Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra Informatiky a výpočetní techniky

Filtrování paketů

číslo úlohy 21

Semestrální práce z předmětu
KIV/PD
(Přenos dat)

Pavel Tuček, A00227
tuca@students.zcu.cz

TUTORIAL
http://home.zcu.cz/projects/pd/index.html

Obsah

1. Zadání

Filtrování paketů - rozdíly mezi stavovým a paketovým fitrem, možnosti použití filtračních pravidel pro modifikaci IP paketů (nastavení TOS apod.), NAT, DNAT, SNAT, PAT, source-routing apod.
S přihlédnutím k Linuxovému jádru 2.4.x/2.6.x a *BSD.

2. Úvod

Tento článek bude pojednávat o důležitém tématu, který se týká Unixových systémů, a tím je *filtrování paketů.* Ano, pochopili jste to správně, jedná se o bezpečnost Vašeho počítače (nebo spíše o bezpečnost firewallu) a o řízení dat, která tímto firewallem projdou.
Řekneme si základní pojmy, které s tímto tématem souvisí a zkusíme si samotnou konfiguraci pod Linuxem a v BSD systému. Ke konfiguraci se využívá mnoho nástrojů z nichž mi přišel nejrozumější a nejlépe konfigurovatelný, také asi nejvíce používaný, IPTABLES, o kterém bude řeč níže v tomto článku. Zmíním samozřejmě i jiné nástroje, které se používají v jiných systémech než je Linux.

Po přečtění tohoto článku by měl být "zručný" člověk, který se trochu orientuje pod Linuxem, schopný nastavit svůj firewall tak, aby byl bezpečný pro okolní síť a zároveň bezpečný i zevnitř.

3. Co je to firewall a k čemu slouží?

Všichni co se pohybují v oblasti informačních technologií už tento termín jistě několikrát slyšeli a nezůstali jenom u naslouchání, přešli také k činům, vrhli se hned do konfigurace firemního firewallu nebo minimálně si nakonfigurovali domácí síť na dialupu. Přece jenom bych rád osvětlil-pro ty neznalé-co to vůbec pojem firewall znamená.

Firewally obecně jsou systémy sídlící na hranici mezi několika sítěmi, které zpracovávají provoz jdoucí přes tuto hranici (z obou směrů). Firewally umožňují filtraci a monitorování provozu. Pokročilejší implementace do síťového provozu i aktivně zasahují. Nejčastější nasazení firewallu bývá mezi LAN a Internetem, kdy chceme "hodnou" lokální síť chránit před "zlým" Internetem.

Pokud si chcete vytvořit firewall i pod systémem FreeBSD, tak se podívejte na tento seit, kde je pěkně vysvětleno, jak si postavit firewall. Používá se k tomu nástroj NetBoz. Celý systém je postavený na tom, že se bootuje z distibučního CD a konfigurace se ukládá na disketu, což je dnes celkem používaný způsob. Najdete to na http://www.linuxzone.cz/index.phtml?ids=1&idc=334

3.1 Druhy firewallů

Rozlišujeme tedy dva základní typy firewallů:

- *proxy, aplikační brány*
  Aplikační proxy pracují na úrovni protokolů jednotlivých služeb. Velice pěkný článek, kde je popsáno co to jsou vůbec proxy a aplikační brány, najdete v elektronickém archívu Jiřího Peterky.
- *paketové filtry* (shrnu jen krátce, jelikož se budu zabývat podrovněji v dalším bodě)
  Paketové filtry sledují pohyb dat po síti po jednotlivých paketech. Podle daných pravidel posuzují jednotlivé pakety a rozhodují o jejich dalším osudu. Minimálně však určují zda se paket propustí nebo zahodí. Paketové filtry nerozumějí vyšším vrstvám než sí'tové a nedokáží se podle nich rozhodovat. To co mají k dispozici jsou atributy paketů jako zdrojová/cílová adresa, rozhraní, ze kterého paket přišel, zdrojový/cílový port, etc. Samotný paketový filtr je rychlý, nenáročný na systémové zdroje. Celý proces se odehrává v jádře.

Poznámka: V dalším textu budu považovat termíny paketový filtr a firewall za ekvivalentní.

4. Paketový filtr

Paketový filtr funguje tak, že zkoumá obsah hlavičky každého IP datagramu. Samotná hlavička obsahuje zejména IP adresu původce i adresáta, zdrojový a cílový port specifikující program, kterému je datagram určen, a další informace popisující komunikaci, ke které datagram náleží. Paketový firewall je vlastně jakýsi filtr, který na základě těchto informací rozhoduje o tom, které pakety mohou být připuštěny až k programům, nebo které naopak smějí opustit počítač.

4.1 Paketový filtr v Linuxu

V linuxovém jádře je vestavěna poměrně silná a flexibilní podpora pro filtrování paketů (speciálně IP datagramů, i když je možno filtrovat i jiné protokoly, např. IPX) zpracovávaných síťovým subsystémem jádra. Filtrování je možno provádět ve třech různých fázích: bezprostředně po přijetí datagramu ze sítě, během jeho přesměrování do jiné sítě a před vysláním datagramu do sítě. Pro aktivaci těchto funkcí je třeba zapnout volby Network firewalls a IP firewalling. Poznamenejme, že není nutné, aby byl uzel nakonfigurován jako router (IP forwarding/gatewaying), i když je to obvyklé.

Na začátku se snaží jádro rozhodnout, zda-li je příchozí paket určen pro tento počítač, nebo jestli je potřeba jej routovat jinam.

- Je-li adresátem on sám, předá paket k dalšímu zpracování do vstupního (INPUT) řetězce. Pokud vyhoví tamějším filtrovacím pravidlům, dostane jej ke zpracování některý z lokálních programů poslouchající na cílovém portu.
- Pokud je datagram určen někomu jinému a počítači je zkonfigurován jako router, tedy pokud je povoleno routování paketů proměnnou /proc/sys/net/ipv4/ip_forward == 1, paket bude postoupen do řetezce FORWARD a počítač se jej pak pokusí podle svých možností doručit příjemci. Pokud je směrování paketů zakázáno (implicitní stav), bude paket zahozen.
- Poslední možností je, že datagram vytvořil některý z lokálních programů. Potom je nutné, aby paketový filtr opustil přes řetezec OUTPUT.

Kromě zmíněných řetezců INPUT, OUTPUT a FORWARD existují ještě další dva, a to PREROUTING a POSTROUTING, které se ovšem používají především k jiným účelům než k filtrování paketů.

4.2 Paketový filtr v BSD systémech

V současné době existují na Internetu dva rozdílné typy běžně používaných firewallů. První se nazývá *packet filtering router*, který běží jako součást jádra operačního systému na stroji s více síťovými rozhraními, a který na základě nastavených pravidel povoluje nebo zakazuje průchod paketů mezi těmito rozhraními. Tako pravidla bývají označována jako *chain-lists*, řetězce, kterými filtrující program prochází a pokud najde shodu s filtrovaným paketem, aplikuje pravidlo a skončí.
Druhým používaným typem je *proxy-server*, který pomocí dalších služeb ověřuje uživatele a jejich pakety případně dále směřuje do sítě. Tyto dva typy mohou být kombinovány dohromady, kde například filtrovací router povoluje pouze pakety z proxy-serveru, který předtím ověřil oprávněnost přístupu uživatelů. Takový proxy-server se pak nazývá *bastion host.*

Oproti Linuxu, který využívá především nástroje iptables jsou systémy BSD vybaveny o trochu jiným programem, který umožňuje filtrování paketů i pod těmito systémy. Jedná se především o nástroj ipfilter (ipf), který může být už v jádru a nebo zaveden jako modul. Spolu s ipf se může použít i nástroj ipfirewall (ipfw), který může, stejně jako ipfilter, být součástí jádra, a dále navíc tento nástroj přidává možnost tzv. "tvarování" provozu (traffic shaper), což je systém vyvažování zátěže. K dispozici jsou i některé komerční produkty jako např. Tripwire, který ochranu rozšiřuje na úroveň souborového systému a umí hlídat změny souborů. Pro iptables/netfilter, ipfilter (a jeho verzi pro OpenBSD) je určen graficky orientovaný konfigurační nástroj Firewall Builder, který uživateli v GTK grafice dovoluje jednoduše nastavovat pravidla pro uvedené programy.

4.2.1 IPFirewall ve FreeBSD

IPFW je systém, který je dodávaný s FreeBSD a je volitelně součástí jádra. Ovládá se pomocí uživatelského příkazu ipfw. IPFW má dvě části První část je filtrovací mechanismus a druhá část obsahuje monitorovací systém, který umožňuje získávat přehled o provozních podmínkách routeru. Dále ve spojení s částí jádra dummynet zprostředkuje vyvažování zátěže a je možné jej podobným způsobem použít i pro stanice nesloužící jako routery. Mezi jeho významné vlastnosti lze zařadit podporu pro jiné úrovně síťové komunikace než TCP/IP, schopnost přesměrovávat požadavky na jiné porty (divert), podpora “stateful” módu, kde filtrovací pravidla vznikají dynamicky za běhu systému. Je sice možné nastavit ho na "otevřený", ale nedoporučuje se to.

V současné době je k dispozici rozšířená verze ipfw2, která oproti své starší verzi prináší řadu rozšíření. Oproti netfilteru zde není žádná kostra, ale funkcionalita je v zásadě stejná. Narozdíl od netfilteru je implicitně po instalaci velice restriktivně nastaven, s čímž je potřeba počítat třeba u instalací na dálku. Pěkný návod jak na to najdete na stránce http://www.freebsd.cz/~michal/doc/ipfw.html.

5. Stavový firewall (Stateful inspection firewall)

Většinou nadstavba paketového filtru. Hlídá všechny platné tcp spojení a jejich porty, které ukládá do tabulky. V případě, že dorazí paket, který neodpovídá žádnému platnému spojení uplatní se na něj činnost definovaná politikou (zahození, předání ...).

Jádra řady 2.4 přicházejí tedy s koncepčně novým přístupem, kdy při filtrování berou v úvahu nejen informace obsažené v záhlaví zkoumaného datagramu, ale dokáží na něj nahlížet komplexně, v kontextu spojení, do kterého patří. Stavový firewall rozezná paket, který otevírá nové spojení, od paketů, které tuto komunikaci realizují, a díky tomu můžeme precizněji filtrovat datové toky.

Výhody stavového filtru

Stavový firewall přináší značné zjednodušení filtrovacích pravidel oproti dobám, kdy jsme pomocí ipchains museli brát v úvahu různé kombinace adres, vysokých portů a SYN flagů. Nyní tedy můžeme, díky klasifikaci paketů, imlicitně povolit, aby firewallem protékaly pakety patřící k již navázaným spojením (ESTABLISHED), a omezení stanovujeme na úrovni navázání spojení.

A jaké jsou nevýhody?

Jedinou větší nevýhodou jsou vyšší nároky na hardware firewallu. V paměti se musí totiž udržovat stavové informace o všech spojeních.

5.1 Kategorie

Každý zkoumaný datagram (a to nejen TCP segment, ale i UDP paket) je pak zařazen do některé z těchto kategorií:

- NEW - datagram otevírá novou komunikaci
- ESTABLISHED, RELATED - datagram je součástí již navázaného spojení nebo s ním nějakým způsobem souvisí.
- INVALID - datagram není součástí žádného spojení nebo se jej nepodařilo identifikovat.

Na základě stavové informace můžeme tedy pakety třídit. Můžeme například stanovit, že spojení mohou být navazována pouze směrem z vnitřní sítě ven a ne naopak, přičemž pakety již otevřených spojení mohou putovat oběma směry.

```
iptables -P FORWARD DROP
iptables -A FORWARD -i eth0 -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -i eth1 -j ACCEPT
```

6. QoS (Quality of Service)

Každý provider a každý uživatel by měl nabízet (požadovat) služby co nejlepší kvality (= Quality of Service).
V poslední době dochází k rozvoji služeb, jejichž úspěšnost z pohledu uživatele významně závisí na časových charakteristikách komunikace přes počítačovou síť. Jde například o služby Internet telefonie, videokonference a další interaktivní služby. Uživatel požaduje, aby mu byla poskytnuta určitá definovaná kvalita služby (QoS). Například videosignál určitého rozlišení o určitém počtu obrázků za sekundu, zvuk určité šířky pásma nebo přenos souboru dat v určitém čase. Tyto požadavky na QoS aplikací lze splnit, když se odpovídajícím způsobem mapují na QoS počítačové sítě.
Nejvýznamnější parametry, které definují QoS počítačové sítě jsou následující:

- **Ztrátovost paketů** - kolik procent paketů nedorazí od odesílatele k adresátovi,
- **Průchodnost** - objem dat v bajtech přenesený za jednotku času,
- **Zpoždění** - doba potřebná k přenosu paketu od odesílatele k adresátovi,
- **Změna zpoždění** - jak se mění zpoždění jednotlivých paketů během přenosu.

Existují 3 základní přístupy, jak zajistit QoS v sítích nad IP: TOS směrování, Integrované služby a Rozlišované služby.

Související odkazy

Více informací o tom, jak například nakonfigurovat omezení rychlosti pomocí *cbq*, naleznete v sekci Konfigurace v Linuxu, podsekce Metody omezování šířky pásma.

6.1 TOS (Type of Service)

Jedná se o pole, ze kterého se používá prvních 5 bitů. Určuje se priorita a typ služby.
Požadavky na uživatele jsou popsány v RFC 1122, 1123 a jejich dodatcích. Používá se v protokolech OSI IS-IS Intra-domain Routing Protocol (RFC 1142) a OSPF (RFC 2328).

D (Delay) - přenos s minimálním zpožděním
T (Throughput) - přenos s maximálním výkonem
R (Reliability) - přenos s maximální spolehlivostí
C (Cost) - přenos nejméně nákladný

Přenos bez zvláštních požadavků TOS=0
Aby nastavení TOS bylo účinné, musí být podporováno všemi směrovači na přenosové cestě

8 9 10 11 12 13 14 15 C R T D

6.2 Integrované služby (Integrated Services) - intserv

V tomto případě aplikace oznámí počítačové síti své požadavky na přenos dat, požaduje určité QoS počítačové sítě, například určitou minimální průchodnost a určité maximální zpoždění.
Počítačová síť ověří zda je k dispozici dostatek prostředků pro uspokojení požadavku a rozhodne se, že:

- požadavkům vyhoví (admission control) - počítačová síť musí informovat všechny komponenty (směrovače). např. určitá šířka pásma spoje mezi dvěma směrovači, velikost fronty paketů uvnitř směrovače...
- požadavkům nemůže vyhovět (spojení není provedeno a aplikace se může rozhodnout, zda požádá o méně náročné QoS počítačové sítě).

Proto existuje **RSVP** (Resource Reservation Protocol) démon, který poskytuje 3 rozhraní: pro aplikace, řídící pro kernel a pro směrování. Tento protokol je dost složitý a má velkou režii.
Existují tedy i jednodušší rezervační protokoly, např. **YESSIR.** Vše probíhá zatím na experimentální

úrovni a ve výsledku jsou tyto protokoly příliš náročné (vysoká režie).
Neustále se zvyšuje rychlost průchodu paketů směrovači a je tedy třeba maximálně zjednodušit zpracování jednotlivých procházejících paketů a minimalizovat objem stavové informace, kterou musí směrovače o jednotlivých spojeních udržovat.
Dnes se tedy moc nevyužívá, přechází se na rozlišované služby.

6.3 Rozlišované služby - diffserv

V případě rozlišovaných služeb [diffserv] aplikace neoznamuje předem počítačové síti své požadavky na QoS. Rezervační protokoly se zde nevyužívají.
Směrovače neudržují žádnou stavovou informaci o jednotlivých spojeních.
Používá se zde pole TOS v protokolu IP. Tyto položky nastavují hraniční směrovače a vnitřní routery jej pak podle toho směrují.
Existuje několik přístupů k diferenciovaným službám:

- Zajištěné služby (Assured services) - každému uživateli je staticky nastavena konkrétní kapacita. Pokud ji uživatel překročí, data navíc už nemají zajištěnu stejnou kvalitu služeb.
- Premium services - pro danou službu je zajištěna špičková kapacita, která je zajištěna, ale nesmí být překročena. Daný proud je "uhlazen" na tuto hranici, aby nedocházelo ke špičkám.
- Víceuživatelské sdílení (User Share differentiation) - uživatel má garantovánu minimální kapacitu. Aktuální volná kapacita je pak sdílena všemi uživateli podle jejich domluvené kapacity.

PHB (Per Hop Behaviour) - řeší otázku, co dělat s pakety se stanovenou prioritou. IETF definuje 2 typy:

- AF (Assured Forwarding) - pakety rozděleny do tříd a paket v dané třídě má přiřazenu hodnotu přednosti před zahozením
- EF (Expedited Forwarding) - minimalizuje zpoždění a rozptyl až do výše provozního profilu, zbytek je zahozen.

7. Vysvětlení pojmů

Zde budou vysvětleny základní pojmy, které souvisí s danným tématem - filtrování paketů - a je tedy dobré je pochopit, jelikož pak budeme lépe rozumět danému problému a bude tedy jednodušší případně i experimentovat při následné konfiguraci.

7.1 NAT (Network Address Translation)

Tato tabulka slouží pro překlad síťových adres.

Využívá se ke dvěma hlavním účelům:

1. Skrývá vnitřní strukturu sítě.
2. Převádí neveřejné adresy na jednu veřejnou adresu, abychom mohli i ze strojů z vnitřní sítě přistupovat ven.

Obsahuje tři řetězce, které se však nepoužívají k filtrování paketů (i když i to je možné), ale jak už její název napovídá, ke změnám adresy datagramu. Funguje to tak, že pokud paket při průchodu vyhoví zadanému pravidlu, tak mu je podle určeného vzoru změněna adresa jeho odesílatele resp. příjemce podle určeného vzoru. Podle toho hovoříme buď o překladu adres odesílatele (SNAT - Source NAT) nebo překladu adres příjemce (DNAT - Destination NAT).

A jak vůbec vypadá pouť datagramu NAT tabulkou?

Příchozí pakety, ať už jejich destinace jakákoliv, prochází řetězcem PREROUTING, kde jim můžeme měnit adresu příjemce, tedy DNAT.
Odchozí pakety, bez ohledu na odesílatele zase mohou být SNATovány (zamaskování adresy odesílatele)

v řetězci POSTROUTING.
Zvláštním případem je odchozí řetězec OUTPUT, který lze použít k DNATování paketů vzniklých pouze na lokálním počítači. V praxi se však OUTPUT DNAT používá jen zřídka.

Uvedu jednoduchý příklad:

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

router tedy bude nahrazovat IP odcházejících paketů svojí IP adresou.

Výhody

- skrytí sítě - struktura sítě není viditelná zvenčí
- cena - je potřeba pouze jedna veřejná IP adresa
- není třeba zvláštní konfigurace klientů - stačí jako bránu nastavit server provádějící NAT
- není potřeba úprava aplikací

Nevýhody

- skrytí sítě - příchozí dotazy z vnější sítě nemůžou přistupovat dovnitř naší sítě, dokud není komunikace iniciována zevnitř nebo pokud nemáme speciální software pro přeposílání specifických portů
- problémy při použití DHCP - spravování překladu při dynamickém přidělování adres ve vnitřní síti

7.2 Mangle

Tato tabulka zavádí nový přepínač pro výběr paketu:

- mark - pravidlo uspěje pokud byl paket dříve označen.

  ```
  iptables -t mangle -A INPUT -m mark --mark 1
  ```

akce, které s tím souvisí:

- **MARK** - označí paket. pozor značka není součástí paketu -> po opuštění počítače, kde vznikla, se ztratí..

  ```
  iptables -t mangle -A PREROUTING -p tcp --dport 22 -j MARK --set-mark 2
  ```

- **TOS** - změní TOS paketu

  ```
  iptables -t mangle -A PREROUTING -p TCP --dport 22 -j TOS --set-tos 0x10
  ```

- **TTL** - změní TTL paketu

  ```
  iptables -t mangle -A PREROUTING -i eth0 -j TTL --ttl-set 64
  iptables -t mangle -A PREROUTING -i eth0 -j TTL --ttl-dec 1
  iptables -t mangle -A PREROUTING -i eth0 -j TTL --ttl-inc 1
  ```

7.3 DNAT (Destination NAT)

Když je modifikována cílová adresa prvního paketu, pak je to Cílová NAT - například se změní cíl.
Cílová NAT je vždy prováděna před započetím směrování, těsně po přijetí paketu z drátů.
Port-forwarding, load-sharing a transparentní proxy jsou speciální formou DNAT.

```
iptables -t nat -A PREROUTING -p tcp -d 15.45.23.67 --dport 80 -j DNAT --to-
destination 192.168.1.1-192.168.1.10
```

7.4 SNAT (Source NAT)

Když je modifikována zdrojová adresa prvního paketu, pak je to Zdrojová NAT - například se změní zdroj vysílání. Zdrojová NAT je vždy prováděna až po dokončení směrování, těsně předtím než je paket vyslán po drátech. Masquerading je speciální forma SNAT.

- **SNAT** - mění zdrojovou adresu/port.

```
iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to-source 194.236.50.155-
194.236.50.160:1024-32000
```

- **MASQUERADE** - forma SNATu vhodná pro dynamicky přidělovaná IP (např. vytáčená spojení), kdy maskujeme stroje s neveřejnými IP za IP rozhraní firewallu na venkovní síti. Navíc zahazuje tabulku spojení po shození rozhraní.

```
iptables -t nat -A POSTROUTING -p TCP -j MASQUERADE --to-ports 1024-31000
```

7.5 PAT (Port Address Translation)

Princip je skoro stejný jako u NAT, ale jde o překlad adres portů. Je to vlastně funkce obstarávající pomocí routerů komunikaci se všemi ostatními v síti (třeba jako internet) aniž by odhalila jejich vlastní privátní IP-adresy. Všechny odchozí pakety mají jejich IP adresu překládanou do routerovských externích IP adres. Odezvy přijdou zpět na router, který je přeloží zpět na privátní IP adresy původního hosta a jsou tedy připraveny ke konečnému postoupení.

7.6 Source Routing

Hned na začátku musím poznamet, že source routing neni způsobem směrovaní, jak by slovíčko "routing" napovídalo. Místo toho jde o jeden konkretní druh "mostění" (bridging), neboli o jeden konkretni zpusob fungovaní mostu. Samotný "source routing" vyvinula firma IBM pro své sítě Token Ring.
Vše je založeno na myšlence, že způsob průchodu datových rámců skrz jednotlivé mosty se určí předem a potřebné pokyny k průchodu takto zvolenou trasou se vloží do každého jednotlivého rámce. Tyto pokyny přitom mají formu linearního seznamu mostů, přes které má datový rámec postupně projít. Podstatná je na celé věci skutečnost, že o celé trase přenosu datových rámců rozhoduje již jejich odesílatel - odsud přívlastek "source" (doslova: od zdroje). Samotný termín "routing" je v této souvislosti do značné míry opodstatněný, protože odesílatel zajiš'tuje to, co se jinak nazývá směrováním (tedy rozhodovaní o směru přenosu). Problém je ovšem v tom, že rozhodnutí odesílatele je naplňováno na té úrovni (na úrovni tzv. linkové vrstvy), na které jednotlivá přepojovací zařízení pracují jako mosty, zatímco směrování je realizováno na úrovni bezprostředně vyšší (na úrovni tzv. síťové vrstvy).
Source routing je tedy ve skutečnosti "mostěním", a nikoli směrováním. Některé odborné prameny proto používají raději termín "source route bridging", který již uvádí vše na pravou míru.
Zajímavé je na celé věci i to, podle čeho vlastně odesílatel volí nejvhodnější trasu, kterou pak zakóduje do každého odesílaného rámce: na všechny strany vyšle speciální "průzkumný" rámec, který se sám následně šíří do všech existujících směrů, dokud nedojde k hledanému cíli. Od něj se rámec vrací zpět ke svému původnímu odesílateli a nese v sobě informaci o trase, kterou přitom prošel.

8.0 IPTables

Jelikož je tento mohutný a silný nástroj všude na internetu do hloubky popsán, nemá smysl ho zde dalekosáhle rozebírat. Uvedu zde tedy jen nejzákladnější informace, o tom, co to vůbec je a k čemu slouží a také srovnání se starší utilitou *ipchains*.
Nastavíme si jádro a řekneme si jak vytvářet firewall.

IPtables je robustní prostředek na administraci IP paketového filtru, je to nástupce *ipchains/ipfwadm*.

Výhody

- Rychlý, flexibilní, nenáročný
- Flexibilní změna pravidel (nemusí se rebootovat, pokud změníme nějaké pravidlo)

Nevýhody

- Analyzují se jen hlavičky a parametry paketu, nejedná se o *"content filter"*
- Hlavičky můžou být upravené
- Pokud budeme mít velkou síť, tak se budeme hůře orientovat v pravidlech (je dobré mít neustále přehled o tom, co chceme a co děláme)

Srovnání IPTables a IPChains

- IPchains umožňoval jen kontrolu shody SYN flagu, IPTables umožňuje match na SYN, ACK, FIN, RST, URG, PSH, ALL, NONE
- Stavovost - povoluje blokování neautorizovaných nových spojení skrz firewall, i když by normálně pakety byly povolené.Lze aplikovat i na jiné protokoly, nejen na TCP
  Stavová politika - routovací rozhodnutí můžou být založena na následujících stavech:

  NEW - vytvoření nového spojení (SYN flag), nepovolujeme na INPUT řetězec, pokud neběží server
  RELATED - spojení vztahující se nějakým způsobem na už existující spojení
  ESTABLISHED - součást existujícího spojení
  INVALID - nepatří žádnému spojení

  ```
  iptables -A FORWARD -m state --state ESTABLISHED, RELATED
  -j ACCEPT
  ```

- Adresový překlad NAT, IPChains umožňoval jen Maškarádu, IPTables podporuje DNAT, SNAT, REDIRECT, MASQUERADE
- Podpora "Packet Mangling" - změna "mangling" informací v hlavičkách paketů po dobu routovacího procesu (TOS, Marking)
- Ochrana před log-floodingem DOS útoky limitovaným množstvým případných shod v průběhu minuty (dá se aplikovat nejenom na LOG), podpora LOG prefixu a warning levelů (schopnost logování detailů)

  ```
  iptables -A INPUT -s 192.168.0.0/16 -m limit --limit 1/s
  -j LOG

  iptables -A INPUT -s 192.168.0.0/16 -j LOG --log-level warn
  --log-prefix "spojení:"
  ```

- Možnost kontroly (matchování) paketů na základě různých kritérií (MAC, TOS, LENGTH, TTL, ...)
- Modul LOG je separovaný "target" a nemůže být kombinovaný najednou v jednom pravidle jako tomu bylo u IPChains, musí být vytvořeno nové pravidlo (véc pravidel = větší flexibilita).
- REJECT byl zahrnut do externího modulu.

Nastavení jádra (2.4.x)

V *Networking options* vyberete:

```
CONFIG NETFILTER
CONFIG IP NF CONNTRACK
CONFIG IP NF FTP
CONFIG IP NF IRC
CONFIG IP NF IPTABLES
CONFIG IP NF MATCH LIMIT
CONFIG IP NF MATCH MAC
```

```
CONFIG IP NF MATCH MARK
CONFIG IP NF MATCH MULTIPORT
CONFIG IP NF MATCH TOS
CONFIG IP NF MATCH LENGTH
CONFIG IP NF MATCH TTL
CONFIG IP NF MATCH TCPMSS
CONFIG IP NF MATCH UNCLEAN
CONFIG IP NF MATCH OWNER
CONFIG IP NF MATCH STATE
CONFIG IP NF FILTER
CONFIG IP NF TARGET REJECT
CONFIG IP NF TARGET MIRROR
CONFIG IP NF NAT
CONFIG IP NF TARGET MASQUERADE
CONFIG IP NF TARGET REDIRECT
CONFIG IP NF TARGET LOG
CONFIG IP NF TARGET TOS
CONFIG IP NF TARGET TCPMSS
CONFIG IP NF COMPAT IPCHAINS
CONFIG IP NF COMPAT IPFWADM
```

Vytváření firewallu

- Firewall vytváříme vždy technikou *"všechno zakaž, něco povol", nikdy ne, však naopak.*

```
iptables -F
iptables -X
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
```

- Maškarádování vnitřní sítě

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

# pri dynamicky pridelovane IP (DHCP, PPP, ..)
iptables -t nat -A POSTROUTING -o eth0 -j MASQUARADE

# pri pevne staticke IP
iptables -t nat -A POSTROUTING -s $INT_NETWORK -o eth0
-j SNAT --to $EXT-INTERFACE

# povoleni odpovedi a vztahujicich se spojeni
iptables -A FORWARD -m state --state ESTABLISHED, RELATED
-j ACCEPT
iptables -A FORWARD -o $EXT-INTERFACE -s $INT-NETWORK
-j ACCEPT
```

- Povolení v INPUT chainu příslušných serverů (DNS, SSH, FTP, HTTP, ..)
- Aplikovaní DROP, REJECT, LOG na všechny nepropuštěné pakety

9. Modifikace IP paketů

Pokud chceme modifikovat IP pakety (nemyslí se tím IP adresu), využiju k tomu již známé informace o SNAT a DNAT. Použijeme k tomu nástroj *iptables*, který je již také známý.
Pěknou stránku, kde je popsáno jak na to, naleznete na
http://www.netfilter.org/documentation/HOWTO/cz/NAT-HOWTO-6.html
Pokud jste už zapomněli, co to jsou Zdrojová NAT, Cílová NAT (což se někdy stává - mě taky :-( ) a jak fungují, naleznete to též na stránce uvedené dříve a nebo se podívejte do sekce Vysvětlení pojmů.

10. Srovnání Linuxu a BSD systému

Když bych měl srovnat tyto dva systémy, tak bych se měl asi zajímat převším otázkou bezpečnosti. Přijde mi, že BSD systémy jsou sice asi nejvhodnější systémy na firewall a to proto, že mají nejlepší bezpečnostní zajištění vůbec, ale na druhou stranu jsou dost pomalé, zaostávají ve všech výkonostních testech za ostatními distribucemi. Podle výrobců to ale nebylo účelem, aby systém byl rychlý, ale především bezpečný.

Co se týká filtrace paketů, OpenBSD řeší odvěký problém - aktivní FTP protokol - po svém. Na internetové bráně musí běžet program ftp-proxy, který zařizuje vzájemné propojování FTP klientů a serverů.
PF je tzv. stavový firewall, což znamená, že se posuzuje pouze první paket a pokud projde, další pakety tvořící spojení vytvořené tímto prvním paketem už neprocházejí pravidly. Informace o tomto aktivním spojení se nazýva stav (state). To velmi zjednodušuje psaní pravidel. Paketový filtr je tedy *stavový*, ale stavy kolem ftp-proxy zatím neumí sám vysledovat. Takže chcete-li aktivní FTP, musíte povolit spojení odkudkoliv z portu 20 na vysoké porty firewallu.
Linuxový netfilter je v tomhle mnohem lepší. Některým FTP klientům navíc nevyhovuje, že datové spojení jde z adresy firewallu, nikoliv z původního FTP serveru.

11. Konfigurace v Linuxu

Zde jsou uvedena jednotlivá nastavení v Linuxu, např. skript s použitím asi nejmocnějšího nástroje pro filtrování paketů - *iptables*, nebo jak nastavit šířku pásma.

11.1 Vlastní skript pro nastavení firewallu s použitím iptables

(Konfigurováno na Debian GNU/Linux testing)
Jako perličku nakonec celého vyprávění o filtrování paketů jsem dodal asi mojí největší zkušenost v Linuxu a to samotné nastavení firewallu (plně funkční, takže kdo se nechce zabývat zdlouhavým studováním všech možných pravidel, nechť ho zneužije) do takové míry, aby vše filtroval podle standardních postupů a pravidel - iptables. Jelikož bych chtěl zachovat ale bezpečnost tohoto serveru, na kterém mimo jiné mám uložené i tyto stránky, tak jsem byl donucený změnit IP adresu v celém skriptu. Doufám, že nikde nezneužije tento výtvor k neprospěchu, ale spíše ho využije ku prospěchu lidstva. Myslím, že nemá smysl dělat více komentářů, jelikož kdo prostudoval celý tutorial, musel pochopit celou teorii filtrování paketů pod Linuxem.

Poznámka: Podotýkám jen, že jde tedy o stroj, který má dvě síťová rozhraní (eth0 a eth1)a filtruje pakety pomocí pravidel iptables z eth0 na eth1. Vše je uloženo ve skriptu, který se jmenuje *./skript*.

Je to trochu složitější, musí se totiž nastavit kompletní iptables, ale to už známe z předchozího vyprávění. Základem všeho je porozumět používání filtračních pravidel pomocí iptables. Kompletní soupis pravidel je ve skriptu a jsou k němu udělané komentáře, takže nemá cenu zde vypisovat celý skript. Kdyby náhodou přesto někdo nevěděl, najde pomoc v mém tutoriálu – sekce iptables. Jen bych chtěl zdůraznit, kde by měl být samotný skript umístěn. Je více možností, já uvedu tedy nastavení, které jsem použil já.

1. Ujistěte se, že se nespouští automaticky ipchains
   Zjistěte, že ipchains init.d skript není symbolicky připojen na S soubor v nějakém rc adresáři
2. Stopněte ipchains, jestli běží
   `service ipchains stop`
3. Spusťte lsmod, abyste viděli, jestli ipchains modul je stále nahrán. Jestliže je, tak ho odstraňte např. pomocí rmmod, abyste zaručili, že není aktivní.
4. Udělejte si symbolický link iptables init.d startovacího skriptu do run states 2, 3, 5.
   např. do adresáře /etc/rc2.d/
   nazvěte ho třeba
   `@S90skript`
5. Uložte si skript s pravidly třeba do adresáře /root/firewall. Spusťte ho, abyste nahráli nastavení

pravidel. Možná budete muset pustit příkaz dos2unix, aby došlo k odstranění konce řádků (carraige returns).
6. Uložte nastavení pravidel do /etc/sysconfig/iptables.
   To můžete udělat dvěma způsoby,
   ```
   service iptables save
   iptables-save > /etc/init.d/iptables
   ```
7. Nastavení pravidel bude obnoveno témto
   ```
   /etc/init.d/iptables script
   ```
   při startu.

To už je vše, mělo by to fungovat bez problémů, ale může se stát, že se nastavení pravidel pokaždé bude přepisovat na defaultní, pak je nutné nastavení uložit pomocí iptables save.

Pro ty, co by teď hledali výpis skriptu, tak je zklamu, protože už takhle to má dost stran :-) , ale najdete ho na webu http:/www.tuca.gamon.cz/projects/pd/iptables_skript

11.2 Metody omezování šířky pásma

Tyto metody se musí začít používat v případě, že dojde k zahlcení linky a vstupní fronty FIFO, jelikož by docházelo k zahazování paketů. Problém by byl tím ještě horší u spojovaných služeb, kde by docházelo k zdvojování paketů. Nyní tedy bude následovat stručný popis jednotlivých metod:

- **RED (Random Early Detection)** - při překročení určité hranice roste zahazování paketů lineárně
- **WRED (Weighted Random Early Detection)** - Oproti RED podporuje navíc priority.
- **TBF (Token Bucket Filter)** - idea: máme vědro s dírou, do kterého v pravidelných intervalech přitéká 1 token. Příchozí paket je propuštěn dále, jestliže je k dispozici potřebný počet tokenů ve vědru (podle jeho délky), jinak je zahozen.
- **SFQ (Stochastic Fairness Queuing)** - Pakety jsou rozděleny do toků. Ty jsou pak obsluhovány cyklicky (Round Robin).
- **WFQ (Weighted Fairness Queuing)** - Stejné jako SFQ, ale fronty mají priority.
- **CBQ (Class Based Queuing)** - Pakety jsou řazeny do tříd, kde jsou dále zpracovány.
- **HTB** - Obdobné CBQ.
- **Prio** - Existuje několik prioritních front.

Pojmy, se kterými bychom se mohli setkat:

- **policing** - jedná se o mechanismus, který udržuje provoz na vstupu i výstupu tak, aby nepřekročil povolenou šířku pásma pomocí zpožďování nebo zahazování paketů (v Linuxu se pouze zahazuje, neprovádí se zpožďování, neexistovalo tady žádné řízení na vstupu, na rozdíl např. od hardwareových routerů CISCO, i když současné kernely by již toto měly umět díky Ingress Qdisc - CONFIG_NET_SCH_INGRESS, neviděl jsem ale nikde příklady použití)
- **shaping** - metoda zpožďování paketů tak, aby na výstupu nepřekročily danou povolenou šířku pásma. Takto se také nazývá metoda zahazování paketů za účelem snížení provozu.
- **scheduling** - rozhodování, které pakety jsou důležitější a půjdou ven přednostně před ostatními.

11.2.1 CBQ (Class-Based Queueing)

Jedná se o sh-skript usnadňující nastavení omezení šířky pásma.
Používá nástroje z balíku iproute2.
Slouží k omezení šířky pásma a k nastavení priorit.
Funguje pouze s ethernetem a ARCnetem.

Použití skriptu CBQ

Skript *cbq.init* se nachází na adrese http://uf.kadan.cz/cbq

První věc, kterou musíme udělat je, že musíme zakompilovat do kernelu podporu pro QoS.
Volba *Networking options / IP: advanced router*, a v jádře buď přímo, nebo jako moduly jednotlivé QoS

moduly.
Pak nainstalujeme iproute a do adresáře */etc/init.d/* vložíme skript *cbq*.

```
# cp cbq.init-v0.7 /etc/init.d/cbq-init
chmod u+x /etc/init.d/cbq-init
```

Nastavíme cestu k adresáři s konfiguracemi vložením řádku

```
CBQ_PATH=/etc/cbq
```

do souboru. Skript připravíme k použití příkazem

```
# update-rc.d cbq defaults
```

Zbývá nám připravit konfiguraci pro CBQ. Ta sestává ze souborů v adresáři */etc/cbq* který musíme také vytvořit.

Příklad (Ukázka omezení rychlosti na eth1)

```
DEVICE=eth1,10Mbit,1Mbit
RATE=32Kbit
WEIGHT=3Kbit
PRIO=5
RULE=192.168.254.0/24
```

Související články:

Velice pěkný článek, který popisuje návrh CBQ a samotnou konfiguraci, najdete na stránkách LinuxZone, Praktické využití CBQ

11.2.2 RShaper

RShaper je modul do jádra. Je velmi jednoduchý na instalaci i použití. Pro řadu jednodušších síťových konfigurací je zajímavý právě svou jednoduchostí.
K modulu do jádra je k dispozici ovládací program **rshapectl.**

11.2.2.1 Instalace RShaperu

K překladu modulu budeme potřebovat hlavičkové soubory aktuálně bežícího jádra. Pokud jsme si překládali jádro sami, nastavíme proměnnou KERNELDIR a adresář se zdroji. Pokud ne, nainstalujeme hlavičkové sooubory. V ukázce používám jádro 2.4.18-686

```
# apt-get install gcc glibc-dev kernel-headers-2.4.18-686
# tar xzf rshaper-2.01.tar.gz
# cd rshaper-2.01
# export KERNELDIR=/usr/src/kernel-headers-2.4.18-686
# make
# make install
# depmod -a
```

Po překladu můžeme modul zavést a odzkoušet

```
# modprobe rshaper
```

Po odzkoušení upravíme konfiguraci aby se modul zaváděl při startu systému. Do souboru */etc/modules* přidáme řádku

```
rshaper
```

Do adresáře */etc/modutils* vložíme soubor *rshaper* s obsahem

```
options rshaper mode=2
```

A přegenerujeme konfiguraci modulů

```
# update-modules
```

11.2.2.2 Konfigurace RShaperu

První věcí kterou si musíme určit je mód ve kterém bude rshaper pracovat. Tento se zadává jako číselný parameter mode a může nabývat hodnot
0 - filtrace odchozích paketů
1 - filtrace příchozích paketů
2 - obousměrná filtrace příchozích i odchozích paketů (je nutné mít jádro 2.4)

Další část konfigurace se provádí programem rshaperctl.

```
rshaperctl [-nt] [address[/mask] bytes-per-second [queue-len]]
```

12. Konfigurace ve FreeBSD

Pokud chceme provést vlastní konfiguraci firewallu a nechceme k tomu použít konfigurační soubor, využijeme nástroje ipfw.

Jeho funkce lze popsat asi takto:

- přidávání / mazání pravidel,
- vypisování pravidel a jejich skupin /chains – řetězců/ a čítačů paketů,
- smazání všech pravidel,
- resetování čítačů.

12.1 Instalace IPFW

IPFW lze celkem jednoduše použít hned po instalaci systému, protože je již zkompilován jako modul:

```
kldload ipfw
```

Nastavený je implicitně na zakázat vše. Konfigurace se standardně načítá ze souboru /etc/rc.firewall. Tento soubor obsahuje seznamy pravidel pro firewall ve spustitelném tvaru (spouští klientský program ipfw).

Takto lze načíst otevřenou konfiguraci:

```
sh /etc/rc.firewall open
```

Další parametry firewall uzavírají nebo nastavují různé skupiny pravidel. Tyto parametry je možné specifikovat ve startovacím souboru /etc/rc.conf pomocí příkazu:

```
firewall_type=”typ”
```

kde pak startovací /etc/rc vyvolá právě /etc/rc.firewall s příslušným parametrem. Startovací /etc/rc.conf musí dále obsahovat:

```
firewall_enable=”yes”
```

Pro běh firewallu jako součást kernelu, je potřeba zadat do konfiguračního souboru pro kompilaci kernelu následující řádky:

```
options IPFIREWALL Přidává
podporu pro firewall
options IPFIREWALL_VERBOSE Přidává
kód pro logování pomocí syslog
options
IPFIREWALL_VERBOSE_LIMIT=10 Nastaví maximum položek
záznamů o pro jedno pravidlo
options IPDIVERT umožní
přesměrovávat pakety na jiné porty pro NAT
options
DUMMYNET balancování zátěže
```

Takto zkompilovaný kernel po spuštění systému automaticky spustí firewall, a nastaví jej podle /etc/rc.conf a /etc/rc.firewall.

12.2 Samotná konfigurace IPFW

Konfigurace IPFW je celkem jednoduchá a stačí k tomu vědět jen pár základních pravidel.
Budou následovat příklady, které by měly danný problém přiblížit více:

```
ipfw add deny tcp from evil.crackers.org to nice.people.org 23 via ed0
```

zakazuje komunikaci protokolem TCP od evil.crackers.org do portu 23 stanice nice.people.org jdoucí přes rozhraní ed0

```
ipfw add deny log tcp from evil.crackers.org/24 to nice.people.org
```

zakazuje a loguje komunikaci protokolem TCP ze sítě crackers.org a 24-bitovou maskou na jakýkoliv port stroje nice.people.org

Implicitně nastavený řetězec po instalaci, který zakazuje jakýkoliv přístup zvenčí:

```
allow 100 ip from any to any via lo0
deny 200 ip from any to 127.0.0.0/8
deny 65535 ip from any to any
```

Čísla označují pořadí v jakém se budou pakety porovnávat s pravidly řetězce, číslem 65535 je označené implicitní pravidlo, které nelze změnit, a které určuje implicitní chování firewallu.

Další příklady:

```
ipfw flush
```

smaže všechny řetězce a ponechá pouze implicitní pravidlo '*deny 65535 ip from any to any*', tedy úplně uzavře síťovou komunikaci.

```
ipfw add fwd localhost,23 log tcp from evil.crackers.org to
evil.crackers.org 23
```

přesměruje pakety jdoucí z evil.crackers.org na lokální počítač na port 23 na fbi.gov port 23.

Syntaxe příkazu ipfw:

```
ipfw [rule_number] [set set_number] [prob match_probability] action [log [logamount
number]] body

rule_number ..... vzestupné číslo pravidla (implicitní a neměnné 65535).
udává pořadí zpracování
set set_number ..... číslo množiny pravidel (1-30); lze takto seskupovat
pravidla a manipulovat s nimy
prob match_probability ... float od 0 do 1; používá se s dummynet (například simulace
```

```
cest paketů)
action ................... viz výše uvedené příklady
log [logamount number] ... zda logovat uvedené pakety pomocí syslog a jejich počet
body ..................... viz výše uvedené příklady
```

13. Nástroje pro měření a testování

Pro měření a testování se dají použít např. tyto programy:

ttcp - pro testování propustnosti TCP nebo UDP spojení. Na jednom konci se spustí v módu odesilatel (transmit), na druhém v módu příjemce (receive). Nejspíš bude už součástí distibuce.
iptraf - obecný nástroj pro monitorování síťového provozu.

14. Závěr

Všechny informace jsem čerpal z internetu, stránek s tutoriály nebo z příslušných RFC dokumentů (), ptal jsem se také svých kolegů, kteří už s filtrováním paketů měli nějaké zkušenosti. Přidal jsem také moje zkušenosti, které jsem měl z minulosti a nebo jsem je získal při řešení ne jednoho problému, když jsem se snažil nastavit svůj firewall. Věřím, že po přečtení těchto stránek si bude moci "linuxář" nastavit svůj vlastní firewall a bude tím ochráněn před "ošklivým" okolním světem.

Přínosy:
Za největší přínos této práce, kromě toho že jsem zvládnul úspěšně prostudovat velké množství materiálů v elektronické podobě a podle nich pak nastavit danný firewall , považuji i to, že jsem se zdokonalil v psaní HTML, a myslím si, že už nebudu nikdy používat k psaní webových stránek laciné frontendy, které přidávají do kódu nejsmírné množství "balastu". Budu tedy používat jedině a vždy buď nástroje jako je Bluefish a nebo nejlépe Tex. Ten mi přirostl celkem k srdci a myslím, že to je asi nejlepší nástroj v Linuxu, který jsem kdy používal pro tvorbu webových stránek. Zapomněl jsem totiž podotknout, že tato práce na začátku vznikala v aplikaci OpenOffice.org HTML Document, po jisté době jsem ale přešel k již zmiňovanému programu Bluefish 0.1.1.

15. Literatura

- http://www.root.cz/clanek/980
- http://www.underground.cz
- http://netfilter.kernelnotes.org
- http://www.linuxguruz.com
- http://www.linux.cz/linuxdoc/HOWTO
- Linux 2.4 packet filtering howto
- Linux 2.4 NAT HOWTO
- IP-Masquerade HOWTO
- Source-Route Bridging
- Linux IPCHAINS HOWTO
- iptables tutorial
- ftp://ftp.rfc-editor.org/in-notes/rfc1918.txt
- http://netfilter.samba.org
- http://people.unix-fu.org/andreasson/iptables-tutorial/iptables-tutorial.html
- http://www.malibyte.net/security/fwscripts.html
- http://linuxguruz.org/iptables/
- Část příručky FreeBSD věnující se firewallům
- Man stránka příkazu ipfw
- man iptables
- man ipchains