tím, že se vkládá do IP paketů, aby prošly všude tam, kde projde IP a nebyly vyžadovány multiprotokolové směrovače, ale současně se ICMP považuje za součást síťové (a nikoli transportní) vrstvy. Toto jediné porušení vrstevnatých modelů je motivováno praktickými důvody.[11]

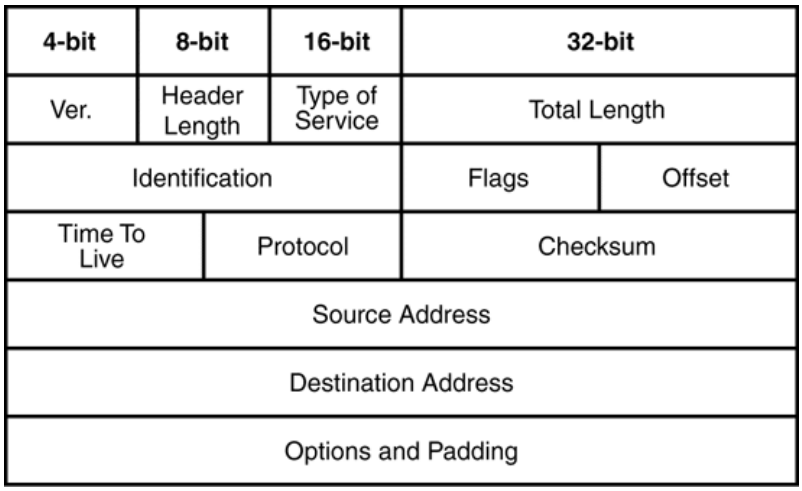
Na této vrstvě je uzel identifikován jako celek svou IP adresou.

3.1.2 Protokoly transportní vrstvy: TCP, UDP

Na transportní vrstvě se řeší požadavky na spojovanou komunikaci a spolehlivost. Na této úrovni dosud není možné řešit kvalitu služeb (QoS). Existují dva hlavní transportní protokoly – TCP a UDP.

Transportní vrstva má za úkol rozlišovat na uzlu různé služby, úlohy, démony. To se děje pomocí tzv. multiplexu (sběru dat) a demultiplexu (rozdělo-

[11] Viz Peterka. *Rodina protokolů TCP/IP*, část 2.



Obrázek 3: Hlavička protokolu IP

vání dat) mezi různé entity vyšších vrstev. Entity aplikační vrstvy jsou identifikovány nepřímo, pomocí tzv. portů. Jedna entita může být asociována s více porty, ale s jedním portem nesmí být asociováno více entit. Význam některých portů je pevně dán.[12] Přehled některých důležitých čísel portů je uveden v tabulce 3. Spojení mezi dvěma uzly je jednoznačně určeno pětic*í transportní protokol, IP adr1, port1, IP adr2, port2*.

Transportní služby mohou být tří typů: *stream*, *raw* nebo *datagram*. *Stream* vytváří iluzi roury, k členění na bloky dochází interně a transparentně, přenos je spolehlivý. Takto funguje protokol TCP.

Raw je speciální režim pro přímý přístup k nižším vrstvám, využívá se na testování, pro ping, OSPF.

Typ transportní služby *datagram* vytváří iluzi blokového přenosu. Přenos je nespolehlivý, nespojovaný, ztráty a duplicity nejsou ošetřeny. Takto funguje UDP.[13]

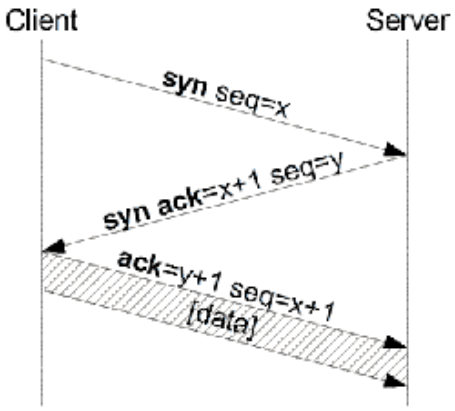
TCP je služba spojovaná a spolehlivá, jedná se o typ přenosu stream. Pro-

[12]Dříve se význam portů zveřejňoval v RFC, naposledy RFC 1700, nyní pouze on-line na adrese http://www.iana.org/assignments/port-numbers.

[13]Peterka. *Rodina protokolů TCP/IP*, část 2.

20/TCP	FTP data
21/TCP	FTP —- control (příkazy)
22/TCP,UDP	Secure Shell (SSH) -— používá se pro bezpečné přihlášení, bezpečný přenos souborů (scp, sftp) a forwardování portů
25/TCP	Simple Mail Transfer Protocol (SMTP) -— používá se pro routování e-mailů mezi mailovými servery
53/TCP,UDP	Domain Name System (DNS)
80/TCP,UDP	Hypertext Transfer Protocol (HTTP)
113/TCP,UDP	Authentication Service (auth)
443/TCP,UDP	Hypertext Transfer Protocol over TLS/SSL (HTTPS)

Tabulka 3: Seznam čísel portů důležitých protokolů TCP a UDP



Obrázek 4: Třícestný handshake

tokol TCP používá k ustanovení spojení tzv. třícestný handshake, který je znázorněn na obrázku 4[14]. Tento handshake začítá tím, že počítač, který žádá o spojení, zašle paket s nastaveným příznakem SYN. Počítač na druhé straně odpoví buď SYN/ACK, nebo SYN/RST, aby klientovi oznámil, zda spojení bylo přijato, nebo odmítnuto. Když klient přijme SYN/ACK, odpoví paketem s nastaveným příznakem ACK a spojení je navázáno. Během tohoto inicializačního handshake jsou také dohodnuty další specifické podmínky TCP spojení. Hlavička TCP paketu je na obrázku 5.

16-bit							32-bit
Source Port							Destination Port
Sequence Number							
Acknowledgement Number (ACK)							
Offset Reserved	U	A	P	R	S	F	Window
Checksum							Urgent Pointer
Options and Padding							

Obrázek 5: Hlavička protokolu TCP

Význam polí v TCP hlavičce:
Source port – port procesu generujícího datagram
Destination port – označuje, kterému procesu na cílovém uzlu jsou data určena
Sequence Number – sekvenční číslo prvního datového oktetu v segmentu (pokud není nastaven příznak SYN). Pokud je nastaven příznak SYN, jedná

[14]Viz Odvárka. *TCP handshake krok za krokem.*

se o tzv. initial sequence number (ISN) a první datový oktet má číslo ISN + 1
Acknowledgement Number – má význam, pouze když je nastaven kontrolní bit ACK. Toto číslo je nastaveno na hodnotu, kterou odesílatel očekává v poli Sequence Number v následujícím paketu. Je-li ustaveno spojení, je toto číslo vždy posíláno.
Data Offset – specifikuje číslo vyjádřené 32bitovým slovem. Indikuje, kde v segmentu začínají data přenášená tímto datagramem.
Reserved – toto 6ti bitové pole je rezervované a mělo by vždy být nulové
Control Bits – kontrolní bity (příznaky) zajišťující „handshaking“ a ostatní specifické procesy

- URG – Urgent Pointer
- ACK – Acknowledgement
- PSH – Push funkce
- RST – Reset spojení
- SYN – synchronizace sekvenčních čísel
- FIN – oznámení, že odesílající nemá žádná další data

Window – množství dat oktetech, které je potvrzováno najednou
Checksum – kontrolní součet, není povinný a v tom případě je 0
Urgent Pointer – údaj je platný, pouze pokud je nastaven příznak URG
Options – pole proměnné délky určené pro volitelné parametry TCP, parametr je používán např. pro indikaci maximální velikosti segmentu, kterou je přijímají strana schopna zpracovat
Padding – specifické množství nulových bitů doplňujících hlavičku tak, aby měla 32 bitovou hranici (tj. aby byla beze zbytku dělitelná 32)

Informace tvořená dvojicí *IP adresa, číslo portu* je nazývána socket. Socket představuje definitivní informaci o cíli TCP komunikace.

3.2 Co je to router

Router, česky směrovač, je zařízení, které se nachází na rozhraní dvou nebo více sítí a zajišťuje předávání paketů mezi těmito sítěmi. Routery používají hlavičky paketů a směrovací tabulky k forwardování paketů. Ke vzájemné komunikaci a ke konfigurování nejlepší cesty mezi dvěma hostiteli využívají také protokoly (např. ICMP). Routery téměř neprovádějí filtrování dat.

Jako router může fungovat také operační systém, např. Linux nebo MS Windows NT, takže slovo router má vlastně dva významy. Buď označuje celé zařízení, nebo pouze program s odpovídající funkcionalitou.[15]

Router je schopen předávat pakety z jedné sítě do druhé i v případě, že každá síť používá jiný linkový protokol, např. *Ethernet* a *Token Ring*. Je to

[15]Viz Dostálek. Kabelová. *Velký průvodce protokoly TCP/IP a systémem DNS*, s. 128.

proto, že u každého paketu odstraní hlavičku linkového protokolu a znovu ho zabalí do nového linkového rámce podle potřeb sítě, do které ho předává. Při přebalování paketů ale obsah IP datagramu router nesmí měnit.[16]

Někdy se ve stejném významu jako router používá i gateway (brána). Rozdíl mezi bránou a směrovačem je v tom, že směrovač leží na rozhraní jakýchkoli sítí, kdežto brána je na rozhraní vnitřní sítě a Internetu.

V souvislosti s pojmem router je vhodné zmínit i další zařízení, která se od routerů odlišují, ale mají některé podobné funkce: *bridge* a *switch*. Bridge (most) také zpracovává tok dat, ale pracuje pomocí jiných metod.[17] Switch (přepínač) je aktivní síťový prvek, který propojuje jednotlivé segmenty sítě.[18] Obecně můžeme říci, že routery leží na hranicích sítí, kdežto switche a bridge se umísťují uvnitř jedné sítě.

3.2.1 Routování

Router rozhoduje o další cestě paketu podle routovacích tabulek. Podle způsobu vzniku a úprav směrovacích tabulek rozlišujeme routování na *statické* a *dynamické*. Při statickém routování je obsah routovacích tabulek neměnný, pevně zadaný. Při dynamickém routování se často základ routovacích tabulek zadá ručně a pak se tyto tabulky dynamicky upravují podle aktuálního stavu sítě. Pro tyto úpravy se používají dvě základní metody routování: *vector-distance routing* a *vector-state routing*.

Směrování mají na starosti specializované směrovací protokoly, např. RIP a OSPF.

Při doručování paketů rozlišujeme přímé a nepřímé doručování. Při přímém doručování je příjemce i odesilatel na stejné síti, k nepřímému doručování dochází, pokud se síť příjemce liší od sítě odesilatele.

K routování dochází podle následujícího algoritmu:[19]

- Vezmi plnou IP adresu příjemce I_d a zjisti, zda se příjemce nachází ve stejné síti jako ty. Pokud ano, použij přímé směrování, jinak
- Začni prohledávat směrovací tabulku postupně podle velikosti prefixu.

[16] S výjimkou položky TTL, kterou zmenší aspoň o jednotku. Dosáhne-li TTL nuly, datagram je zahozen. Kvůli změně TTL musí také router pokaždé přepočítávat kontrolní součet, viz Dostálek. Kabelová. *Velký průvodce protokoly TCP/IP a systémem DNS*, s. 129.

[17] Bridging se provádí na 2. (spojové) vrstvě (L2), routing pak na 3. (síťové) vrstvě (L3) referenčního modelu ISO/OSI. Most tedy směruje rámce podle jejich hardwarové MAC adresy, zatímco router se rozhoduje podle IP adresy unitř přenášeného datagramu. Klasický most proto není schopen rozlišovat sítě.

Z různých důvodů se však do mostů tato schopnost implementuje, takže most může ležet i na pomezí sítí. Při projektování větší sítě si musíme vybrat mezi přemostěním nebo rozdělením na různé podsítě propojené routery. Pokud je v routované síti počítač fyzicky přesunut z jedné síťové oblasti do jiné, musí mu být přidělena jiná IP adresa. Pokud je počítač přesunut uvnitř přemostěné (bridged) sítě, není potřeba nic překonfigurovávat. (Viz též Peterka. *Bridge.*)

[18] Nejjednodušší switche pracují na 2. vrstvě modelu OSI, ale existují tzv. L3 switche, které pracují ve 3. vrstvě modelu OSI. V současné době se pojem L3 switch používá jako synonymum pro router. Můžeme se dokonce setkat s pojmem L4 switch (viz NetworkWorld research center: LANs/Routers), což je zařízení, které umí analyzovat protokoly 4. vrstvy OSI modelu. (Viz také Peterka. *Kdy použít switch.*)

[19] Uvedený algoritmus se nazývá TCP/IP destination driven routing (routování řízené cílovou adresou), je popsáno např. v Peterka. *Rodina protokolů TCP/IP*, novější způsob routování se nazývá policy based routing (routování založené na politikách), které umožňuje routovat téměř podle každého bitu v hlavičce paketu, např. podle zdrojové adresy, hodnoty TOS atd. (Viz Andreasson, *Iptables Tutorial 1.2.2*). Ještě podrobnější popis routovacího algoritmu je možno nalézt v Endorf, *Detekce a prevence počítačového útoku*, s. 71–72.

Pokud se hodnota prefixu právě prohledávané položky shoduje se stejnolehlou částí I_d (příslušným počtem vyšších bitů), doručuj nepřímo dle této položky. Jinak pokračuj další položkou, pokud existuje.

- Existuje-li implicitní cesta (default route), použij tuto cestu. Jinak
- Skonči chybou. Generuj ICMP zprávu „Destination unreachable“.

3.3 Co je to firewall

Firewall je zařízení, které je umístěno na hranicích sítí s různou mírou důvěryhodnosti a které má za úkol kontrolovat a omezovat síťový provoz mezi těmito sítěmi podle administrátorem definovaných pravidel.

Firewally můžeme rozdělit na *paketové* a *aplikační*, paketové mohou být *stavové* nebo *bezstavové*. Aplikačním firewallům se také říká *proxy*.

Paketové firewally jsou obecně méně bezpečné než aplikační firewally, protože kontrolují pouze hlavičky paketů, nikoli data samotná. Umožňují průchod paketům podle portu, na který přicházejí, a nekontrolují, zda pakety odpovídají protokolům, které jsou na daných portech očekávány.

Aplikační firewally, tzv. proxy, fungují pro konkrétní protokoly, nejčastěji HTTP a FTP. Dříve se zřizovaly kvůli kešování obsahu, v současné době je jejich hlavní význam v tom, že dokáží zkontrolovat, zda příchozí pakety odpovídají komunikačnímu protokolu pro daný port. Každý přicházející paket rozbalí, zkontrolují a zabalí nebo zahodí.

Některé firewally fungují jako paketové a aplikační současně, některé mají pouze jednu z těchto funkcí. Linuxový firewall, který je zabudován do jádra operačního systému, je pouze paketový, proto je vhodné jeho funkčnost doplnit ještě proxy serverem, který může běžet i na jiném stroji.[20]

Vzhledem k rozsahu této práce není možné se zabývat podrobně i nastavením proxy serveru, proto se v příštím textu soustředíme především na router a paketový firewall obsažený v jádru OS Linux.

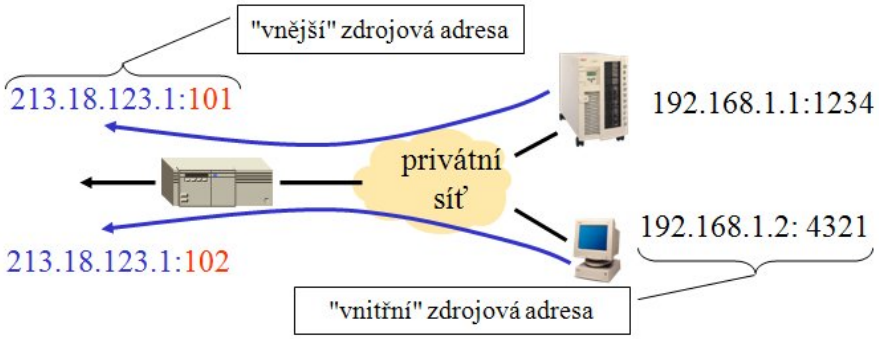
3.3.1 Dynamický překlad adres

Důležitou vlastností paketových firewallů je schopnost dynamického[21] překladu adres (NAT – Network Address Translation).

Tato funkce byla zavedena původně z jiného důvodu: ukázalo se totiž, že IP adres není dostatek pro všechny, proto nemůže být každé zařízení přímo přístupné z internetu a mít vlastní veřejnou IP adresu. Problém je možno obejít tak, že do Internetu se vystaví pouze jeden počítač s veřejnou IP adresou jako brána a ostatní počítače umístěné ve vnitřní síti mají IP adresy z neveřejného rozsahu. Počítač na perimetru pak překládá požadavky počítačů ve vnitřní síti tak, jako by je vysílal do internetu on sám. Současně uchovává v tabulce seznam, který počítač ve vnitřní síti vyslal požadavek, a odpověď z Internetu

[20] Viz Perkins. *Firewally*, s. 3–13.

[21] Kromě dynamického překladu adres existuje také statický překlad, při kterém však jsou uzly vnitřní sítě dostupné z vnější sítě.



Obrázek 6: PAT - Port Address Translation

pošle zpět na tento počítač. Tím se před světem skryje celá síť a vystupuje jako jedno velmi zatížené zařízení s jednou IP adresou.

Postupem času se ukázalo, že pokud je překlad adres prováděn dynamicky, funguje jako dobré zabezpečení. Útočníkovi se totiž nepodaří se z Internetu připojit na žádný počítač ve vnitřní síti. NAT ale také zabraňuje připojit se na počítač správcům sítě a těm, kdo to potřebují. Tento problém je možno řešit přesměrováním portů, tzn. firewall přesměruje všechny požadavky, přicházející na určitý port, na pevně daný port počítače ve vnitřní síti. Toto řešení se nazývá PAT (Port Address Translation) a je znázorněno na obrázku 6.[22]

3.4 Firewall a bezpečnost

Použití firewallů úzce souvisí s bezpečností sítí. Proto by v návaznosti na toto téma bylo možno rozepsat se o těch, kdo mohou mít zájem firewall překonat a narušit bezpečnost sítě, kterou firewall chrání. V rámci rozsahu této práce není možné se podrobně věnovat psychologii útočníků[23] nebo detailně popsat různé způsoby napadení systémů[24]. Z možných útoků popíšeme velmi stručně pouze ty nejdůležitější, které je přímo třeba vzít v úvahu při konfiguraci firewallu.

3.4.1 Útoky na firewall

Hlavními útoky, s kterými bychom měli při konfiguraci firewallu počítat, jsou útoky směřující k odepření služby. Jedná se o útoky, které mají za následek vyčerpání síťových zdrojů počítače, nazývají se také DoS útoky (Denial of Service). Patří mezi ně především záplavy (floods) – ICMP floods, UDP floods, Smurf (Šmoula) a distribuované DoS útoky (DDoS). Speciální kategorií těchto útoků jsou TCP/IP útoky[25]. Mezi tyto útoky patří ping of death (ping smrti), teardrop (slza) a záplavy SYN.[26]

[22] Viz Peterka. *Rodina protokolů TCP/IP, část 3*.
[23] O tom podrobněji např. Perkins, *Firewally*, s. 27–33.
[24] Viz např. Hatch. *Linux. Hackerské útoky* nebo Endorf. *Hacking*.
[25] Tyto útoky jsou dokladem nedokonalosti návrhu TCP/IP.
[26] Hatch. *Linux. Hackerské útoky*, s. 265–272.

Zaplavování spočívá v tom, že se zašle silný proud IP paketů konkrétnímu hostiteli, takže se spotřebuje veškerá jeho přenosová kapacita, a tím se blokuje ostatní provoz. Distribuované útoky DDoS jsou útoky, kdy velké množství počítačů současně provádí útok DoS na jednoho hostitele.

Aby mohl útočník provést DDoS útok, musí získat vládu nad celou sítí počítačů, kde si nainstaluje pomocné programy pro vzdálenou koordinaci a řízení útoku.

Při útoku smurf je do sítě zaslán paket ICMP ECHO REQUEST (ping) se zfalšovanou adresou odesilatele, zpravidla z vnitřní sítě, takže všechny stroje odpoví na ping a tohoto hostitele zaplaví. Síť tak zesílí původní útok.

Ping smrti je ICMP paket větší než 65536 bytů, dokázal shodit např. linuxové jádro starší než 2.0.24. Teardrop byl fragmentovaný paket, který nebylo možno správně defragmentovat, od verze 2.0.32 již linuxové jádro obsahuje záplatu, která tento problém řeší.

Záplavy SYN se provádějí tak, že je hostiteli zasíláno velké množství žádostí o ustanovení TCP spojení, ale nenásledují další pakety potvrzující sestavení tohoto spojení. Žádosti o sestavení TCP spojení se ukládají do speciální fronty, ze které jsou po sestavení spojení nebo po uplynutí určitého času odstraněny. Pokud ale žádostí přichází velký počet, fronta se zaplní a další spojení není možno navazovat.

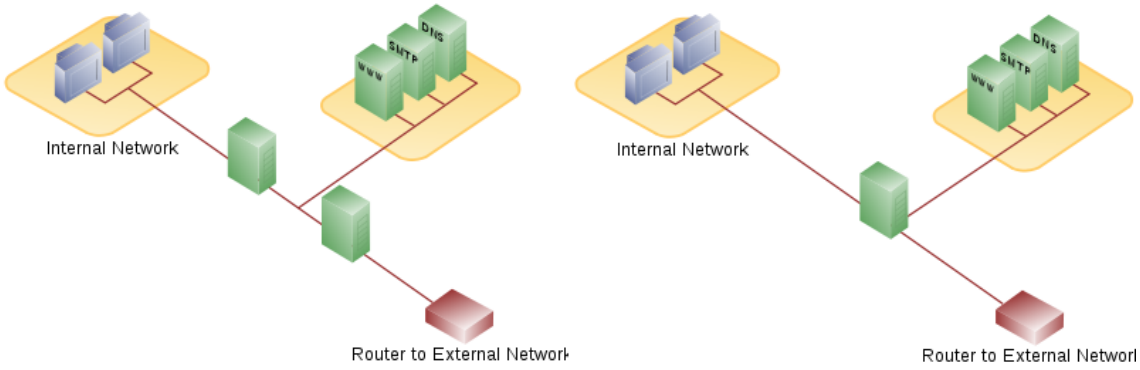
3.4.2 Ingresní (vstupní) a egresní (výstupní) filtrace

Existují některé IP adresy, které by za normálních okolností neměly překračovat hranici mezi Internetem a sítí organizace. Termín ingresní filtrace se vztahuje k procesu filtrace zjevných pseudo adres vstupujících do sítě, zatímco engresní filtrace se týká odchozího provozu. Jednoduchým příkladem může být, že podniky s veřejným IP prostorem na internetu by měly odmítat všechen provoz dělající si nárok, že je z jejich adresního prostoru, ale přitom přichází z vnějšího Internetu. A podobně, tentýž podnik by měl filtrovat všechen odchozí provoz a povolit pouze pakety se zdrojovými adresami pocházejícími z vlastního IP prostoru, nic jiného by nemělo opouštět jeho hranice.[27]

3.4.3 Struktura sítě

Bezpečnost sítě je možno zvýšit zřízením Demilitarizované zóny (DMZ). DMZ je část sítě, kam se umísťují servery, které jsou přístupné z vnějšího Internetu. Tím se snižuje riziko napadení sítě. Prakticky je možno DMZ realizovat dvěma způsoby: buď má firewall tři síťové karty, jedna je připojena do důvěryhodné chráněné vnitřní sítě, jedna do DMZ a jedna do vnější nedůvěryhodné sítě, nebo jsou použity dva firewally, první mezi vnitřní sítí a DMZ a druhý mezi DMZ a vnějším Internetem. Obě situace jsou znázorněny na obrázku 7.

[27] Viz Endorf. *Detekce a prevence počítačového útoku*, s. 80.



Obrázek 7: Možné uspořádání sítě s DMZ

3.5 Shrnutí

V tomto oddíle jsme si stručně vysvětlili fungování zásobníku TCP/IP, zmínili jsme se o modelu ISO OSI a vysvětlili jsme si, k čemu se používá router a firewall. Pro srovnání jsme zmínili i další pojmy jako gateway, switch a bridge. V souvislosti s firewally nelze opominout bezpečnostní hledisko, proto jsme si popsali některé způsoby narušení bezpečnosti, se kterými bychom měli při konfiguraci firewallu počítat, a zmínili jsme základní opatření, která by se měla provést na rozhraní mezi vnitřní sítí a Internetem.

4 Router a firewall na Linuxu

4.1 Stručná historie Linuxu

U zrodu Linuxu stál Linus Benedict Torvalds, student informatiky na Helsinské univerzitě, který v roce 1991 ohlásil vytvoření základu operačního systému pro architekturu PC, postavenou na procesorech Intel. Spousta vývojářů z celého světa přispěla svými myšlenkami, takže se Linux stal použitelným. Cílem bylo vytvořit volně šiřitelný operační systém s vlastnostmi Unixu.

Linus Torvalds využil v Linuxu nástroje, které již dříve existovaly jako součást projektu GNU, založeného Richardem Stallmanem. (Často se používá pro Linux termín GNU/Linux, aby se zdůraznily obě složky projektu).

V roce 1994 byla vydána první stabilní verze 1.0, která měla následující vylepšené vlastnosti:

- podpora TCP/IP sítí
- síťové programové rozhraní kompatibilní s BSD sokety
- ovladače síťových karet
- podpora SCSI rozhraní
- přepracovaný souborový systém (Enhanced file system)

Od verze 1.2 kernel obsahoval IP firewall, který byl obsluhován pomocí utility ipfwadm.[28]

Verze 2.0 z roku 1996 už byla spustitelná na dalších hardwarových platformách, např. PowerPC, Sun, Sparc nebo Motorola a uměla využívat i víceprocesorové stroje.

Verze 2.2 vyšla v lednu 1999 a jednalo se o vyspělý operační systém, který obsahoval v jádru nový paketový filtr – firewall ipchains, který nahradil předchozí firewallové služby.

Verze 2.4 přinesla další významná vylepšení:

- podpora 64GB operační paměti na intelovských strojích
- efektivnější multiprocessing a multithreading
- lepší podpora USB a IEEE 1394 (FireWire)
- souborový systém ReiserFS s podporou žurnálování
- kvalitní řešení firewallu (iptables)

V současné době je k dispozici jádro 2.6.[29]

4.1.1 Open source a bezpečnost

Projekt GNU/Linux byl od začátku vyvíjen jako otevřený software, na jeho vývoji se účastnila celá komunita programátorů propojených přes Internet. Jejich zájmem bylo a je vytvářet kvalitní software, který pracuje tak, jak má, a neobsahuje bezpečnostní díry. Spolupráci na otevřeném software popsal Eric Raymond ve článku *„Katedrála a tržiště“*[30], kde popsal výhody otevřeného softwaru. Tento software nemůže používat tzv. „security skrze obscurity“, tj. nelze doufat, že bezpečnostní chyby zůstanou neobjevené, a proto není třeba se jimi zabývat.

Ve skutečnosti často ten, kdo objeví nějakou bezpečnostní díru v programu, současně s upozorněním na tuto zranitelnost zašle i záplatu, která problém řeší.

Uživatel programu také není závislý pouze na jednom dodavateli, který jediný může program upravovat a dokonce by do něj teoreticky mohl naprogramovat vlastní zadní vrátka do uživatelova systému.

Bezpečnost softwaru ale naneštěstí je jen malou částí bezpečnosti systému, je třeba programy také správně nastavit a v neposlední řadě je třeba brát v úvahu i lidský faktor.

4.2 Routování v Linuxu

Funkce routeru je v linuxu zařízena jednak v jádru povolením forwardování paketů, jednak routovací tabulkou.

[28] Domovská stránka projektu se nachází na adrese http://www.xos.nl/resources/ipfwadm/.

[29] Viz Kučera. *Historie Linuxu pěkně od začátku*. Podrobněji o vzniku Linuxu např. Raymond. *Umění programování v Unixu*, s. 64–72.

[30] Česky k dispozici např. na adrese http://www.zvon.org/ZvonHTML/Translations/cathedral-bazaar/front_cs.html

Forwardování paketů je v linuxu implicitně zakázáno a povoluje se příkazem:

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

V distibuci Slackware, kterou jsme použili při instalaci firewallu, se forwarding zapíná startovacím skriptem *rc.forwad*.

Routovací tabulku můžeme vypsat příkazem `route`, jak je ukázáno v následujícím výpisu:

```
beeblebrox:~ # route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
172.19.0.0      *               255.255.0.0     U~0      0        0 eth0
link-local      *               255.255.0.0     U~0      0        0 eth0
loopback        *               255.0.0.0       U~0      0        0 lo
default         asa.plbohnice.c 0.0.0.0         UG       0      0         0 eth0
```

4.2.1 Statické routování

Nastavení routování provádí systém na základě nastavení síťových rozhraní, pokud z nějakého důvodu routovací tabulka není nastavena správně, je možno ji editovat opět pomocí příkazu route. Protože takto upravenou tabulku si systém uchovává pouze do následujícího restartu, je třeba potřebný příkaz zapsat do některého startovacího skriptu, např. v distribuci Slackware se k tomuto účelu používá soubor */etc/rc.d/rc.local*.

Ruční úprava routovací tabulky se provádí příkazem route, např.:[31]

```
#route del -net 192.168.27.0 netmask 255.255.255.0 dev eth1
#route add -net 192.168.24.0 netmask 255.255.252.0 dev eth1
#route del -net 62.0.0.0 netmask 255.0.0.0 dev eth0
```

4.2.2 Dynamické routování

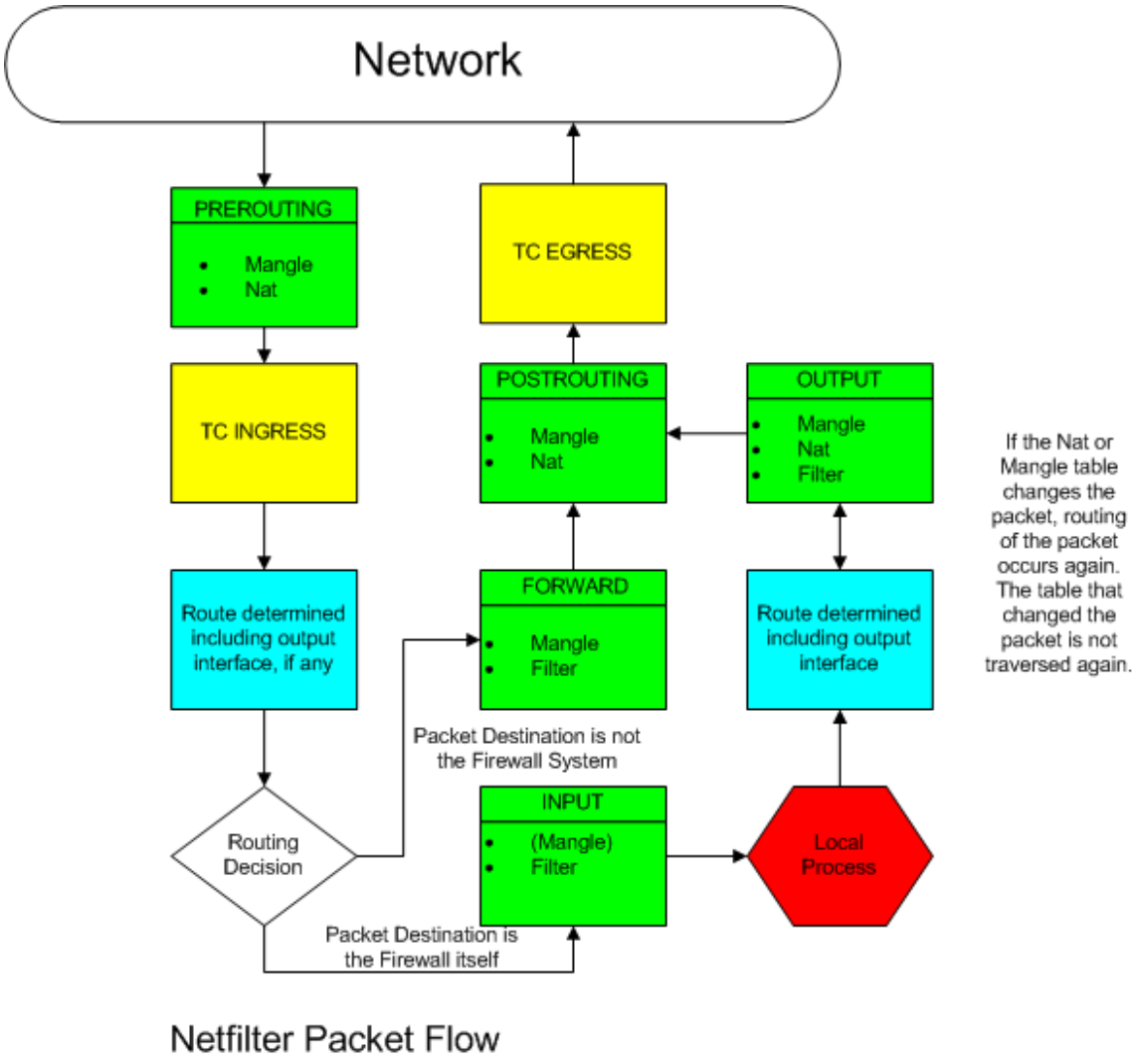
Při dynamickém routování jsou routovací tabulky upravovány programem podle aktuálního stavu sítě. Takovým programem může být v Linuxu např. program *routed* nebo *quagga*. Protože v našem konkrétním případě bylo použito pouze jedno připojení do Internetu, využili jsme statické routování. Rozsah této práce neumožňuje věnovat dynamickému routování více místa.

4.3 Netfilter

Netfilter je rámec, který poskytuje linuxovému jádru schopnost přijímat síťové pakety a manipulovat s nimi.

Projekt odstartoval Rusty Russell v r. 1988, v r. 1999 založil skupinu Netfilter Core Team. Netfilter je licencován pod GPL a byl včleněn do jádra 2.3.

[31] Podrobná dokumentace k příkazu route je, jak je v Linuxu obvyklé, k dispozici ve formě manuálových stránek, které se vyvolají příkazem `man route`.



Obrázek 8: Vztah mezi routováním a Netfilterem

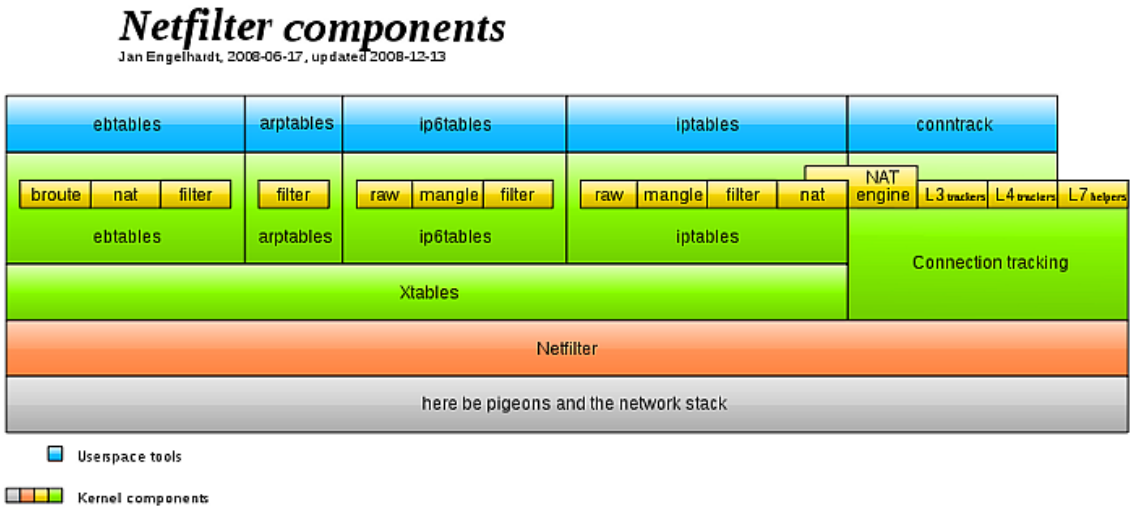
Předsedové Core teamu byli od r. 2003 Herald Welte[32], od r. 2007 Patric McHardy.

Ip_tables, ip6_tables, arp_tables a ebtables jsou nejdůležitější jaderné moduly, které využívají Netfilter. Tyto moduly poskytují systém pro definování pravidel firewallu založený na tabulkách, pomocí něhož je možno filtrovat nebo modifikovat pakety. Tabulky mohou být administrovány pomocí nástrojů uživatelského prostoru iptables, ip6tables, arptables a ebtables.

4.3.1 Vztah mezi routováním a programem Netfilter

Obrázek 8 ukazuje vztah mezi routováním a programem Netfilter. Základní princip je, že routování rozhoduje, kam má být paket poslán, zatímco firewall

[32]Welte se zapsal do historie tím, že v r. 2004 obdržel historický výrok německého soudu proti Sitecom Germany, který umísťoval Netfilter/iptables do svých routerů a neřídil se podmínkami GPL. (Viz např. *The German GPL Order - Translated*).



Obrázek 9: Netfilter: komponenty

určuje, zda paket smí na toto místo jít. Firewall může ovlivnit routování způsoby, které budou popsány dále.[33]

Modré obdélníky na obrázku 8 označují místa v procesu, kde dochází k routovacím rozhodováním, zelená pole znamenají místa, kde dochází k zpracování jednotlivými tabulkami Netfilter/iptables. Žlutě jsou označena místa vstupu a výstupu dat. Tzv. Politika TC (Traffic Control) Ingress (ingresní kontroly) se používá ke snížení objemu příchozího provozu, TC Egress (egresní kontroly) k regulaci odchozího provozu.[34]

Pro konfiguraci našeho firewallu jsou důležité pouze iptables, ostatní moduly jsme nevyužili. Proto se v dalším popisu soustředíme hlavně na ně.[35]

4.3.2 NAT v Linuxu

V Linuxu je možno použít dva typy NAT, buď *Fast-NAT* nebo *Netfilter-NAT*. Fast-NAT je implementován v jádru v rámci kódu pro IP routování, zatímco Netfilter-NAT je implementován v jádru v kódu Netfilteru. Fast-NAT je o mnoho rychlejší než Netfilter-NAT, ale neumí sledovat spojení (connection tracking). Důsledkem toho je, že neumí dostatečně SNAT (Source NAT) pro celou síť a není schopen používat NAT pro složitější protokoly jako FTP, IRC a další, které Netfilter zvládá bez problémů.[36]

4.4 Firewall iptables

Iptables se přidržují základního schématu, který přinesl program ipfwadm: seznam pravidel, z nichž každé specifikuje, co hledáme v paketu a co udělat

[33] Obrázek je převzat ze stránek o programu Shorewall, což je grafická nadstavba pro Netfilter/iptables. Eastep *Shorewall and Routing*.

[34] Podrobněji jsou ingresní a egresní kontroly popsány v oddíle 3.4.2 a 4.4.5.

[35] Egresní a ingresní kontroly jsou také implementovány pomocí iptables.

[36] Viz Andreasson. *Iptables Tutorial 1.2.2*.

s takovým paketem. Ipchains navíc dodaly koncept řetězců pravidel a iptables řetězce dále rozšířily na tabulky. Jedna tabulka se využívala na rozhodování pro NAT, další pro rozhodování, jak paket filtrovat. Navíc byly změněny tři body, na kterých je prováděno filtrování, takže jakýkoli paket prochází pouze skrz jeden filtrační bod.

Toto rozdělení dovolilo iptables používat informace, které o paketu získala vrstva connection tracking (dříve tato informace byla svázána s NAT). Tím iptables převyšují svého předchůdce ipchains, protože mají schopnost monitorovat stav spojení a přesměrovat, upravovat nebo zastavit datové pakety na základě stavu spojení, nejen podle zdroje, cíle nebo datového obsahu paketu.

Proto se firewallu používajícímu iptables říká stavový firewall na rozdíl od ipchains, které dokáží vytvořit pouze bezstavový firewall.

4.4.1 IP filtrování

IP filtr funguje převážně v 2. vrstvě TCP/IP modelu. Iptables však (stejně jako většina současných IP filtrů) jsou schopny také pracovat ve vrstvě 3. Podle definice však IP filtrování je filtrování na druhé vrstvě.

Pokud by se IP filtrování striktně chovalo podle definice, bylo by možno pakety filtrovat pouze podle údajů v IP hlavičce (viz obr. 3). Avšak protože implementace iptables se nedrží přesně definice IP filtrování, je možno filtrovat i podle údajů ležících v hlubších vrstvách paketu (TCP, UDP) i ve vyšších vrstvách (MAC adresa).

IP filtrování však nedokáže sledovat proud dat nebo skládat data paketů dohromady. Je to proto, že tyto akce spotřebovávají mnoho paměti a procesorového výkonu. K tomu účelu jsou navrženy proxy (např. Squid). Iptables dokáží sledovat pakety a určit, zda jsou ze stejného proudu (pomocí pořadových čísel, čísel portů atd.). Tomu se říká sledování spojení (connection tracking) a pomocí něj je možné provádět překlad adres podle zdroje (SNAT) a podle cíle (DNAT) stejně jako zjišťování stavu spojení.

Hlavní termíny používané ve spojitosti s IP filtrováním a jejich význam shrnuje tabulka 4.

Iptables pracují s hlavičkami internetové a transportní vrstvy. Je sice možné pomocí nich provádět základní filtrování na aplikační a síťové vrstvě, ale iptables pro to nebyly navrženy a nejsou k tomuto účelu vhodné.[37]

4.4.2 Stavový stroj

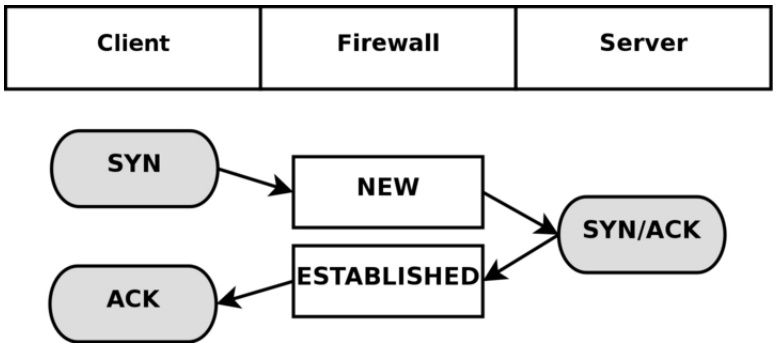
Stavový stroj je zvláštní část iptables, která by se vlastně neměla nazývat stavový stroj, protože je to systém na sledování spojení (connection tracking machine). Avšak většina lidí používá pojem stavový stroj. I když to jsou víceméně synonyma, může jejich zaměňování být matoucí.

Sledování spojení (connection tracking) se provádí tak, že Netfilter zná stav určitého spojení. Firewally, které toto implementují, se obecně nazývají stavové firewaly. Stavový firewal je obecně mnohem bezpečnější než bezstavový,

[37] Viz Andreasson. *Iptables Tutorial 1.2.2.*

Drop/Deny	Paket je smazán a nic dalšího se nestane.
Reject	Jako Drop/Deny, kromě toho, že je také zaslána hostiteli odesílajícímu paket zpráva, že paket byl zahozen.
State	Specifický stav paketu ve srovnání s celým proudem paketů. Např. první paket, který dorazí na firewall, je označen new (odpovídá SYN paketu v TCP spojení), nebo část již ustanoveného spojení – established.
Chain	Chain (řetězec) obsahuje soubor pravidel, která se aplikují na paket procházející řetězcem. Každý řetězec má specifický účel, a také specifickou oblast, ve které se aplikuje (tj. jen forwardované packety nebo jen pakety určené pro tohoto hostitele).
Table	Každá tabulka má specifický účel, v iptables jsou čtyři tabulky: raw, nat, mangle a filter.
Match	Toto slovo může mít ve spojitosti s IP filtrováním dva rozdílné významy. 1. jednotlivá shoda, která říká pravidlu, že tato hlavička musí obsahovat tuto informaci. Např. `--source` match nám říká, že zdrojová adresa musí být z určitého síťového rozsahu nebo adresy. 2. pokud celé pravidlo dosáhne shody (match), uplatní se instrukce jump nebo target.
Target	Obecně existuje target (cíl) pro každé pravidlo. Pokud se pravidlo plně shoduje s paketem (match), target nám říká, co dělat s paketem. Např. zda se má zahodit, přijmout nebo NATovat atd.
Rule	Pravidlo je množina matchů (shod) nebo několik matchů dohromady s jediným targetem.
Ruleset	Kompletní skupina pravidel vložených do implementace IP filtru. Většinou jsou zapsány v nějakém konfiguračním souboru.
Jump	Instrukce jump (skok) je stejná jako target s výjimkou toho, že místo jména cíle zapíšete název jiného řetězce. Pokud se pravidlo shoduje s paketem, paket bude zaslán na tento řetězec a bude zpracován, jak je to obvyklé v tomto řetězci.
Connection tracking	Firewall, který implementuje connection tracking, je schopen sledovat spojení/proudy. To ale má často dopad na využití procesoru a paměti, avšak takovýto firewall je mnohem bezpečnější.
Accept	Přijme (accept) paket a nechá ho projít pravidly firewallu. Je to opačná instrukce než Drop/Deny nebo reject.
Policy	Jsou dva druhy politik: řetězcové politiky, které říkají, jak má firewall implicitně zacházet s paketem, pokud pro něj nenalezne vhodné pravidlo. Druhý typ politik je bezpečnostní politika zapsaná v nějaké dokumentaci. Její důkladné studium by mělo předcházet implementaci firewallu.

Tabulka 4: Základní termíny IP filtrování



Obrázek 10: Stav spojení TCP

protože umožňuje napsat mnohem přesnější soubor pravidel – rule-set.

V iptables může paket nabývat ve vztahu k sledovanému spojení čtyři různé tak zvané stavy: NEW, ESTABLISHED, RELATED a INVALID. Pravidlem s parametrem `--state` můžeme snadno kontrolovat, kdo nebo co má povoleno iniciovat nové spojení.

Sledování spojení je prováděno speciálním systémem v jádru nazývaným *conntrack*. Conntrack může být načten buď jako modul, nebo jako vnitřní část jádra samotného. Většinou však potřebujeme specifičtější sledování spojení, než může poskytnout implicitní conntrack. Proto existují také části systému conntrack, které umějí zpracovávat TCP, UDP a ICMP protokoly. Tyto moduly vezmou specifické informace z paketů, takže jsou schopny vystopovat každý proud dat. Informace, které conntrack shromáždí, jsou pak použity k určení, ve kterém stavu se proud dat nachází. Např. UDP proudy jsou obecně jednoznačně určeny zdrojovou IP adresou a portem a cílovou adresou a portem.

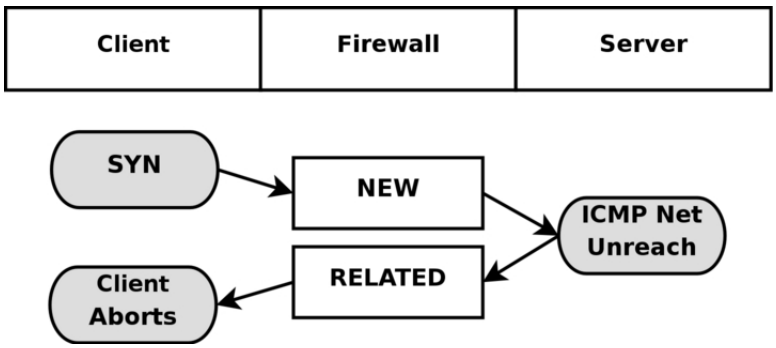
V minulých jádrech byla možnost vypnout defragmentaci. Avšak protože iptables a Netfilter mají funkci sledování spojení, tato volba byla z jádra odstraněna. Důvodem je, že sledování spojení by bez defragmentace nepracovalo správně, takže defragmentování paketů bylo přičleněno k systému conntrack a provádí se automaticky. Lze jej vypnout pouze současně se sledováním spojení.

Sledování spojení se provádí vždy v řetězci PREROUTING, kromě paketů generovaných lokálně, ty jsou obsluhovány v řetězci OUTPUT. To znamená, že iptables přepočítávají stav v řetězci PREROUTING. Pokud pošleme úvodní paket proudu, stav je nastaven na NEW v řetězci OUTPUT, odpověď od druhého stroje má pak stav ESTABLISHED. Pokud první paket není od nás, stav NEW je nastaven v řetězci PREROUTING. Takže všechny změny a přepočty jsou prováděny v řetězcích PREROUTING and OUTPUT tabulky *nat*. Situace pro TCP spojení je zachycena na obrázku 10.

Velmi důležitou funkcí ICMP je skutečnost, že je využíván k informování hostitele o tom, co se stalo s určitým UDP a TCP spojením nebo pokusem o spojení. Z tohoto důvodu jsou ICMP odpovědi často označovány jako RELATED – vztahují se k původnímu spojení nebo pokusu o spojení.

Příkladem je třeba ICMP host unreachable nebo ICMP Network unreachable. Tyto pakety by vždy měly být vždy doručeny původnímu hostiteli,

který se neúspěšně snaží připojit na nějaký jiný počítač, ale požadovaná síť něbo počítač může být mimo provoz, takže poslední router snažící se dosáhnout požadovanou sít odpoví ICMP zprávou, která to oznamuje. To ukazuje obrázek 11.



Obrázek 11: Stav spojení ICMP

4.4.3 Konfigurace firewallu pomocí iptables

Název programu *iptables* je odvozen od tabulek (můžeme si je představit jako jakési katalogy), kterými se paket řídí při průchodu firewallem. Každá z tabulek má své položky – řetězce (chain).

Existují tři základní tabulky: *filter* (výchozí), *nat* a *mangle*.[38]

Tabulka filter má tři řetězce: INPUT, FORWARD a OUTPUT. Pokud paket vstupuje do firewallu a nepokračuje dál, je na něj aplikován řetězec INPUT, pokud má být předán dál, je aplikován řetězec FORWARD, pokud byl paket generován na firewallu (neprošel tedy pravidlem FORWARD) a vystupuje ven, aplikuje se na něj pravidlo OUTPUT.

Tabulka nat má také tři řetězce: PREROUTING, POSTROUTING a OUTPUT. Pomocí řetězce PREROUTING můžeme měnit cílovou adresu paketů (DNAT – Destination NAT), což se používá hlavně při forwardování portů do vnitřní sítě, která není přístupná z Internetu. POSTROUTING naopak upravuje odchozí spojení (Source NAT – SNAT), také se nazývá *masquerade* (tzv. maškaráda). To se používá, pokud za jednu veřejnou IP adresu chceme skrýt více počítačů ve vnitřní síti. Řetězec OUTPUT v tabulce nat se používá ještě před modifikací odchozích paketů.

Tabulka mangle má všech pět řetězců: INPUT, FORWARD,OUTPUT, PREROUTING a POSTROUTING. Používá se na úpravy hlaviček paketů, např. TTL, TOS atd.[39]

Jak je v linuxu obvyklé, manuálové stránky jsou dostupné příkazem `man iptables`. Syntaxe nástroje pro ovládání modulu ip_tables je

```
iptables [tabulka] [akce] [chain] [ip_část] [match] [target] [target_info]
```

[38] Existuje ještě tabulka *raw*, která se používá hlavně k testovacím účelům.
[39] Viz Botoš. *Vše o iptables: úvod*.

-A	--append	- Přidání nového pravidla na konec řetězce
-D	---delete	- Smaže pravidlo (buď ho zadáte ve tvaru, v němž jste ho přidávali, nebo použijete jeho číslo, to získáte rozšířenou volbou –lin.).
-R	--replace	- Nahradí číslo pravidla jiným pravidlem
-I	--insert	- Vložení nového pravidla na začátek řetězce
-L	--list	- Vypsání všech pravidel v řetězci. Pokud není zadán řetězec, vypíšou se všechny řetězce + jejich pravidla
-F	--flush	- Vyprázdní všechna pravidla v řetězci (to samé, jako kdybyste to dělali po jednom)
-N	--new-chain	- Vytvoříme si vlastní řetěz
-X	--delete-chain	- Smažeme si vlastní řetěz (nejde smazat výchozí)
-P	--policy	- Výchozí politika řetězce
-E	--rename-chain	- Přejmenování vlastního řetězce

Tabulka 5: Přípustné hodnoty akce v příkazech iptables

Parametr tabulka se uvádí přepínačem -t, za kterým může následovat *filtr*, *nat*, *mangle* nebo *raw*, implicitně bez uvedení tohoto parametru je nastavena tabulka filtr.

Přípustné akce jsou shrnuty v tabulce 5.

Zdrojová a cílová IP adresa se definuje pomocí přepínačů: pro zdrojovou IP -s (--src, --source) a pro cílovou IP -d (--dst, --destination). Pokud nepoužíváme výchozí masku, můžeme IP adresu napsat také ve tvaru 192.168.0.1/24 či 192.168.0.1/255.255.255.0.

Pro zjištění, ze kterého rozhraní paket přišel či které se právě snaží opustit, použijeme přepínače -i (--in-interface) pro vstupní rozhraní, anebo -o (--out-interface) pro rozhraní, které se pokouší opustit.

Užitečný je parametr -m multiport, pomocí kterého můžeme v jednom pravidle zadat až 15 portů oddělených čárkou. Přípustné volby jsou: zdrojové porty, cílové porty nebo oba porty současně (--destination-ports, --source-ports nebo --ports). Tato možnost nebyla v našem konfiguračním souboru *firewall.conf* využita, ačkoli by její použití zmenšilo počet pravidel a tím zvýšilo přehlednost.

Podrobný popis možností nástroje iptables zde není možno z důvodu rozsahu práce podat, je možno jej nalézt v literatuře[40] nebo v manuálových stránkách.

4.4.4 Ukládání a načítání konfigurace iptables

K ukládání a načítání konfigurace iptables je možno použít shellový skript nebo přímo příkaz iptables-save a iptables-restore. Tyto příkazy mají oproti shellovému skriptu výhodu, že jsou rychlejší, načtení všech pravidel pomocí iptables-restore se totiž provádí jedním krokem, kdežto při načítání ze skriptu dochází k tomu, že vždy při načítání řádky se přečtou do paměti nejprve všechna pravidla uložená v paměťovém prostoru Netfilteru,

[40] Např. Botoš. *Vše o iptables* nebo ještě podrobnější Andreasson. *Iptables Tutorial 1.2.2.*

tato sada pravidel se upraví a opět vloží do Netfilteru najednou.

Pokud tedy má skript např. 50 pravidel, tato operace se provede 50krát. V případě několika tisíc pravidel v rulesetu už dochází ke značným časovým prodlevám, a je proto nanejvýš žádoucí používat příkazy `iptables-save` a `iptables-restore`.

4.4.5 Ingresní a egresní kontroly

Mnoho hackerských útoků spoléhá na falšování zdrojové IP adresy. Egresní filtrování je nejdůležitější metodou, jak tomu zabránit. Routery by měly kontrolovat všechen odchozí provoz a propouštět jej pouze v případě, že adresa uvedená v paketu odpovídá adrese vnitřní sítě. Následující kód ukazuje vzorové ingresní a egresní filtry.[41]

```
#!/bin/sh
# egress_filtering_netfilter.sh
# Sample Ingress/Egress filters with iptables on
# machine that also acts as a~forwarding gateway.
# Change IP networks, salt to taste.

# Internal network is assumed to be eth0
internal_net=192.168.5.0/24
# External network is assumed to be eth1
my_ip_addr=192.168.4.2/32

# Egress Filters: Allow only our internal IPs and
# external interface addrs out of eth1
/sbin/iptables -A OUTPUT -o eth1 -s $my_ip_addr -j ACCEPT
/sbin/iptables -A OUTPUT -o eth1 -s $internal_net -j ACCEPT

# Ingress Filters: Allow only our internal IPs and
# external interface addrs in from eth1
/sbin/iptables -A INPUT -i eth1 -d $my_ip_addr -j ACCEPT
/sbin/iptables -A INPUT -i eth1 -d $internal_net -j ACCEPT

# Egress/Ingress Filters on eth0:
# Allow only traffic to/from the internal net through eth0
/sbin/iptables -A OUTPUT -o eth0 -d $internal_net -j ACCEPT
/sbin/iptables -A INPUT -i eth0 -s $internal_net -j ACCEPT

# Block clearly-spoofed packets
# Deny any restricted ip networks from traversing Carbon at all
for badnet in   127.0.0.1/32  10.0.0.0/8  172.16.0.0/12 \
              192.168.0.0/16 224.0.0.0/4    240.0.0.0/5
do
/sbin/iptables -A INPUT  -i eth0 -s $badnet -j DROP
/sbin/iptables -A OUTPUT -o eth0 -s $badnet -j DROP
/sbin/iptables -A INPUT  -i eth1 -s $badnet -j DROP
/sbin/iptables -A OUTPUT -o eth1 -s $badnet -j DROP
done
```

4.5 Shrnutí

V této části jsme se zaměřili na specifika nastavení routeru a firewallu na operačním systému Linux. Po stručném seznámení s historií OS Linux a vývojem

[41] Viz Hatch. *Linux. Hackerské útoky*, s. 275–277.

implementace firewallu v jádru systému jsme si popsali Netfilter a nástroj pro jeho nastavování iptables. Popsali jsme si, jak se iptables používají a jak s jejich pomocí můžeme implementovat ingresní a egresní kontroly.

5 Konfigurace routeru a firewallu v prostředí konkrétní organizace

V této části popíšeme konfiguraci konkrétního linuxového firewallu nainstalovaného v Psychiatrické léčebně Bohnice (PLB). V současné době již tento linuxový firewall není v provozu a i celá síť léčebny byla přečíslována, proto je možno zveřenit v této práci všechny konfigurační soubory a nastavení, což by jinak z bezpečnostních důvodů nebylo možné.

5.1 Vývoj nastavení firewallu v PLB

Původně byl v organizaci používán Floppyfw[42], což je linuxový router a firewall s pokročilými vlastnostmi, který se vejde na jednu disketu. Bezdiskový počítač nabootuje z diskety, kterou po startu systému odpojí, a veškerá data i program jsou uloženy v paměti. Jeho konfiguraci provedli mí předchůdci a nějakou dobu jsem ho spravovala.

Postupně přibývalo požadavků na konfiguraci firewallu a kvůli úpravám bylo třeba stále opravovat data na disketě, systém neumožňoval žádné logování ani pokročilejší bezpečnostní funkce, navíc se v PLB se objevovaly požadavky na instalaci IDS (Intrusion Detection System).

Proto byl Floppyfw nahrazen novým firewallem, jehož konfigurace je předmětem této bakalářské práce. Konfigurační soubory zpočátku vycházely z nastavení Floppyfw a postupně jsme je upravovali a vylepšovali. Byl nainstalován na distribuci Slackware[43]. Tuto distribuci jsme zvolili proto, že jsme s ní již měli nějaké zkušenosti a vyhovovaly nám zvláště startovací skripty, které jsou ve stylu BSD (narozdíl např. od distribuce Red Hat, která používá startovací skripty ve stylu System V).[44]

Tento firewall byl později nahrazen komerčním firewallem od firmy Cisco. Důvodem výměny byla cílená dotace ministerstva na bezpečnost. Na počítači s linuxovým firewallem jsme sice rozběhli IDS Snort, ale jeho vyladění se nepodařilo dokončit k úplné spokojenosti. Od nového firewallu ASA5510 SSM se očekávalo zavedení pokročilejšího IPS (Intrusion Prevention System) a umožnění připojování přes VPN (Virtual Private Network), které se na linuxovém firewallu spíše z časových důvodů nepodařilo rozběhnout.

Tato očekávání nový Cisco firewall částečně splnil, výhodou je, že máme k dispozici odbornou pomoc při jeho nastavování a fungující VPN. Pokrokem je také Intrusion Prevention System (IPS), který bezpečnostní incidenty zadržuje (na rozdíl od IDS, které incidenty loguje, tento typ IPS nebezpečný paket

[42]Domovská stránka projektu je http://www.zelow.no/floppyfw.

[43]České stránky projektu Slackware je možno najít na adrese http://www.slackware.cz/, oficiální stránky jsou na adrese http://www.slackware.com.

[44]Viz Häring. *Start systému: jakou roli hraje proces init? (1. část)*.

nepropustí, ale uchovává logy maximálně 5 dní). Ze zabezpečení pomocí IPS by ale byl ještě větší užitek, kdyby organizace měla zavedenu pozici bezpečnostního manažera, který by v rámci své náplně práce pravidelně vyhodnocoval logy podle principů NSM (Network Security Management).[45]

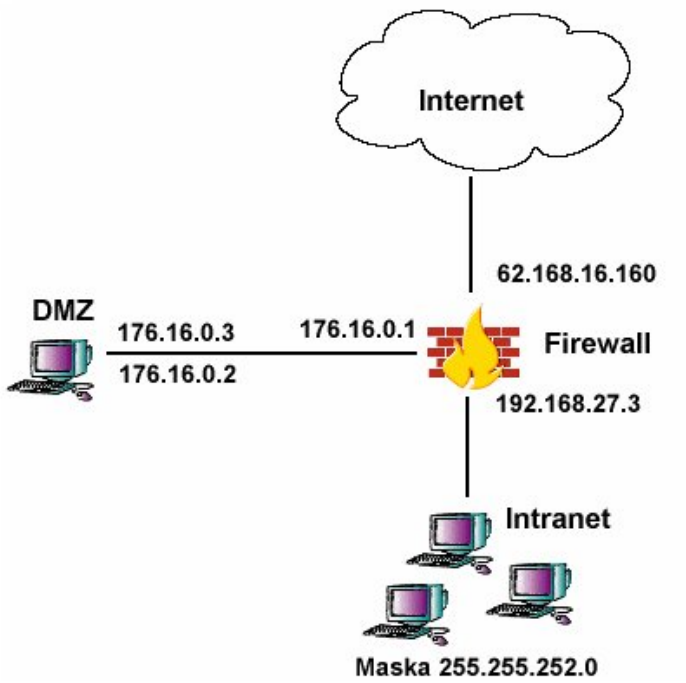
Na druhou stranu se ukázalo, že tento Cisco firewall nefunguje jako plnohodnotný router a není možno v něm např. namapovat na jednu síťovou kartu dvě IP adresy. Tento nedostatek může být natolik závažný, že bude třeba stávající firewall nahradit nějakým jiným.

5.2 Instalace serveru

Jak už jsme uvedli v předcházející části, linuxový firewall byl zprovozněn na distribuci Slackware. Kvůli bezpečnosti byly nainstalovány pouze nejnutnější balíčky bez grafického režimu. Navíc byl zprovozněn systém LIDS[46] (Linux Intrusion Detection System).

Tento systém neumožňuje v normálním stavu ani administrátorovi spouštět žádné programy, úplný přístup k systému je možno obdržet až po vypnutí LIDSů (buď na právě používané konzoli nebo v celém systému) pomocí dalšího hesla.[47]

5.3 Struktura sítě PLB



Obrázek 12: Struktura sítě PLB

[45]Viz Kurtz, *Hacking bez tajemství*, s. 2–7.
[46]Domovská stránka projektu je http://www.lids.org.
[47]Viz Hatch. *Linux. Hackerské útoky*, s. 71–72.

Na perimetru PLB byl použit jako firewall server se třemi síťovými kartami, jedna vedla do Internetu, jedna do Intranetu a jedna do DMZ (Demititarizovaná zóna). Struktura sítě je znázorněna na obrázku 12.

Na síťové kartě do DMZ byla nastavena IP adresa 176.16.0.1 s maskou sítě 255.255.255.0, vnitřní síť měla IP 192.168.27.3 s maskou 255.255.252.0. Je třeba upozornit na to, že adresa v DMZ neodpovídá rozsahu adres pro vnitřní sítě[48], ale v době nasazování firewallu nebyla vhodná situace k přečíslování DMZ. Toto přečíslování proběhlo až později společně s přečíslováním IP adres v intranetu při nasazování nového Cisco firewallu v souvislosti s tím, že se vyskytla obava, aby nový firewall neměl potíže s maskou sítě, která není ani třídy A, ani B, ani C.

Server v DMZ měl na jednu síťovou kartu asociovány dvě IP adresy. To bylo proto, že původně na tomto serveru běžely dvě instance internetového serveru Apache a každá z nich poslouchala na jiné IP adrese. Firewall pak měl za úkol nepropustit instanci Apache, která obstarávala Intranet, do vnějšího Internetu.[49] Později byl Intranet přesunut na server umístěný ve vnitřní síti, v konfiguračním souboru firewallu se tato skutečnost odrazila definováním dvou proměnných `$SERVER_IP` a `$SERVER_IP2`.

Dvě IP adresy asociované na jednu síťovou kartu využívaly také dvě instance DNS serveru bind, jedna sloužila jako veřejné DNS, druhá jako DNS pro Intranet. Stejně jako u webového servery bylo úkolem firewallu tyto dvě instance od sebe oddělit a nepropustit do Internetu informace podávající o vnitřní síti interní DNS. V současnosti je i toto vyřešeno. DNS pro intranet byl přesunut do vnitřní sítě a veřejné DNS je nyní hostované u poskytovatele domény.

Na serveru v DMZ byl také spuštěn proxy server Squid. Veškeré požadavky z vnitřní sítě na internetové stránky musely být předány proxy serveru, protože přímý přístup z Intranetu do Internetu byl implicitně zakázán, povoleno bylo pouze několik vyjímek pro administrátory (např. pro řešení problémů s nefungující proxy) a také bylo třeba zavést několik vyjímek kvůli tomu, že proxy server Squid neposkytoval plnohou funkcionalitu FTP protokolu.

5.4 Popis původního konfiguračního souboru serveru

Síťová rozhraní popisuje soubor *rc.inet1.conf*. Z něj vychází konfigurační soubor *firewall.conf*, který definuje rozhraní firewallu. Vlastní nastavení firewallu je v souboru *rc.firewall*, který si zde popíšeme.

Jedná se o shellový skript. V části 4.4.4 jsme popsali možnosti načítání konfigurace pomocí příkazu `iptables-restore`, ale vzhledem k tomu, že počet pravidel není až tak příliš velký, můžeme využít toho, že shellový skript se dá snadno prohlížet a je možno do něj zapisovat vysvětlující poznámky.

Na začátku skriptu (řádek 4) jsme připojili k souboru konfigurační soubor firewall.conf. V dalším kroku jsme vypsali hodnoty proměnných použitých v pravidlech (řádky 9–18). Řádek 21 vynuluje defaultní tabulku *table*, řádek

[48] Viz RFC 1918.

[49] Z bezpečnostního hlediska to nebylo ideální řešení, ale pozůstatek doby, kdy PLB měla jen jeden server.

22 vynuluje tabulku *nat*. Příkaz `iptables -X` smaže všechny řetězce kromě programově zabudovaných.

Řádky 27–29 definují politiku firewallu – DROP v řetězcích INPUT, OUTPUT i FORWARD. Následující pravidla definují výjimky z implicitní politiky, ve které je vše zakázáno.

Pro správnou funkci je třeba povolit loopback (viz řádky 32–33). Velmi důležité je pravidlo pro NAT (řádek 36). Následuje povolení přímého průchodu přes firewall pro administrátory (řádky 38 a 40).

V době konfigurace firewallu neběžel v DMZ ještě ntp server, proto bylo nutno otevřít pro novellovský server přístup ven na port 123. To je zachyceno na řádcích 43 a 44.

Na řádcích 46 až 59 jsou definovány různé výjimky kvůli konkrétním aplikacím: pro ERP, pro právní program Aspi, FTP. Běžný ftp provoz prochází přes proxy server Squid v DMZ, ale ten neposkytuje plnohodnotný ftp protokol. Kvůli některým konkrétním aplikacím bylo třeba otevřít přímý přístup do internetu na porty 21 a 22.

Při připojování k poštovnímu serveru postfix v DMZ protokolem imap docházelo k velkým časovým prodlevám. Řádek 59 tento problém vyřešil.

V řádcích 61–77 jsou uvedeny příkazy pro otevření přístupu na novellovské servery z vnějšího Internetu pro firmu vykonávající dohled nad těmito servery. Tuto část konfiguračního souboru by bylo možno zjednodušit použitím direktivy `-m multiport` a zapsáním několika portů do jednoho řádku.

Řádky 79 až 83 povolují přímé spojení mezi serverem s informačním systémem a firmou, která tento IS spravuje (pro port 3389).

Kromě poštovního serveru postfix byl ve vnitní síti v testovacím provozu také poštovní systém GroupWise. Jeho webové rozhraní bylo přístupné přes protokol 443 z celého internetu, povolují to řádky 86–88.

Administrátoři měli přístup do DMZ přes protokol ssh z vlastních IP adres v internetu (viz řádky 90 až 99).

Na firewall bylo možno se připojit ze serveru v DMZ a z PC administrátora ve vnitřní síti přes protokol ssh (viz řádky 106 a 108).

Webový server běžel v DMZ, přístup na něj z vnějšího internetu zajišťovaly příkazy na řádku 111 a 112.

V DMZ také běžely dvě instance DNS serveru, vnější DNS odpovídal na IP adrese `SERVER_IP2` 176.16.0.3, pro vnější internet jej zpřístupňovaly řádky 115–118.

Antispam SpamAssassin na poštovním serveru v DMZ měl nainstalován modul DCC, který se připojoval k různým serverům v Internetu a kontroloval podle hash součtů, zda přicházející e-maily nejsou na některém ze serverů označeny jako spamy. K povolení těchto akcí sloužily řádky 121 až 124.

Příjem pošty – smtp protokol – povolují řádky 127 a 128.

Řádky 131 až 133 definují stavový firewall. Povolují navazovat nová spojení z vnitřní sítě, nikoli však z vnějšího Internetu, řádky 139, 140 umožňují firewallu přijmout cokoliv z vnitřní sítě.

Problematický je následující řádek 150, který umožňuje navázat spojení z DMZ jak do vnitní sítě, tak do vnějšího Internetu:

```
# Keep state. (for DMZ)
iptables -A FORWARD -m state --state NEW -i ${DMZ_DEVICE} -j ACCEPT
```

Bezpečnějším řešením je, aby z DMZ byl neomezený přístup do Internetu, avšak do vnitřní sítě organizace by bylo možno navázat spojení pouze v odůvodněných případech.

Zbývající část konfiguračního souboru byla přejata z konfiguračního souboru floppyfw[50] a má řešit různé bezpečnostní incidenty, např. poskytovat ochranu proti scanu portů, útokům SYN flood, Ping of Death, smurf.

Řádek 184 umožňuje nastavit zpětné volání modemem s dynamickým rozhraním,[51] takže v případě naší konkrétní konfigurace je tento příkaz zbytečný, ale také nepůsobí žádnou škodu.

5.5 Nalezené problémy a návrh jejich řešení

Při zkoumání konfiguračního souboru firewallu jsme narazili na slabé místo, týkající se politiky přístupu do DMZ. Přístup do DMZ by měl být povolen z vnitřní sítě, ale nikoli naopak z DMZ do vnitřní sítě. Zde by se mělo povolovat navazování spojení pouze v odůvodněných případech. To znamená, že řádek 150 v souboru *rc.firewall* by měl mít tvar:

```
# Keep state. (for DMZ)
iptables -A FORWARD -m state --state NEW -i ${DMZ_DEVICE} \
-o ${OUTSIDE_DEVICE} -j ACCEPT
```

Dalším problémem by mohly být neúplné ingresní a egresní kontroly. V našem konkrétním případě není třeba se zabývat možností, že by přes firewall z intranetu do Internetu odcházely pakety s IP adresou vnitřní sítě, toto je dostatečně ošetřeno v řádku 36 konfiguračního souboru *rc.conf* (pomocí SNAT). Můžeme ale ošetřit, aby v žádném případě neodcházely přímo z firewallu pakety s jinou zdrojovou adresou, než je adresa firewallu. K tomu může sloužit následující příkaz:

```
/sbin/iptables -A OUTPUT -o ${OUTSIDE_DEVICE} -s !${OUTSIDE_IP} -j DROP
```

Je však dobré odfiltrovat adresy z vnějšího Internetu, které jsou z rozsahů vyhrazených pro vnitřní sítě. To by bylo možno udělat např. pomocí následujících příkazů:

```
for badnet in 127.0.0.1/32 10.0.0.0/8 172.16.0.0/12 192.168.0.0/16 \
224.0.0.0/4 240.0.0.0/5
do
/sbin/iptables -A INPUT -i ${OUTSIDE_NETWORK} -s $badnet -j DROP
/sbin/iptables -A FORWARD -i ${OUTSIDE_NETWORK} -s $badnet -j DROP
done
```

Konfigurační soubor by také bylo možno trochu zjednodušit použitím direktivy `-m multiport` a zapsáním několika portů do jednoho řádku, avšak

[50] Viz Domácí stránky projektu Floppyfw http://www.zelow.no/svn/floppyfw/files/firewall.ini.

[51] Viz Hubert. *Linux 2.4 Advanced Routing HOWTO.* Konkrétní vysvětlení parametrů je možno nalézt na http://www.filewatcher.com/p/linux-2.0.35.tar.bz2.5700498/linux/Documentation/networking/ip_dynaddr.txt.html.

tato možnost se neukázala být tak užitečná, jak jsme předpokládali. Vlastně se uplatnila pouze u povolování přístupu k novellovskému serveru (řádky 62–76), kde místo šesti řádků konfiguračního souboru vystačíme se dvěma. Takového zkrácení je vzhledem k celkovému rozsahu konfiguračního souboru zanedbatelné.

Upravený konfigurační soubor je uveden v příloze jako *rc.firewall-opraveny*. Pro větší přehlednost je navíc také v příloze připojen výstup příkazu `diff rc.firewall rc.firewall-opraveny`, který zachycuje rozdíly mezi původním a opraveným konfiguračním souborem.

5.6 Shrnutí

V této části jsme si popsali použití firewallu v podmínkách konkrétní organizace – Psychiatrické léčebny Bohnice – tak, jak fungoval v letech 2005–2008. Podrobně jsme probrali konfigurační soubor firewallu *rc.firewall* a přitom se ukázalo, že v nastavení bylo slabé místo, způsobené méně bezpečným nastavením politiky přístupu do DMZ. Také v konfiguračním souboru byly neúplné ingresní a egresní kontroly. Navrhli jsme upravený konfigurační soubor, který je uveden v příloze jako *rc.firewall-opraveny* a pro větší přehlednost je na také připojen výstup příkazu `diff rc.firewall rc.firewall-opraveny`, který zachycuje rozdíly mezi původním a opraveným konfiguračním souborem.

6 Závěr

Práce má tři hlavní části, v první části nazvané *TCP/IP, router a firewall* jsme nastínili teorii protokolů TCP/IP a vysvětlili pojmy router, firewall a stručně se dotkli některých otázek týkajících se bezpečnosti.

V druhé části nazvané *Router a firewall na Linuxu* jsme uvedli možnosti routování v Linuxu a více jsme se zabývali programem Netfilter a pak zvláště modulem iptables, který využívá programu Netfilter. Nastínili jsme také některé funkce a možnosti firewallu iptables.

Ve třetí části práce nazvané *Konfigurace routeru a firewallu v prostředí konkrétní organizace* jsme popsali situaci v Psychiatrické léčebně Bohnice, kde byl Linuxový firewall v provozu v letech 2005 až 2008. Údaje o struktuře sítě i IP adres je nyní možné zveřejnit, protože na konci roku 2008 došlo k restrukturalizaci sítě a změně IP adres ve vnitřní síti i DMZ.

Prozkoumali jsme konfigurační soubory firewallu ve světle teoretických poznatků, uvedených v první a druhé části práce, a bylo nalezeno několik problematických míst nebo částí, které by bylo možno vylepšit. Původní i opravený konfigurační soubor jsou uvedeny v příloze. Pro snadnější porovnání je v příloze také uveden výstup programu `diff`, který ukazuje rozdíly mezi oběma konfiguračními soubory.

Upravený konfigurační soubor by bylo však ještě třeba otestovat v produkčním prostředí.

7 Seznam literatury

ANDREASSON, Oskar. *Iptables Tutorial 1.2.2* [online] 2001-2006. [cit. 11.10.2009] Dostupné z: http://www.frozentux.net/documents/iptables-tutorial/

BOTOŠ, Csaba: *Vše o iptables: úvod.* [online] Vystaveno 10. 1. 2006. Dostupné z: http://www.root.cz/clanky/vse-o-iptables-uvod/

DOSTÁLEK, Libor – Kabelová, Alena. *Velký průvodce protokoly TCP/IP a systémem DNS.* 3. aktualizované a rozšířené vydání. Brno: Computer Press, 2005. 542 s. ISBN 80-722-6675-6

EASTEP, Tom. *Shorewall and Routing.* [online] Vystaveno 2005.[cit. 12.9.2009] Dostupné z: http://docs.huihoo.com/shorewall/2.0/Shorewall_and_Routing.html

ENDORF, Carl – MELLANDER, Jim – SCHULTZ, Eugene: *Hacking. Detekce a prevence počítačového útoku.* Přel. Ivo Blachovicz. 1. vyd. Praha: Grada, 2005. 356 s.
ISBN 80-247-1035-8

HABRMAN, Robert: *Síťové protokoly (I. část) – Model OSI* [online] [cit. 20.10.2009]. Dostupné z: http://www.owebu.cz/pc-site/vypis.php?clanek=1262

HÄRING, David. *Start systému: jakou roli hraje proces init? (1. část)* [online] LinuxZone. Vystaveno 17.04.2002. [cit. 16.10.2009]
Dostupné z: http://www.linuxzone.cz/index.phtml?ids=29&idc=195

HATCH, Brian – LEE, James – KURTZ, George. *Linux Hackerské útoky. Bezpečnost Linuxu – tajemství a řešení.* Přel. Jan Kuklínek, Lukáš Paulíček. 1. vyd. Praha:, SoftPress, 2001. 576 s. ISBN 80-86497-17-8

HUBERT, Bert – MAXWELL, Gregory - Mook, Remco van. *Linux 2.4 Advanced Routing HOWTO.* [online] Vystaveno 8.10.2000. [cit. 20.10.2009]
Dostupné z: http://www.linuxdocs.org/HOWTOs/Adv-Routing-HOWTO-12.html

HUNT, Craig. *Konfigurace a správa sítí TCP/IP.* Přel J. Veselský. 1. vydání. Brno: Computer Press, 1997. 458 s. ISBN 80-7226-024-3

KRASEK, Jáchym. *Peer to peer sítě od A do Z: Gnutella a Ares.* [online] Vystaveno 8. 7. 2008. [cit. 15.10.2009] Dostupné z:
http://www.lupa.cz/clanky/peer-to-peer-site-od-a-do-z-gnutella-a-ares/

KUČERA, Jan. *Historie Linuxu pěkně od začátku.* [online] [cit. 8.11.2009]
Dostupné z: http://www.fi.muni.cz/usr/jkucera/pv109/2003/xsumsky.htm

NetworkWorld research center. *LANs/Routers. Layer 4 switch.* [online] [cit. 15.1.2010] Dostupné z: http://www.networkworld.com/details/725.html?def

ODVÁRKA, Petr. *TCP handshake krok za krokem.* [online] Vystaveno: 3.12.2000. [cit. 15.9.2009] Dostupné z:
http://www.svetsiti.cz/view.asp?rubrika=Technologie&temaID=&clanekID=27

PERKINS, Charles – STREBE, Matthew. *Firewally a proxy-servery. Praktický průvodce.* Přel. L. Hendrychová, J. Milaštík. 1. vydání. Brno: Computer Press, 2003. 450 s. ISBN 80-7226-983-6

PETERKA, Jiří. *Bridge* [online] Archiv článků a přednášek Jiřího Peterky [cit. 17.11.2009]
Dostupné z: http://www.earchiv.cz/a97/a711k150.php3

PETERKA, Jiří. *Kdy použít switch?* [online] Archiv článků a přednášek Jiřího Peterky [cit. 17.11.2009]
Dostupné z: http://www.earchiv.cz/a93/a341c120.php3

PETERKA, Jiří. *Rodina protokolů TCP/IP, verze 2.5. Část 2: Architektura TCP/IP* Katedra softwarového inženýrství Matematicko-fyzikální fakulta Univerzity Karlovy [online] Dostupné z: http://www.earchiv.cz/i_downpredn.php3

PETERKA, Jiří. *Rodina protokolů TCP/IP, verze 2.5. Část 3: IP adresy.* Katedra softwarového inženýrství Matematicko-fyzikální fakulta Univerzity Karlovy [online] 2008 Dostupné z: http://www.earchiv.cz/i_downpredn.php3

PETŘÍČEK, Miroslav. *Stavíme firewall (1).* [online] Vystaveno 18. 12. 2001. Root. [cit. 18.10.2009] Dostupné z: http://www.root.cz/clanky/stavime-firewall-1/

PUŽMANOVÁ, Rita. *Jak se orientovat v RFC aneb Průvodce profesionála.* [online] Vystaveno 30.4.2002. [cit.23.9.2009] Dostupné z:
http://www.lupa.cz/clanky/jak-se-orientovat-v-rfc-aneb-pruvodce-profesionala/

RAYMOND, Eric S. *Umění programování v Unixu.* Přel. Bogdan Kiszka. 1. vyd. Brno: Computer Press, 2004. ISBN 80-251-0225-4

VESELÝ, Arnošt. *Operační systémy II.* 2. vyd., 1.dotisk. Praha: Česká zemědělská univerzita v Praze. Provozně ekonomická fakulta, 2008. 255 s. ISBN 978-80-213-1553-2

The German GPL Order – Translated. Přel. z němčiny do angličtiny Jens Maurer [online] Groklaw. Vystaveno 25.7.2004 [cit. 28.2.2010] Dostupné z: http://www.groklaw.net/article.php?story=20040725150736471

VONDRÁČEK, Jan. *Linux jako internetová gateway (1).* [online] Vystaveno 14. 1. 2004. [cit. 20.10.2009]
Dostupné z: http://www.root.cz/clanky/linux-jako-internetova-gateway-1/

Přílohy

Výpis souboru firewall.conf

```
# eth0 default device.
OUTSIDE_IP=62.168.16.162
OUTSIDE_DEVICE=eth0
OUTSIDE_NETMASK=255.255.255.248
OUTSIDE_NETWORK=62.168.16.160
OUTSIDE_BROADCAST=62.168.16.167

# Your inside network, this has 10.42.42.* set as default, this is
# addresses assigned for internal networks according to RFC 1918.

# eth1 is the default device for the internal network.
INSIDE_IP=192.168.27.1
INSIDE_DEVICE=eth1
INSIDE_NETWORK=192.168.24.0
INSIDE_NETMASK=255.255.252.0
INSIDE_BROADCAST=192.168.27.255

# DMZ network:
DMZ_DEVICE=eth2
DMZ_IP=176.16.0.1
DMZ_NETMASK=255.255.255.0
DMZ_NETWORK=176.16.0.0
DMZ_BROADCAST=176.16.0.255

# Misc
DEFAULT_GATEWAY=62.168.16.161
NAME_SERVER_IP1=176.16.0.3
NAME_SERVER_IP2=
HOSTNAME=jean
DOMAIN=plbohnice.cz
```

Výpis souboru rc.firewall

```
#!/bin/sh
# Firewall setup.
#
. /etc/firewall.conf
#
SERVER_IP=176.16.0.2
SERVER_IP2=176.16.0.3
#
echo "Starting firewall with the following config:"
echo
echo "                Inside               Outside               DMZ"
echo "   Network:  ${INSIDE_NETWORK}        ${OUTSIDE_NETWORK}        ${DMZ_NETWORK}"
echo "    Device:  ${INSIDE_DEVICE}               ${OUTSIDE_DEVICE}               ${DMZ_DEVICE}"
echo "IP Address:  ${INSIDE_IP}        ${OUTSIDE_IP}        ${DMZ_IP}"
echo "   Netmask:  ${INSIDE_NETMASK}        ${OUTSIDE_NETMASK}        ${DMZ_NETMASK}"
echo " Broadcast:  ${INSIDE_BROADCAST}        ${OUTSIDE_BROADCAST}        ${DMZ_BROADCAST}"
echo "   Gateway:  [None Set]               ${OUTSIDE_GATEWAY}        [None Set]"
echo

# Flushing the chains.
iptables -F
iptables -t nat -F
iptables -X
iptables -Z   # zero all counters

# Policy for chains DROP everything
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

#lo
iptables -A OUTPUT -m state --state NEW,ESTABLISHED,RELATED -d 127.0.0.1 -j ACCEPT
iptables -A INPUT -m state --state NEW,ESTABLISHED,RELATED -s 127.0.0.1 -j ACCEPT

# Masquerading for server
iptables -t nat -A POSTROUTING -o ${OUTSIDE_DEVICE} -j SNAT -s ${SERVER_IP} --to-source ${OUTSIDE_IP}
# Katka muze vsude
iptables -t nat -A POSTROUTING -o ${OUTSIDE_DEVICE} -j SNAT -s 192.168.27.20 --to-source ${OUTSIDE_IP}
#linux druheho administratora muze vsude
iptables -t nat -A POSTROUTING -o ${OUTSIDE_DEVICE} -j SNAT -s 192.168.25.88 --to-source ${OUTSIDE_IP}

#novell muze na ntpserver
```

```
iptables -t nat -A POSTROUTING -p udp -o ${OUTSIDE_DEVICE} -j SNAT -s 192.168.27.4 --destination-port 123 \
        --to-source ${OUTSIDE_IP}

#Vema v3demo.vema.cz potrebuje port 4147
#vedouci ERP
iptables -t nat -A POSTROUTING -p tcp -o ${OUTSIDE_DEVICE} -j SNAT -s 192.168.26.28 --destination-port 4147 \
        --to-source ${OUTSIDE_IP}

# antivir muze na aspi
iptables -t nat -A POSTROUTING -p tcp -o ${OUTSIDE_DEVICE} -j SNAT -s 192.168.27.199 \
                --to-source ${OUTSIDE_IP}:9000

#Primarka kvuli vyzkumu
iptables -t nat -A POSTROUTING -p tcp -o ${OUTSIDE_DEVICE} -j SNAT -s 192.168.25.51 --to-source ${OUTSIDE_IP}:21
iptables -t nat -A POSTROUTING -p tcp -o ${OUTSIDE_DEVICE} -j SNAT -s 192.168.25.51 --to-source ${OUTSIDE_IP}:22

#Win XP ten paket zahodi a server ceka na vytimeoutovani - nouzove reseni pro IDENT
iptables -A FORWARD -p tcp --destination-port 113 -j REJECT

# Novell
iptables -t nat -A POSTROUTING -o ${OUTSIDE_DEVICE} -j SNAT -s 192.168.27.4 --to-source ${OUTSIDE_IP}
iptables -A PREROUTING -t nat -p tcp -d ${OUTSIDE_IP} --dport 4242 -j DNAT --to 192.168.27.4:4242
iptables -A FORWARD -p tcp -d 192.168.27.4 --dport 4242 -o ${INSIDE_DEVICE} -j ACCEPT
iptables -A PREROUTING -t nat -p tcp -d ${OUTSIDE_IP} --dport 8009 -j DNAT --to 192.168.27.4:8009
iptables -A FORWARD -p tcp -d 192.168.27.4 --dport 8009 -o ${INSIDE_DEVICE} -j ACCEPT
iptables -A PREROUTING -t nat -p tcp -d ${OUTSIDE_IP} --dport 2200 -j DNAT --to 192.168.27.4:2200
iptables -A FORWARD -p tcp -d 192.168.27.4 --dport 2200 -o ${INSIDE_DEVICE} -j ACCEPT

iptables -A PREROUTING -t nat -p udp -d ${OUTSIDE_IP} --dport 4242 -j DNAT --to 192.168.27.4:4242
iptables -A FORWARD -p udp -d 192.168.27.4 --dport 4242 -o ${INSIDE_DEVICE} -j ACCEPT
iptables -A PREROUTING -t nat -p udp -d ${OUTSIDE_IP} --dport 8009 -j DNAT --to 192.168.27.4:8009
iptables -A FORWARD -p udp -d 192.168.27.4 --dport 8009 -o ${INSIDE_DEVICE} -j ACCEPT
iptables -A PREROUTING -t nat -p udp -d ${OUTSIDE_IP} --dport 2200 -j DNAT --to 192.168.27.4:2200
iptables -A FORWARD -p udp -d 192.168.27.4 --dport 2200 -o ${INSIDE_DEVICE} -j ACCEPT
#uvolnit Novelove porty zvenku
iptables -A FORWARD -m state --state NEW -d 192.168.27.4 -i ${OUTSIDE_DEVICE} -j ACCEPT

#Hippo server (pristup ven i dovnitr)
iptables -t nat -A POSTROUTING -o ${OUTSIDE_DEVICE} -j SNAT -s 192.168.27.109 --to-source ${OUTSIDE_IP}
iptables -A PREROUTING -t nat -p tcp -s 81.19.2.252  -d ${OUTSIDE_IP} --dport 3389 \
                -j DNAT --to 192.168.27.109:3389
iptables -A FORWARD -p tcp -d 192.168.27.109 --dport 3389 -o ${INSIDE_DEVICE} -j ACCEPT
iptables -A FORWARD -m state --state NEW -d 192.168.27.109 -s 81.19.2.252 -i ${OUTSIDE_DEVICE} -j ACCEPT

#groupwise pristup zvenku
iptables -t nat -A POSTROUTING -o ${OUTSIDE_DEVICE} -j SNAT -s 192.168.26.207 --to-source ${OUTSIDE_IP}
iptables -A PREROUTING -t nat -p tcp -d ${OUTSIDE_IP} --dport 443 -j DNAT --to 192.168.26.207:443
iptables -A FORWARD -p tcp -d 192.168.26.207 --dport 443 -o ${INSIDE_DEVICE} -j ACCEPT

#venkovni pristup ssh pro administratory na server do DMZ
#katerina
iptables -A PREROUTING -t nat -p tcp -s 195.113.5.3 -d ${OUTSIDE_IP} --dport 22 -j DNAT --to ${SERVER_IP}:22
iptables -A FORWARD -m state --state NEW -d ${SERVER_IP} -s 195.113.5.3  -i ${OUTSIDE_DEVICE} -j ACCEPT

#pristup zvenku pro 2.admina
iptables -A PREROUTING -t nat -p tcp -s 193.0.231.40 -d ${OUTSIDE_IP} --dport 22 -j DNAT --to ${SERVER_IP}:22
iptables -A FORWARD -m state --state NEW -d ${SERVER_IP} -s 193.0.231.40 -i ${OUTSIDE_DEVICE} -j ACCEPT

iptables -A FORWARD -p tcp -d ${SERVER_IP} --dport 22 -o ${DMZ_DEVICE} -j ACCEPT

#venkovni pristup pro dohled na switche - tuneluje tudy port 80 ze switchu
iptables -A PREROUTING -t nat -p tcp -s 147.32.48.21 -d ${OUTSIDE_IP} --dport 22 -j DNAT --to ${SERVER_IP}:22
iptables -A FORWARD -m state --state NEW -d ${SERVER_IP} -s 147.32.48.21 -i ${OUTSIDE_DEVICE} -j ACCEPT

#pristup na marvin ze sivt_katerina
iptables -A INPUT -m state --state NEW -d ${INSIDE_IP} -p tcp -s 192.168.26.2 --dport 22 -j ACCEPT
#pristup na marvin z jeana
iptables -A INPUT -m state --state NEW  -p tcp -s 192.168.27.2 --dport 22 -j ACCEPT

# Web:
iptables -A PREROUTING -t nat -p tcp -d ${OUTSIDE_IP} --dport 80 -j DNAT --to ${SERVER_IP2}:80
iptables -A FORWARD -p tcp -d ${SERVER_IP2} --dport 80 -o ${DMZ_DEVICE} -j ACCEPT

# DNS:
iptables -A PREROUTING -t nat -p udp -d ${OUTSIDE_IP} --dport 53 -j DNAT --to ${SERVER_IP2}:53
iptables -A FORWARD -p udp -d ${SERVER_IP2} --dport 53 -o ${DMZ_DEVICE} -j ACCEPT
iptables -A PREROUTING -t nat -p tcp -d ${OUTSIDE_IP} --dport 53 -j DNAT --to ${SERVER_IP2}:53
iptables -A FORWARD -p tcp -d ${SERVER_IP2} --dport 53 -o ${DMZ_DEVICE} -j ACCEPT

# pro antispam DCC:
```

```
iptables -A PREROUTING -t nat -p udp -d ${OUTSIDE_IP} --dport 53 -j DNAT --to ${SERVER_IP2}:6277
iptables -A FORWARD -p udp -d ${SERVER_IP2} --dport 6277 -o ${DMZ_DEVICE} -j ACCEPT
iptables -A PREROUTING -t nat -p tcp -d ${OUTSIDE_IP} --dport 53 -j DNAT --to ${SERVER_IP2}:6277
iptables -A FORWARD -p tcp -d ${SERVER_IP2} --dport 6277 -o ${DMZ_DEVICE} -j ACCEPT

# SMTP (Internal mail server):
iptables -A PREROUTING -t nat -p tcp -d ${OUTSIDE_IP} --dport 25 -j DNAT --to ${SERVER_IP}:25
iptables -A FORWARD -p tcp -d ${SERVER_IP} --dport 25 -o ${DMZ_DEVICE} -j ACCEPT

# Keep state.
iptables -A FORWARD -m state --state NEW -i ${INSIDE_DEVICE} -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -m state --state NEW,INVALID -i ${OUTSIDE_DEVICE} -j DROP

# This is mainly for PPPoE usage but it won't hurt anyway so we'll just
# keep it here.
iptables -A FORWARD -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --clamp-mss-to-pmtu

iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
# Ping and friends.
iptables -A OUTPUT -p icmp -j ACCEPT # to both sides.
iptables -A INPUT  -p icmp -j ACCEPT

# And also, DHCP, but we can basically accept anything from the inside.
iptables -A INPUT -i ${INSIDE_DEVICE} -j ACCEPT
iptables -A OUTPUT -o ${INSIDE_DEVICE} -j ACCEPT

# Keep state. (for DMZ)
iptables -A FORWARD -m state --state NEW -i ${DMZ_DEVICE} -j ACCEPT

# We don't like the NetBIOS and Samba leaking. (from DMZ)
iptables -t nat -A PREROUTING -p TCP -i ${DMZ_DEVICE} --dport 135:139 -j DROP
iptables -t nat -A PREROUTING -p UDP -i ${DMZ_DEVICE} --dport 137:139 -j DROP
iptables -t nat -A PREROUTING -p TCP -i ${DMZ_DEVICE} --dport 445 -j DROP
iptables -t nat -A PREROUTING -p UDP -i ${DMZ_DEVICE} --dport 445 -j DROP

# Accepting packets between Inside and DMZ
iptables -A FORWARD -s ${INSIDE_NETWORK}/${INSIDE_NETMASK} -d ${DMZ_NETWORK}/${DMZ_NETMASK} -j ACCEPT
iptables -A FORWARD -s ${DMZ_NETWORK}/${DMZ_NETMASK} -d ${INSIDE_NETWORK}/${INSIDE_NETMASK} -j ACCEPT

# And also, DHCP, but we can basically accept anything from the inside. (for DMZ)
iptables -A INPUT -i ${DMZ_DEVICE} -j ACCEPT
iptables -A OUTPUT -o ${DMZ_DEVICE} -j ACCEPT

# And, some attempt to get interactive sesions a bit more interactive under load:
iptables -A PREROUTING -t mangle -p tcp --sport ssh  -j TOS --set-tos Minimize-Delay
iptables -A PREROUTING -t mangle -p tcp --sport ftp -j TOS --set-tos Minimize-Delay
iptables -A PREROUTING -t mangle -p tcp --sport ftp-data -j TOS --set-tos Maximize-Throughput

#ochrana proti scanu portu
iptables -A FORWARD -p tcp --tcp-flags SYN,ACK,FIN,RST RST -m limit --limit 1/s -j ACCEPT
#ochrana proti syn-flood
iptables -A FORWARD -p tcp --syn -m limit --limit 1/s -j ACCEPT
#ping of death
iptables -A FORWARD -p icmp --icmp-type echo-request --limit 1/s -j ACCEPT
#vsechny ostatni zahodit
iptables -A INPUT -p tcp --syn -j DROP

# Finally, list what we have
iptables -L -n

# This enables dynamic IP address following
echo 7 > /proc/sys/net/ipv4/ip_dynaddr

# trying to stop some smurf attacks.
echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
```

Výpis souboru rc.firewall-opraveny

```
#!/bin/sh
# Firewall setup.
#
. /etc/firewall.conf
#
SERVER_IP=176.16.0.2
SERVER_IP2=176.16.0.3
#
echo "Starting firewall with the following config:"
echo
```

```
11  echo "               Inside               Outside               DMZ"
12  echo "   Network:  ${INSIDE_NETWORK}        ${OUTSIDE_NETWORK}        ${DMZ_NETWORK}"
13  echo "    Device:  ${INSIDE_DEVICE}               ${OUTSIDE_DEVICE}               ${DMZ_DEVICE}"
14  echo "IP Address:  ${INSIDE_IP}        ${OUTSIDE_IP}        ${DMZ_IP}"
15  echo "   Netmask:  ${INSIDE_NETMASK}        ${OUTSIDE_NETMASK}        ${DMZ_NETMASK}"
16  echo " Broadcast:  ${INSIDE_BROADCAST}        ${OUTSIDE_BROADCAST}        ${DMZ_BROADCAST}"
17  echo "   Gateway:  [None Set]               ${OUTSIDE_GATEWAY}        [None Set]"
18  echo
19
20  # Flushing the chains.
21  iptables -F
22  iptables -t nat -F
23  iptables -X
24  iptables -Z   # zero all counters
25
26  # Policy for chains DROP everything
27  iptables -P INPUT DROP
28  iptables -P OUTPUT DROP
29  iptables -P FORWARD DROP
30
31  #lo
32  iptables -A OUTPUT -m state --state NEW,ESTABLISHED,RELATED -d 127.0.0.1 -j ACCEPT
33  iptables -A INPUT -m state --state NEW,ESTABLISHED,RELATED -s 127.0.0.1 -j ACCEPT
34
35  # Block clearly-spoofed packets
36  iptables -A OUTPUT -o ${OUTSIDE_DEVICE} -s !${OUTSIDE_IP} -j DROP
37
38  for badnet in      127.0.0.1/32 10.0.0.0/8 172.16.0.0/12 192.168.0.0/16 224.0.0.0/4 240.0.0.0/5
39  do
40      iptables -A INPUT  -i  ${OUTSIDE_DEVICE} -s $badnet -j DROP
41      iptables -A FORWARD -i  ${OUTSIDE_DEVICE} -s $badnet -j DROP
42  done
43
44  # Masquerading for server
45  iptables -t nat -A POSTROUTING -o ${OUTSIDE_DEVICE} -j SNAT -s ${SERVER_IP} --to-source ${OUTSIDE_IP}
46  # Katka muze vsude
47  iptables -t nat -A POSTROUTING -o ${OUTSIDE_DEVICE} -j SNAT -s 192.168.27.20 --to-source ${OUTSIDE_IP}
48  #linux druheho administratora muze vsude
49  iptables -t nat -A POSTROUTING -o ${OUTSIDE_DEVICE} -j SNAT -s 192.168.25.88 --to-source ${OUTSIDE_IP}
50
51  #novell muze na ntpserver
52  iptables -t nat -A POSTROUTING -p udp -o ${OUTSIDE_DEVICE} -j SNAT -s 192.168.27.4 --destination-port 123 \
53          --to-source ${OUTSIDE_IP}
54
55  #Vema v3demo.vema.cz potrebuje port 4147
56  #vedouci ERP
57  iptables -t nat -A POSTROUTING -p tcp -o ${OUTSIDE_DEVICE} -j SNAT -s 192.168.26.28 --destination-port 4147 \
58          --to-source ${OUTSIDE_IP}
59
60  # antivir muze na aspi
61  iptables -t nat -A POSTROUTING -p tcp -o ${OUTSIDE_DEVICE} -j SNAT -s 192.168.27.199 \
62                  --to-source ${OUTSIDE_IP}:9000
63
64  #Primarka kvuli vyzkumu
65  iptables -t nat -A POSTROUTING -p tcp -o ${OUTSIDE_DEVICE} -j SNAT -s 192.168.25.51 --to-source ${OUTSIDE_IP}:21
66  iptables -t nat -A POSTROUTING -p tcp -o ${OUTSIDE_DEVICE} -j SNAT -s 192.168.25.51 --to-source ${OUTSIDE_IP}:22
67
68  #Win XP ten paket zahodi a server ceka na vytimeoutovani - nouzove reseni pro IDENT
69  iptables -A FORWARD -p tcp --destination-port 113 -j REJECT
70
71  # Novell
72  iptables -t nat -A POSTROUTING -o ${OUTSIDE_DEVICE} -j SNAT -s 192.168.27.4 --to-source ${OUTSIDE_IP}
73  iptables -A PREROUTING -t nat -p tcp -d ${OUTSIDE_IP} --dport 4242 -j DNAT --to 192.168.27.4:4242
74  iptables -A PREROUTING -t nat -p tcp -d ${OUTSIDE_IP} --dport 8009 -j DNAT --to 192.168.27.4:8009
75  iptables -A PREROUTING -t nat -p tcp -d ${OUTSIDE_IP} --dport 2200 -j DNAT --to 192.168.27.4:2200
76
77  iptables -A PREROUTING -t nat -p udp -d ${OUTSIDE_IP} --dport 4242 -j DNAT --to 192.168.27.4:4242
78  iptables -A PREROUTING -t nat -p udp -d ${OUTSIDE_IP} --dport 8009 -j DNAT --to 192.168.27.4:8009
79  iptables -A PREROUTING -t nat -p udp -d ${OUTSIDE_IP} --dport 2200 -j DNAT --to 192.168.27.4:2200
80  iptables -A FORWARD -p udp -d 192.168.27.4 -m multiport --destination-ports 2200,4242,8009 \
81                  -o ${INSIDE_DEVICE} -j ACCEPT
82  iptables -A FORWARD -p tcp -d 192.168.27.4  -m multiport --destination-ports 2200,4242,8009 \
83                  -o ${INSIDE_DEVICE} -j ACCEPT
84
85  #uvolnit Novelove porty zvenku
86  iptables -A FORWARD -m state --state NEW -d 192.168.27.4 -i ${OUTSIDE_DEVICE} -j ACCEPT
87
88  #Hippo server (pristup ven i dovnitr)
89  iptables -t nat -A POSTROUTING -o ${OUTSIDE_DEVICE} -j SNAT -s 192.168.27.109 --to-source ${OUTSIDE_IP}
90  iptables -A PREROUTING -t nat -p tcp -s 81.19.2.252  -d ${OUTSIDE_IP} --dport 3389 \
```

```
                -j DNAT --to 192.168.27.109:3389
iptables -A FORWARD -p tcp -d 192.168.27.109 --dport 3389 -o ${INSIDE_DEVICE} -j ACCEPT
iptables -A FORWARD -m state --state NEW -d 192.168.27.109 -s 81.19.2.252 -i ${OUTSIDE_DEVICE} -j ACCEPT

#groupwise pristup zvenku
iptables -t nat -A POSTROUTING -o ${OUTSIDE_DEVICE} -j SNAT -s 192.168.26.207 --to-source ${OUTSIDE_IP}
iptables -A PREROUTING -t nat -p tcp -d ${OUTSIDE_IP} --dport 443 -j DNAT --to 192.168.26.207:443
iptables -A FORWARD -p tcp -d 192.168.26.207 --dport 443 -o ${INSIDE_DEVICE} -j ACCEPT

#venkovni pristup ssh pro administratory na server do DMZ
#katerina
iptables -A PREROUTING -t nat -p tcp -s 195.113.5.3 -d ${OUTSIDE_IP} --dport 22 -j DNAT --to ${SERVER_IP}:22
iptables -A FORWARD -m state --state NEW -d ${SERVER_IP} -s 195.113.5.3  -i ${OUTSIDE_DEVICE} -j ACCEPT

#pristup zvenku pro 2.admina
iptables -A PREROUTING -t nat -p tcp -s 193.0.231.40 -d ${OUTSIDE_IP} --dport 22 -j DNAT --to ${SERVER_IP}:22
iptables -A FORWARD -m state --state NEW -d ${SERVER_IP} -s 193.0.231.40 -i ${OUTSIDE_DEVICE} -j ACCEPT

iptables -A FORWARD -p tcp -d ${SERVER_IP} --dport 22 -o ${DMZ_DEVICE} -j ACCEPT

#venkovni pristup pro dohled na switche - tuneluje tudy port 80 ze switchu
iptables -A PREROUTING -t nat -p tcp -s 147.32.48.21 -d ${OUTSIDE_IP} --dport 22 -j DNAT --to ${SERVER_IP}:22
iptables -A FORWARD -m state --state NEW -d ${SERVER_IP} -s 147.32.48.21 -i ${OUTSIDE_DEVICE} -j ACCEPT

#pristup na marvin ze sivt_katerina
iptables -A INPUT -m state --state NEW -d ${INSIDE_IP} -p tcp -s 192.168.26.2 --dport 22 -j ACCEPT
#pristup na marvin z jeana
iptables -A INPUT -m state --state NEW  -p tcp -s 192.168.27.2 --dport 22 -j ACCEPT

# Web:
iptables -A PREROUTING -t nat -p tcp -d ${OUTSIDE_IP} --dport 80 -j DNAT --to ${SERVER_IP2}:80
iptables -A FORWARD -p tcp -d ${SERVER_IP2} --dport 80 -o ${DMZ_DEVICE} -j ACCEPT

# DNS:
iptables -A PREROUTING -t nat -p udp -d ${OUTSIDE_IP} --dport 53 -j DNAT --to ${SERVER_IP2}:53
iptables -A FORWARD -p udp -d ${SERVER_IP2} --dport 53 -o ${DMZ_DEVICE} -j ACCEPT
iptables -A PREROUTING -t nat -p tcp -d ${OUTSIDE_IP} --dport 53 -j DNAT --to ${SERVER_IP2}:53
iptables -A FORWARD -p tcp -d ${SERVER_IP2} --dport 53 -o ${DMZ_DEVICE} -j ACCEPT

# pro antispam DCC:
iptables -A PREROUTING -t nat -p udp -d ${OUTSIDE_IP} --dport 53 -j DNAT --to ${SERVER_IP2}:6277
iptables -A FORWARD -p udp -d ${SERVER_IP2} --dport 6277 -o ${DMZ_DEVICE} -j ACCEPT
iptables -A PREROUTING -t nat -p tcp -d ${OUTSIDE_IP} --dport 53 -j DNAT --to ${SERVER_IP2}:6277
iptables -A FORWARD -p tcp -d ${SERVER_IP2} --dport 6277 -o ${DMZ_DEVICE} -j ACCEPT

# SMTP (Internal mail server):
iptables -A PREROUTING -t nat -p tcp -d ${OUTSIDE_IP} --dport 25 -j DNAT --to ${SERVER_IP}:25
iptables -A FORWARD -p tcp -d ${SERVER_IP} --dport 25 -o ${DMZ_DEVICE} -j ACCEPT

# Keep state.
iptables -A FORWARD -m state --state NEW -i ${INSIDE_DEVICE} -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -m state --state NEW,INVALID -i ${OUTSIDE_DEVICE} -j DROP

# This is mainly for PPPoE usage but it won't hurt anyway so we'll just
# keep it here.
iptables -A FORWARD -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --clamp-mss-to-pmtu

iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
# Ping and friends.
iptables -A OUTPUT -p icmp -j ACCEPT # to both sides.
iptables -A INPUT  -p icmp -j ACCEPT

# And also, DHCP, but we can basically accept anything from the inside.
iptables -A INPUT -i ${INSIDE_DEVICE} -j ACCEPT
iptables -A OUTPUT -o ${INSIDE_DEVICE} -j ACCEPT

# Keep state. (for DMZ) - povolen pouze volný přístup z DMZ ven
iptables -A FORWARD -m state --state NEW -i ${DMZ_DEVICE} -o ${OUTSIDE_DEVICE} -j ACCEPT

# We don't like the NetBIOS and Samba leaking. (from DMZ)
iptables -t nat -A PREROUTING -p TCP -i ${DMZ_DEVICE} --dport 135:139 -j DROP
iptables -t nat -A PREROUTING -p UDP -i ${DMZ_DEVICE} --dport 137:139 -j DROP
iptables -t nat -A PREROUTING -p TCP -i ${DMZ_DEVICE} --dport 445 -j DROP
iptables -t nat -A PREROUTING -p UDP -i ${DMZ_DEVICE} --dport 445 -j DROP

# Accepting packets between Inside and DMZ
iptables -A FORWARD -s ${INSIDE_NETWORK}/${INSIDE_NETMASK} -d ${DMZ_NETWORK}/${DMZ_NETMASK} -j ACCEPT
iptables -A FORWARD -s ${DMZ_NETWORK}/${DMZ_NETMASK} -d ${INSIDE_NETWORK}/${INSIDE_NETMASK} -j ACCEPT
```

```
171
172 # And also, DHCP, but we can basically accept anything from the inside. (for DMZ)
173 iptables -A INPUT -i ${DMZ_DEVICE} -j ACCEPT
174 iptables -A OUTPUT -o ${DMZ_DEVICE} -j ACCEPT
175
176 # And, some attempt to get interactive sesions a bit more interactive under load:
177 iptables -A PREROUTING -t mangle -p tcp --sport ssh  -j TOS --set-tos Minimize-Delay
178 iptables -A PREROUTING -t mangle -p tcp --sport ftp -j TOS --set-tos Minimize-Delay
179 iptables -A PREROUTING -t mangle -p tcp --sport ftp-data -j TOS --set-tos Maximize-Throughput
180
181 #ochrana proti scanu portu
182 iptables -A FORWARD -p tcp --tcp-flags SYN,ACK,FIN,RST RST -m limit --limit 1/s -j ACCEPT
183 #ochrana proti syn-flood
184 iptables -A FORWARD -p tcp --syn -m limit --limit 1/s -j ACCEPT
185 #ping of death
186 iptables -A FORWARD -p icmp --icmp-type echo-request --limit 1/s -j ACCEPT
187 #vsechny ostatni zahodit
188 iptables -A INPUT -p tcp --syn -j DROP
189
190 # Finally, list what we have
191 iptables -L -n
192
193 # This enables dynamic IP address following
194 echo 7 > /proc/sys/net/ipv4/ip_dynaddr
195
196 # trying to stop some smurf attacks.
197 echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
```

Výstup příkazu `diff rc.firewall rc.firewall-opraveny`

```
35a36,44
> # Block clearly-spoofed packets
> iptables -A OUTPUT -o ${OUTSIDE_DEVICE} -s !${OUTSIDE_IP} -j DROP
>
> for badnet in      127.0.0.1/32 10.0.0.0/8 172.16.0.0/12 192.168.0.0/16 224.0.0.0/4 240.0.0.0/5
> do
>     iptables -A INPUT  -i  ${OUTSIDE_DEVICE} -s $badnet -j DROP
>     iptables -A FORWARD -i  ${OUTSIDE_DEVICE} -s $badnet -j DROP
> done
>
65d73
< iptables -A FORWARD -p tcp -d 192.168.27.4 --dport 4242 -o ${INSIDE_DEVICE} -j ACCEPT
67d74
< iptables -A FORWARD -p tcp -d 192.168.27.4 --dport 8009 -o ${INSIDE_DEVICE} -j ACCEPT
69d75
< iptables -A FORWARD -p tcp -d 192.168.27.4 --dport 2200 -o ${INSIDE_DEVICE} -j ACCEPT
72d77
< iptables -A FORWARD -p udp -d 192.168.27.4 --dport 4242 -o ${INSIDE_DEVICE} -j ACCEPT
74d78
< iptables -A FORWARD -p udp -d 192.168.27.4 --dport 8009 -o ${INSIDE_DEVICE} -j ACCEPT
76c80,84
< iptables -A FORWARD -p udp -d 192.168.27.4 --dport 2200 -o ${INSIDE_DEVICE} -j ACCEPT
---
> iptables -A FORWARD -p udp -d 192.168.27.4 -m multiport --destination-ports 2200,4242,8009 \
>  -o ${INSIDE_DEVICE} -j ACCEPT
> iptables -A FORWARD -p tcp -d 192.168.27.4  -m multiport --destination-ports 2200,4242,8009 \
>  -o ${INSIDE_DEVICE} -j ACCEPT
>
150,151c158,159
< # Keep state. (for DMZ)
< iptables -A FORWARD -m state --state NEW -i ${DMZ_DEVICE} -j ACCEPT
---
> # Keep state. (for DMZ) - povolen pouze volný přístup z DMZ ven
> iptables -A FORWARD -m state --state NEW -i ${DMZ_DEVICE} -o ${OUTSIDE_DEVICE} -j ACCEPT
```

Rejstřík

Seznam tabulek

Seznam obrázků