Bab IX - Keamanan IPV6

Iljitsch van Beijnum

Banyak orang suka berpikir (atau mengatakan) bahwa IPv6 lebih aman dari IPv4. Tapi itu terlalu sederhana. ini seperti mengatakan telur bulat. Sementara itu sulit untuk menyangkal bahwa telur menunjukkan banyak kebulatan, mereka tidak sama dengan $x^2+ y^2+ z^2= r^2$ formula yang menggambarkan sebuah bola yang sempurna. banyak aspek keamanan IPv6 yang di improvisasi dari IPv4 tapi beberapa part ipv6 merupakan komparasi dari ipv4 . Lebih penting lagi, berbicara tentang "keamanan" seolah-olah itu semacam bumbu yang dapat ditambahkan sesuai dengan seseorang Rasa membuat saya amat tidak nyaman. Menurut saya security adalah keaadan pikiran. Itu diketahui apakah salah atau tidak. dan mempersiapkan ketika mereka melakukan pula.Inibertentangan sifat manusia,yang mendorongkita untuk melestarikanenergi mental,menebak kemungkinan ataudiinginkanhasil untuk setiap tindakan yang diberikan, dan mengabaikanhasil lain yang mungkin.Sayangnya,itu orang-orang untuk mengambil data atau uang kita tidak keberatan menghabiskan sedikit lebih otak kekuasaan, dan Hukum Murphy memberitahu kita bahwa apa yang bisa salah melakukannya.

Intinya, bab ini akan menjelaskan perbedaan antara IPv6 dan IPv4 berkaitan dengan keamanan (yang baik dan yang buruk), serta menjelaskan bagaimana perangkat seperti keamanan standar paket filter berlaku untuk IPv6. Tidak ada diskusi tentang keamanan IPv6 lengkap tanpa pengobatan IPsec. IPsec adalah teknologi yang sangat menarik yang memiliki sifat keamanan tak tertandingi, tetapi itu belum hidup sampai potensinya

Perbedaan dari IPv4

Banyak perubahan antara IPv4 dan IPv6 memiliki implikasi keamanan. Hal ini terutama untuk penggunaan yang lebih luas dari ICMP, ruang alamat besar dan link-lokal Address.

Memanfaatkan Batas Hop

Dalam IPv4, itu perlu untuk menyaring pesan ICMP redirect pada link ke seluruh internet, seperti penyerang mungkin mencoba untuk membingungkan host dengan mengirimkan pesan redirect dipalsukan. Dalam IPv6, masalah ini bisa saja jauh lebih buruk, seperti dalam IPv6, penggunaan ICMP sangat diperluas. Namun, penulis RFC 1970 pada tahun 1996 dan penggantinya RFC 2461 pada tahun 1998 datang dengan trik pintar untuk menolak pesan ICMPv6 dikirim oleh penyerang jarak jauh. penemuan tetangga dan semua ICMPv6 lainnya jenis yang hanya digunakan pada subnet tunggal memiliki Batas Hop mereka diatur ke 255 oleh penyelenggara asal Sistem tersebut

Hal ini memungkinkan sistem penerima untuk menentukan apakah sebuah paket dikirim oleh sistem pada subnet yang sama atau dengan sistem remote. Jika pengirim dan penerima berbagi subnet, paket tidak bisa telah dilalui router, jadi Batas Hop tetap harus 255 pada penerima. Sebuah remote Penyerang dapat kerajinan paket dengan semua bidang lain ditetapkan untuk apa pun yang terbaik melayani tujuan penipuan nya, tapi dia tidak bisa membuat paket memiliki Hop Batas 255 di tempat tujuan. Jika penyerang mengirimkan paket dengan Hop Batas 255, penerima akan melihat nilai lebih rendah , router dilapangan akan dikurangi . Pengaturan sebuah Hop awal dibatasi rendah dari 255 jelas biasa melakukan setiap baik baik, dan Hop tertinggi tinggi Batas tidak mungkin karena 255 adalah nilai tertinggi dalam bidang 8-bit.

Sayangnya, meskipun fakta bahwa trik ini pertama kali didokumentasikan pada pertengahan 1990-an, banyak protokol lain yang menggunakan komunikasi link-local, seperti RIPng, OSPFv3, dan DHCPv6, tidak menerapkannya. Pada tahun 2004, mekanisme diperkenalkan ke BGP dengan nama "Generalized Mekanisme TTL Security "(GTSM) di RFC 3682. Karena kedua belah pihak perlu untuk mengimplementasikan GTSM

The Larger Address Space

Net telah melihat banyak cacing (kadang-kadang disebut "virus") beberapa tahun terakhir, tetapi "SQL Slammer“ atau "Sapphire“ cacing di Januari 2003 adalah sesuatu yang istimewa: karena seluruh cacing Adalah terkandung dalam paket UDP tunggal, penyebaran menilai cacing ini adalah belum pernah terjadi sebelumnya. Karena UDPtidak memerlukan jabat tangan atau umpan balik dari sisi lain, host yang terinfeksi hanya bisa mengirim di luar paket yang berisi worm untuk acak tujuan pada kecepatan maksimum. Ini diizinkan SQL Penjara untuk menggandakan jumlah sistem yang terinfeksi setiap 8,5 detik, menginfeksi 90% dari semua rentan host terhubung ke Internet dalam 10 menit. Ini membuatnya pertama dan sejauh ini hanya. memungkinkan host terhubung ke Internet dalam 10 menit. Hal ini membuat yang pertama dan sejauh ini hanya "Warhol worm.“ di Dalam IPv6, ruang alamat adalah 2^{96} kali lebih besar, sehingga menemukan sistem yang rentan akan mengambil 2^{96} kali lebih lama. Jadi dalam IPv6, jumlah host yang terinfeksi setiap $2^{96} \times 8{,}5$ detik, atau setiap 250000000000 miliar tahun. Saya pikir kita bisa setuju bahwa orang yang belum Diberi sistem mereka saat itu hanya menyalahkan diri mereka sendiri.

Dengan kata lain: ruang alamat besar IPv6 ini membuat tidak mungkin bagi penyerang untuk menemukan korban dengan secara acak memindai sistem . Namun, ditentukan dan penyerang pasien masih mungkin dapat menemukan sistem dengan pemindaian ditargetkan. Memindai / 64 subnet masih membutuhkan waktu terlalu banyak waktu, tetapi hanya memindai alamat IPv6 berasal dari alamat Ethernet MAC dari vendor tertentu "hanya" membutuhkan sekitar 17 juta paket. Sebuah koneksi DSL bisa bergerak nomor yang paket dalam hitungan jam, jadi scan seperti ini tidak sepenuhnya layak.

On-link Dangers

Fakta bahwa IPv6 selalu memiliki alamat link-lokal besar jika Anda ingin router untuk berkomunikasi meskipun mereka tidak berbagi subnet prefix atau jika Anda ingin membuat jaringan ad-hoc tanpa

konektivitas ke Internet. Tapi itu tidak begitu besar ketika sistem keuntungan konektivitas IPv6 (bahkan hanya di link lokal) tanpa Anda menyadarinya. Sekarang semakin banyak rasa Linux, BSD, dan lainnya UNIX atau UNIX-seperti sistem operasi gain IPv6-dukungan dalam kernel, itu menjadi cukup umum untuk sistem ini memiliki konektivitas link-lokal tanpa pemilik menyadarinya. Untuk menambahkan filter paket IP atau firewall perangkat lunak yang ada umumnya menyaring hanya IPv4 dan tidak mendapatkan di jalan paket IPv6. Kebanyakan sistem tidak bergantung pada jenis filter untuk menghindari koneksi yang tidak diinginkan, tapi itu sesuatu yang perlu diingat. Saat memindai seluruh subnet satu alamat pada satu waktu adalah tidak benar-benar pilihan, ada cara lain untuk menemukan sistem IPv6 terhubung ke subnet lokal. Salah satu yang jelas adalah ping pemain "ke semua host address multicast, seperti yang ditunjukkan pada Listing 9-1.

Listing 9-1. *Finding IPv6 Systems on the Local Subnet with a Multicast Ping*

```
# ping6 -I eth0 -c 2 ff02::1
PING ff02::1(ff02::1) from fe80::201:2ff:fe29:23b6 eth0: 56 data bytes
64 bytes from ::1: icmp_seq=1 ttl=64 time=0.078 ms
64 bytes from fe80::20a:95ff:fecd:987a: icmp_seq=1 ttl=64
time=0.366 ms (DUP!)
64 bytes from fe80::204:27ff:fefe:249f: icmp_seq=1 ttl=64 time=2.07
ms (DUP!)
64 bytes from fe80::20a:95ff:fef5:246e: icmp_seq=1 ttl=64 time=82.7
ms (DUP!)
64 bytes from ::1: icmp_seq=2 ttl=64 time=0.076 ms
--- ff02::1 ping statistics --2
packets transmitted, 2 received, +3 duplicates, 0% packet loss, time
1006ms
Rtt min/avg/max/mdev = 0.076/17.066/82.734/32.842 ms
```

Anda tidak perlu melakukan root untuk ini, tetapi di bawah Red Hat Linux, pengguna biasa tidak memiliki / Usr / sbin / dimana ping6 berada di jalan mereka. -I memasok interface outgoing. KAME berasal implementasi (keluarga BSD dan MacOS) juga menerima FF02 :: 1% antarmuka sintaksis. The -c 2 mengatakan ping6 bahwa itu harus mengirimkan dua pesan echo request. Mengirim hanya satu tidak akan berbuat baik apapun, ping6 kemudian berhenti setelah menerima balasan pertama. Dalam hal ini, itulah jawaban dari host itu sendiri, yang muncul sebagai balasan dari :: 1.

Node Informasi Query
KAME IPv6 jugamendukung ICMPv6 "simpulinformasi query" mekani sme untuk memberikan bahkan lebih informasi untuk perintah KAME dengan ping6. Itu kebanyakan orang menarik adalah sebagai berikut:

-a ag bertanya untuk target global unicast
-a al bertanya untuk link lokal address
-w bertanya untuk target hostname
-N mencoba untuk menemukan system pada lokal subnet

- ping6 –c 2 –I xl0 –a ag ff02::1 untuk bertanya global IPv6 untuk tiap KAME yang kompatibel dengan interface xl0
- ping6 –c 2 –I xl0 –N –a ag server untuk list alamat untuk tiap KAME yang kompatibel dengan sistem bernama server yang terkonek dengan xl0
- ping6 –c 1 –a ag www.kame.net untuk list global untuk alamat KAME yang kompatibel dengan www.kame.net
- ping6 –c 1 –W 2001:db8:31:5::2 untuk hostname yang dikonfigurasi KAME kompatibel IPv6 dengan alamat 2001:db8:31:5::2

filter

Dalam dunia tidak akan ada yang tidak butuh filter: biasanya lebih baik untuk membuat keputusan apakah akan mengizinkan jenis tertentu lalu lintas IP di tingkat aplikasi. aplikasi yang berbeda memiliki kebutuhan kontrol akses yang berbeda, dan mekanisme yang digunakan aplikasi untuk mendukung kebutuhan tersebut adalah, atau setidaknya bisa, cukup canggih. Ketika saya terhubung ke bank saya, saya tahu itu server mereka aku berbicara dengan karena sertifikat X.509 bahwa browser saya diperiksa sebelum ditampilkan kunci kecil di sudut jendela nya. Atau ketika saya terhubung ke server mail, server tahu itu saya karena Saya menyediakan login saya dan password yang hanya saya tahu. Dibandingkan dengan itu, penyaringan alamat IP dan nomor protokol agak primitif. Lebih buruk lagi, kegunaan dari jenis filtering adalah sangat dikurangi dengan fakta bahwa alamat IP cenderung berubah dari waktu ke waktu, dan penulis aplikasi dan pengguna kadang-kadang secara aktif bekerja di sekitar filter dengan menggunakan non standar atau bahkan dinamis pelabuhan angka. Juga, banyak bidang dalam paket IP dapat dengan mudah "palsu" oleh penyerang. Namun demikian, IP filter sering berguna atau diperlukan.

TCP Wrappers

Fungsi TCP wrapper adalah contoh yang baik dari sesuatu yang ditambahkan ke stereotip lingkungan UNIX pada satu titik dan yang tetap sana sementara dunia sekitar t berubah. Saya ragu bahwa banyak orang akan menggunakan pembungkus TCP dengan IPv6, tapi kadang-kadang itu baik untuk mempertimbangkan awal yang sederhana kami. Idenya adalah bahwa daemon dimulai oleh inetd, atau lainnya daemon yang dihubungkan dengan dukungan perpustakaan yang tepat, mendapatkan beberapa logging tambahan dan akses kemampuan kontrol. Awalnya, ada file /etc/hosts.allow dengan host yang diizinkan akses ke layanan dan file /etc/hosts.deny dengan, Anda dapat menebaknya, host yang ditolak akses ke layanan. Kemudian, kedua jenis klausa dipindahkan ke file hosts.allow. Listing 9-2 menunjukkan file hosts.allow sederhana.

- **Listing 9-2. *A hosts.allow File***

```
# wrap inetd daytime service
daytime : .example.com : deny
daytime : 10.53.64.0/255.255.192.0, [2001:db8:31:2::53] : allow
# The rest of the daemons are protected.
ALL : ALL \
: severity auth.info \
: twist /bin/echo "You are not welcome to use %d from %h.“
```

Dalam contoh ini, semua koneksi dari host di domain example.com menuju "siang hari“ layanan adalah ditolak tanpa lebih lanjut ribut. Meskipun TCP pembungkus tidak jatuh untuk paling jelas trik DNS, memeriksa DNS hanya menyediakan sangat tingkat terbatas keamanan. Host di 10.53.64.0/18 dan host dengan alamat 2001: db8: 31: 2 :: 53 diperbolehkan untuk menggunakan siang hari layanan. File hosts.allow bekerja pada "pertandingan pertama" dasar, sehingga ketika tuan rumah venus.example.com telah address 10.53.65.1, itu ditolak berdasarkan nama domainnya, dan alamat pertandingan di baris berikutnya tidak datang ke dalam bermain. versi yang berbeda dari TCP dukungan pembungkus yang sedikit berbeda hosts.allow klausa, tapi yang saya bisa tes tidak mendukung notasi prefix untuk IPv4 rentang alamat atau apapun yang cocok wildcard untuk IPv6. Klausa akhir log upaya apapun untuk menggunakan layanan (yang aktif dan menggunakan pembungkus TCP) dan mengirimkan kembali pesan kesalahan. Tergantung pada protokol, pesan ini mungkin atau mungkin tidak ditampilkan kepada pengguna.

Pada Red Hat 9 Linux, pembungkus TCP bekerja unreliably. Di bawah FreeBSD, mereka bekerja baik, dengan peringatan bahwa ketika daemon menerima koneksi masuk dan pembungkus TCP menolak sambungan, daemon melihat aneh penghentian dari permintaan yang masuk. Untuk itu MySQL daemon, ini sudah cukup untuk kecelakaan. Ini dapat diselesaikan baik dengan menggunakan packet filter seperti yang dibahas di bawah atau mengikat layanan ke alamat localhost sehingga upaya koneksi dari dunia luar tidak mungkin. Di MacOS 10.4, daemon inetd tidak mendukung pembungkus TCP,tapi daemon lain mungkin masih dikompilasi dengan TCPpembungkusmendukung.Iniadalah kasus untuk itu built-in sshd

Stateful Filtering to Replace NAT

Ketika beberapa host berbagi alamat IPv4 publik tunggal, ini membuat sulit untuk JaringanAlamat kotak Translation yang mengimplementasikan berbagi ini untuk menentukan apa yang harus dilakukan dengan masuk permintaan pembentukan sesi TCP

Dalam IPv6, asumsi adalah bahwa tidak ada NAT, sehingga tidak ada perlindungan otomatis terhadap lalu lintas masuk tanpa diundang baik. Untuk TCP, ini mudah diperbaiki dengan menyaring paket yang berisi header TCP dengan SYN bit set dan ACK bit dibersihkan. Hal ini tidak mungkin untuk membangun sesi TCP dari luar ke dalam, sementara semua paket TCP lainnya, seperti pembentukan sesi pengakuan untuk sesi dimulai dari dalam atau paket milik sudah sesi didirikan, bisa melewati. Untuk UDP, hal itu akan lebih rumit, karena tidak ada cara untuk menentukan apakah paket UDP adalah permintaan yang masuk atau membalas permintaan keluar sebelumnya dengan hanya melihat paket. Jadi untuk memblokir lalu lintas UDP tanpa diundang, itu perlu untuk menerapkan filter stateful yang melacak paket UDP keluar sehingga dapat menentukan apakah paket yang datang diundang atau tidak. Sebuah filter stateful juga dapat menolak paket TCP yang akan lolos filter normal yang hanya terlihat di bendera TCP. (Namun, host penerima akan menolak paket tersebut tetap karena merekam tidak termasuk sesi TCP valid.) Last but not least, filtering stateful sering mendapat pekerjaan yang dilakukan dengan jumlah yang lebih kecil dari aturan filter.

Ip6 tables Linux

2,2 Linux kernel melakukan packet IP penyaringan dengan ipchains. Pada kernel 2.4, yang disukai Metode untuk ini adalah iptables. Nama untuk filter yang sebenarnya adalah "subsistem Netfilter," tapi aku menemukan nama-nama ini tidak mungkin untuk mengingat, sehingga lebih mudah untuk berbicara tentang executable yang menyediakan user interface.red Hat Linux dilengkapi dengan iptables (Versi IPv4) tetapi tidak dengan ip6tables (versi IPv6). Listing 9-3 menunjukkan bagaimana untuk men-download dan menginstal ip6tables, bersama dengan versi yang lebih baru dari iptables yang membutuhkan ip6tables RPM. Output dari perintah yang tersisa. dengan jumlah yang lebih kecil dari aturan filter.

- **Listing 9-3.** ***Downloading and Installing ip6tables***

```
# service iptables stop
# chkconfig iptables off
# wget http://download.fedoralegacy.org/redhat/9/updates/i386/➥
iptables-ipv6-1.2.8-8.90.1.legacy.i386.rpm
# wget http://download.fedoralegacy.org/redhat/9/updates/i386/➥
iptables-1.2.8-8.90.1.legacy.i386.rpm
# rpm --upgrade iptables-1.2.8-8.90.1.legacy.i386.rpm
# rpm --install iptables-ipv6-1.2.8-8.90.1.legacy.i386.rpm
# service ip6tables start
# chkconfig --level 345 ip6tables on
```

Red Hat dan ip6tables bersikeras bahwa ipchains, iptables, dan ip6tables yang saling eksklusif, sehingga dua baris pertama menghentikan iptables berjalan dan menghentikannya dari yang dimuat pada startup di masa depan. Dua baris berikutnya men-download file RPM, dan dua baris setelah itu meng-upgrade ip6tables yang ada dan menginstal ip6tables, masing-masing. Kedua garis hitam yang mulai ip6tables dan membuatnya mulai pada saat boot. Karena iptables hanya menangani IPv4 dan ip6tables hanya menangani IPv6, itu sangat besar nyaman tidak mampu untuk menjalankan kedua. Jadi aku mulai iptables lagi ketika ip6tables berlari, tanpa efek buruk langsung. Anda mungkin berbeda.

iptables mendapatkan namanya dari "tabel" yang berbeda yang pada gilirannya masing-masing berisi satu atau lebih "Rantai" dengan aturan filter. Tabel kita tertarik adalah default bernama "filter," yang secara default berisi tiga rantai: input, output, dan ke depan. Rantai masukan berisi Aturan yang berlaku untuk paket yang datang untuk konsumsi lokal. Rantai keluaran berisi aturan yang berlaku untuk paket lokal yang dihasilkan karena mereka meninggalkan sistem. Aturan dalam rantai maju beroperasi pada paket yang diteruskan ketika sistem bertindak sebagai router. Aturan filter dalam rantai dievaluasi satu demi satu, dan segera setelah paket cocok aturan, tindakan ditunjukkan diambil, meskipun ada aturan berikut. Listing 9-4 mengimplementasikan filter bahwa blok beberapa jenis lalu lintas.

- **Listing 9-4. *Filtering with ip6tables***

```
# ip6tables -A OUTPUT -p tcp --dport 25 -j DROP
# ip6tables -A INPUT -s 2001:db8::/32 -j DROP
# ip6tables -A INPUT -p icmpv6 --icmpv6-type echo-request -d ff02::1 -j DROP
# ip6tables -I INPUT -p tcp --syn -j DROP
# ip6tables -D OUTPUT -p tcp --dport 25 -j DROP
```

flag -A menambahkan aturan filter di akhir rantai, sementara -I menyisipkan mereka di awal dan -D menghapus aturan yang ada. (Nya juga memungkinkan untuk menggunakan nomor baris untuk pemesanan eksplisit.)Itu p tcp dalam aturan pertama sesuai paket TCP, tetapi dport 25 batas ini untuk paket dengan nomor port tujuan 25 (SMTP). Bendera j menentukan tindakan yang akan diambil. berarti DROP paket tersebut akan dibuang begitu saja, tanpa kembali pesan ICMP. Dokumentasi juga menyebutkan TOLAK (Bersama dengan beberapa pilihan lain), yang mengirim kembali ICMP unreachable, tapi ironisnya, upaya saya untuk menambahkan aturan filter dengan tindakan MENOLAK ditolak. Aturan-aturan ini bekerja di bawah asumsi bahwa kebijakan default untuk setiap paket yang tak tertandingi adalah untuk memungkinkan mereka melalui. Kalau tidak, kamu dapat membuat aturan filter yang menerima diizinkan lalu lintas (dengan ACCEPT bukan tindakan DROP), dan kemudian menolak segala sesuatu yang lain. Anda dapat mengubah kebijakan default untuk tabel dengan (misalnya) ip6tables -P DROP INPUT atau ip6tables –P MAJU MENERIMA. ip6tables -L daftar rantai dan kebijakan default. Lihat ip6tables halaman manual untuk informasi lebih lanjut tentang cara menggunakan ip6tables. Anda juga dapat menemukan informasi lebih lanjut tentang Netfilter secara umum di http://www.netfilter.org/.

MacOS and FreeBSD ip6fw

IP Firewall, atau ipfw di IPv4 dan ip6fw di IPv6, telah sekitar untuk waktu yang lama di FreeBSD.Hari-hari ini, banyak pengguna FreeBSD lebih IPF sebaliknya, yang akan kita bahas nanti. MacOS tidakdatang dengan IPF, tapi ip6fw diaktifkan secara default, sehingga pada Mac, menggunakan ip6fw adalah pilihan yang logis. Di bawah FreeBSD, Anda dapat mengaktifkan ip6fw dengan menambahkan baris berikut ke /etc/rc.conf:

ipv6_firewall_enable="YES"
ipv6_firewall_script="/etc/rc.firewall6"
ipv6_firewall_type="open“

Tujuan dari baris pertama jelas. Baris kedua menunjuk ke sistem yang disediakan script yang menyiapkan aturan filter IPv6 dan baris terakhir melewati argument "terbuka" untuk ini naskah. Silahkan lihat pada script di /etc/rc.firewall 6 untuk pilihan lain selain "terbuka", yang memungkinkan semua paket, atau "ditutup," yang menyaring semua paket. Tetapi jika Anda ingin menggunakan salah satu dari ini Pilihan lainnya, Anda perlu untuk menyesuaikan script. Aturan berikut selalu hadir:

65535 deny all from any to any

Jadi jika Anda ingin membiarkan sesuatu (atau semua), Anda harus menginstal satu atau lebih aturan dengan angka yang lebih rendah. Hal ini membuat ide yang sangat buruk untuk memiliki hanya ipv6 garis firewall mengaktifkan = "YES" di yang /etc/rc.d tanpa script untuk mengatur aturan filter. Anda dapat mengubah perilaku default untuk memungkinkan semua paket dengan kompilasi kernel dengan opsi khusus seperti yang dijelaskan di IPFIREWALL yang manusia halaman. Pada Mac, bagaimanapun, hal bekerja sedikit berbeda, dan tindakan default adalah untuk memungkinkan semua paket. Berbeda dengan Linux Netfilter / ip6tables, ip6fw hanya memiliki seluruh sistem daftar filter tunggal. Namun, Nya mungkin untuk menerapkan aturan filter individual hanya input atau output paket. Listing 9-5 tidak sama dengan ip6fw sebagai Listing 9-4 lakukan dengan ip6tables.

- **Listing 9-5.** ***Filtering with ip6fw***

```
% sudo ip6fw -q add unreach admin tcp from any to any 25 out
% sudo ip6fw -q add deny ipv6 from 2001:db8::/32 to any
% sudo ip6fw -q add deny ipv6-icmp from any to ff02::1 in icmptype 128
% sudo ip6fw -q add deny tcp from any to any setup in
% sudo ip6fw list
00100 unreach admin tcp from any to any 25 out
00200 deny ipv6 from 2001:db8::/32 to any
00300 deny ipv6-icmp from any to ff02::1 in icmptype 128
00400 deny tcp from any to any in setup
65535 allow ipv6 from any to any
% sudo ip6fw delete 100
```

Perintah ip6fw itu sendiri adalah sama pada FreeBSD dan MacOS, tapi karena root tidak diaktifkan secara default pada MacOS, itu perlu untuk menggunakan alat sudo untuk mengeksekusi istimewa perintah. Baris pertama menginstal aturan filter yang pelabuhan blok keluar TCP 25 paket. Kapan paket seperti hits filter, jenis ICMPv6 "unreachable" kode "administratif diblokir“ Pesan dikirim kembali sehingga aplikasi segera mengembalikan kesalahan. Blok baris berikutnya semua IPv6 paket dari berbagai dokumentasi, terlepas dari tujuan mereka dan apakah paket adalah masuk atau keluar. Sana Adalah ada ICMPv6 unreachable atau TCP reset untuk filter dengan Sebuah menyangkal tindakan. Baris blok ketiga semua masuk ICMPv6 gema permintaan paket (tipe 128, lihat Bab 8) dengan alamat multicast semua-node sebagai tujuan mereka. Blok baris kelima semua
masuk sesi TCP paket pendirian.

IPFilter

IPFilter adalah filter IP yang tersedia untuk beberapa UNIX dan sistem UNIX-like. Hal ini termasuk dalam FreeBSD, dan contoh-contoh di bawah ini adalah untuk FreeBSD, tetapi jika dokumentasi bisa dipercaya, IPFilter juga bekerja dengan Linux pada kernel 2.4 (beberapa Hackery kernel diperlukan). Bawah FreeBSD 5.x, IPFilter bekerja di luar kotak setelah memungkinkan di /etc/rc.conf, tetapi di bawah FreeBSD 4.x, Anda perlu mengkompilasi sebuah kernel untuk dapat menggunakan IPFilter. Dalam teori, itu juga tersedia sebagai modul kernel loadable, tapi itu tidak bekerja (lihat FreeBSD bug 53.966). Lihat bab "Konfigurasi FreeBSD Inti“ di FreeBSD buku pegangan di http://www.freebsd.org/. Kamu perlu untuk tambahkan baris berikut di file konfigurasi kernel:

options IPFILTER

Anda dapat mengaktifkan IP Filter pada FreeBSD dengan menambahkan baris berikut ke file /etc/rc.conf:

ipfilter_enable="YES"

ipfilter_program="/sbin/ipf"

ipfilter_rules="/etc/ipf.rules"

ipv6_ipfilter_rules="/etc/ipf6.rules"

Secara default, IPF dikompilasi untuk memungkinkan semua paket, tetapi tidak akan Anda benci untuk mengetahui bahwa Andainstalasi menggunakan default sebaliknya? Ini akan menjadi sangat buruk karena, tidak seperti filter lain, karya IPF baik pada IPv4 dan IPv6. Begitujika kamu mengunci diri dari Anda mesin,kamu sangat mengunci diri dari Anda mesin. Itu /etc/ipf.rules file yang disebutkan di atas mengandung IPF tersebut menyaring aturan untuk IPv4, sementara /etc/ipf6.rules berisi aturan untuk IPv6. File-file ini tidak Tekanan ini ent secara default, sehingga Anda perlu membuat mereka dan dimasukkan ke dalam beberapa aturan default. Contohnya:

 # allow everything

pass in all

pass out all

FreeBSD Packet Filter

Dalam versi 5.3, FreeBSD memperoleh port dari Filter Packet OpenBSD (PF). PF adalah banyak seperti IPF, tapi memiliki beberapa fungsi yang lebih maju dan, yang lebih penting, dokumentasi yang ekstensif, yang Anda akan menemukan di http://www.openbsd.org/faq/pf/. Anda juga dapat men-download dokumentasi di PDF memformat dari halaman ini, tetapi memperingatkan bahwa teks sering sangat kecil. lain yang menarik Perbedaan antara PF dan filter lainnya adalah bahwa dalam PF IPv4 dan IPv6 terintegrasi: satu Aturan dapat berlaku untuk kedua protokol. Listing 9-8 adalah versi PF dari kami oleh filter sekarang akrab.

- **Listing 9-8. *PF Filter Rules***

```
pass all
unroutable="{ 2001:db8::/32 192.0.2.0/24 }"
block in from $unroutable to any
block out proto tcp from any to any port = 25
block in proto icmp6 from any to ff02::1 icmp6-type echoreq
block return-rst in proto tcp from any to any flags S/SA
```

Seperti yang Anda lihat, aturan PF terlihat sangat mirip dengan IPF aturan. Namun, ada beberapa penting perbedaan. PF memungkinkan aturan yang berlaku baik untuk paket masuk dan keluar, jadi single pass semua menyelesaikan hal yang sama seperti yang lulus dalam semua dan lulus semua. PF juga menambahkan dukungan untuk macro, seperti yang ada di baris kedua. Isi makro harus antara kutipan dan dapat alamat, protokol, port, dan sebagainya. Dalam hal ini, itu daftar, fitur baru lain. Sebuah daftar adalah satu set alamat, protokol, atau port antara kawat opsional dipisahkan dengan koma. Itu Daftar di baris kedua Listing 9-8 berisi IPv6 dan IPv4 dokumentasiprefiks. Itu ketiga blok baris semua paket yang memiliki sumber alamat di salah satu dari dokumentasi prefix oleh referensi makro. Anda dapat memuat aturan filter dari sebuah file (disebut pf.rules dalam kasus ini) dengan perintah pfctl -f pf.rules. Tidak perlu untuk menyiram aturan ditetapkan pertama; ini dilakukan secara otomatis ketika memuat aturan baru. Jika ada kesalahan dalam salah satu aturan, tidak ada aturan yang dimuat dan yang lama tetap berlaku. Anda dapat secara manual menyiram aturan dengan aturan pfctl -F. Anda dapat menampilkan Sekarang daftar aturan dengan pfctl aturan -s, seperti dalam Listing 9-9.

- **Listing 9-9. *Displaying PF Filter Rules***

```
# pfctl -s rules
No ALTQ support in kernel
ALTQ related functions disabled
pass all
block drop in inet6 from 2001:db8::/32 to any
block drop in inet from 192.0.2.0/24 to any
block drop out proto tcp from any to any port = smtp
block drop in inet6 proto ipv6-icmp from any to ff02::1 icmp6-type echoreq
block return-rst in proto tcp all flags S/SA
```

Windows netsh firewall

Windows netsh utilitas yang kita dimanfaatkan dengan baik dalam bab-bab sebelumnya dapat melakukan lebih dari sekedar memanipulasi pengaturan IPv6. Pada Windows XP Service Pack 2, netsh juga berisi "firewall" konteks. Ini mungkin masalah pendapat, tapi aku tidak menemukan netsh yang Firewall konteks yang sangat berguna sebagai tujuan umum packet filter, meskipun memiliki beberapa pilihan untuk memfasilitasi jenis filtering.Namun, firewall Windows XP memiliki fitur yang sangat berguna yang kurang dalam filter dibahas sehingga jauh: memungkinkan penyaringan berdasarkan program yang menerima paket, bukan hanya pada informasi yang ditemukan dalam paket itu sendiri dan informasi seperti antarmuka input / output.

Cisco IPv6 Access Lists

router Cisco menggunakan "akses daftar" untuk paket filter. Selama bertahun-tahun, Cisco telah perlahan-lahan mengubah cara daftar akses bekerja. Awalnya, iOS hanya didukung daftar akses bernomor, dengan nomor yang berbeda berkisar untuk protokol yang berbeda atau jenis daftar akses. Misalnya, nomor daftar akses dari 1 sampai 99 adalah untuk "standar" daftar akses IPv4 yang hanya mendukung penyaringan alamat sumber, sedangkan nomor 100-199 adalah untuk "extended" daftar akses yang juga dapat melihat alamat tujuan, protokol, dan informasi protokol-spesifik seperti nomor port. Dalam IOS 11.2, Cisco memperkenalkan baru sintaks untuk "daftar akses bernama." IPv6 akses daftar sangat mirip dengan daftar akses IPv4 bernama, kecuali bahwa tidak ada lagi perbedaan antara standar dan diperpanjang daftar akses: semua akses IPv6 daftar diperpanjang daftar akses. 3 Perhatikan bahwa ada tidak bisa menjadi IPv4 dan daftar akses IPv6 dengan nama yang sama.

Filtering stateful dengan daftar akses refleksif

Meskipun filter di Listing 9-12 berhasil menjaga keluar banyak lalu lintas yang tidak diinginkan, kita dapat melakukan lebih baik dengan melakukan filtering stateful pada router mana ini adalah tepat, seperti yang bertindak sebagai Customer Premises Equipment (CPE). Cisco IOS memiliki mekanisme yang disebut "akses refleksif daftar "untuk ini, seperti yang ditunjukkan pada Listing 9-13.

- **Listing 9-13. *Stateful Filtering with a Reflexive Access List***

```
!
no ipv6 access-list out-ipv6-acl
no ipv6 access-list in-ipv6-acl
!
ipv6 access-list out-ipv6-acl
permit tcp any any eq 22 reflect state-acl timeout 7500
permit ipv6 any any reflect state-acl
!
ipv6 access-list in-ipv6-acl
!
evaluate state-acl
deny tcp any any log-input
deny udp any any log-input
```

Filtering Services on the Router

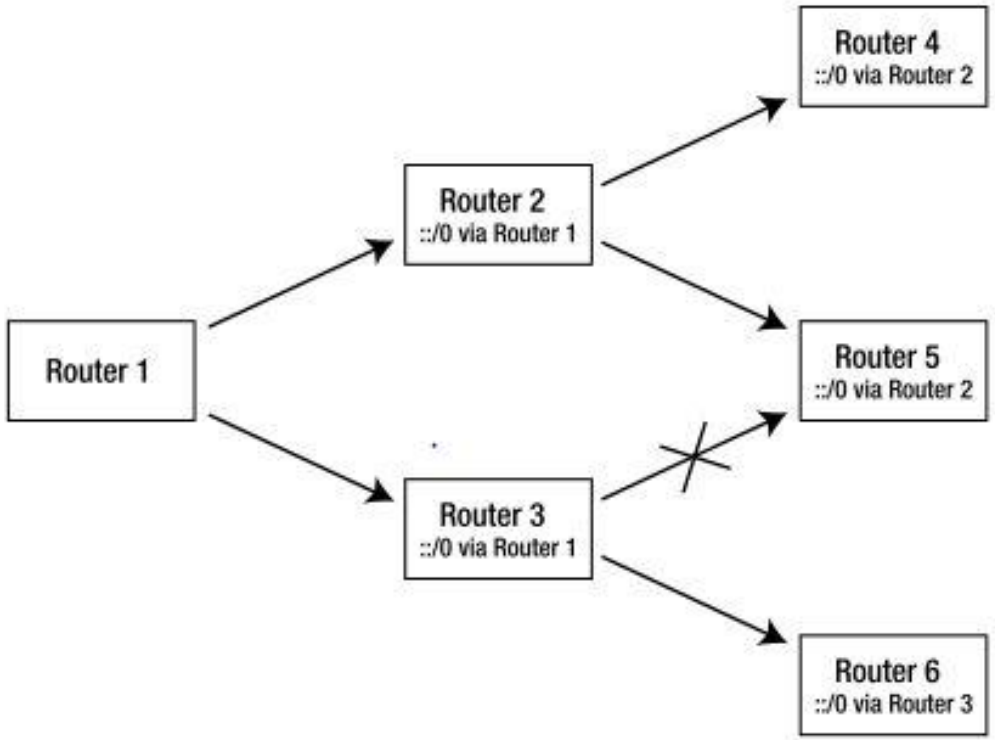
ISP umumnya akan ingin mengizinkan lalu lintas Telnet mengalir melalui jaringan. WalaupunTelnet tidak banyak digunakan seperti dulu, banyak tempat di seluruh Net masih menggunakan protokol, sehingga menyaring keluar grosir akan menjadi langkah yang tidak populer. Di sisi lain, ISP mungkin akan ingin membatasi Telnet akses ke router nya untuk dipercaya alamat hanya untuk mencegah sandi menebak dan untuk menghindari masalah dengan kerentanan dalam Cisco pelaksanaan akses remote,yang telah terjadibeberapa kali di masa lalu. Salah satu cara untuk melakukan hal ini akan menyaringpaket pada port 23 (Telnet yang pelabuhan) di daftar akses pada semua interface, tapi ini sangatmembosankan. IOS memilikicara yang lebih baik untuk mencapai hasil yang sama:nya mungkin untuk menerapkan daftar akses ke sebagian besar protocol atau layanan yang berjalan pada router, tanpa mempengaruhi lalu lintas yang mengalir melalui router. Listing 9-14 mengimplementasikan filter untuk akses jarak jauh ke baris perintah dengan Telnet.

- ***Filtering IPv6 Telnet Access to a Cisco Router***

```
!
ipv6 access-list manage-ipv6
permit ipv6 2001:db8:31::/48 any
!
line vty 0 4
ipv6 access-class manage-ipv6 in
transport input telnet
!
```

Unicast Reverse Path Forwarding

Dalam BCP 38 / RFC 2827, IETF merekomendasikan bahwa ISP hanya menerima paket dari pelanggan mereka dengan alamat sumber yang sebenarnya ditugaskan untuk para pelanggan. Hal ini tidak mungkin untuk mengirimkan paket IP kasar dengan alamat sumber dipalsukan. Memaksa penyerang untuk menggunakan real mereka Alamat IP membuatnya lebih mudah untuk menyaring paket yang bersangkutan dan melacak kembali sumber. Menerapkan filter seperti umumnya de yang baik setiap kali hilir host tidak di bawah Anda control atau memiliki lebih tinggi daripada rata-rata risiko dari yang dikompromikan. Namun, mempertahankan akses daftar pada semua router seperti interface adalah banyak pekerjaan. Ini Di sinilah Cisco unicast Reverse jalanekspedisi (URPF) fitur sangat berguna.



Unicast RPF menggunakan mekanisme yang sama untuk memeriksa paket unicast: pada sebuah antarmuka dengan uRPF diaktifkan, hanya paket dengan alamat sumber yang dicapai lebih antarmuka yang diterima. Pada intinya, uRPF menggunakan tabel routing sebagai filter. Ini memiliki dua keuntungan penting: ada tidak perlu untuk mempertahankan filter secara manual, dan penyaringan lebih efisien karena rute tabel dapat dicari dengan sangat algoritma efisien sementara daftar akses garis dievaluasi satu per satu dari atas ke bawah sampai ada adalah pertandingan.

Filter Limitations

Ada dua keterbatasan penting untuk setiap filter yang bekerja pada basis per-paket dengan IPv6. Yang pertama adalah bahwa, seperti di IPv4, ketika sebuah paket terfragmentasi, TCP atau UDP nomor port yang hadir hanya dalam fragmen pertama. Ketika memblokir paket berdasarkan nomor port, ini bukan banyak masalah: fragmen dengan nomor port di dalamnya akan diblokir, meskipun semua fragmen lain diperbolehkan melalui. (Ini tergantung pada tindakan dikonfigurasi untuk fragmen tajuk atau tindakan default filter, tentu saja.) Tanpa fragmen awal yang memegang TCP atau UDP header, itu tidak mungkin untuk merakit paket asli, jadi pada dasarnya, paket ini diblokir. Namun, jika Anda ingin mengizinkan lalu lintas TCP atau UDP berdasarkan nomor port, Anda juga perlu untuk memungkinkan semua fragmen, karena mereka bisa menjadi bagian dari TCP diperbolehkan atau paket UDP. Dalam keadaan normal, Anda tidak akan melihat paket TCP terfragmentasi di IPv6: karena fragmentasi dilakukan pada sumbernya, mengirimkan paket TCP yang lebih kecil sama mudah. protokol lain mungkin masih terfragmentasi ketika aplikasi atau lapisan atas protocol Penggunaan paket yang terlalu besar untuk (jalan) MTU. Untuk menghindari interaksi menyenangkan antara fragmentasi dan penyaringan,beberapa firewalldan filter (opsional) berkumpul kembali paket sebelumlewat melaluiitu mesin filter.

IPsec

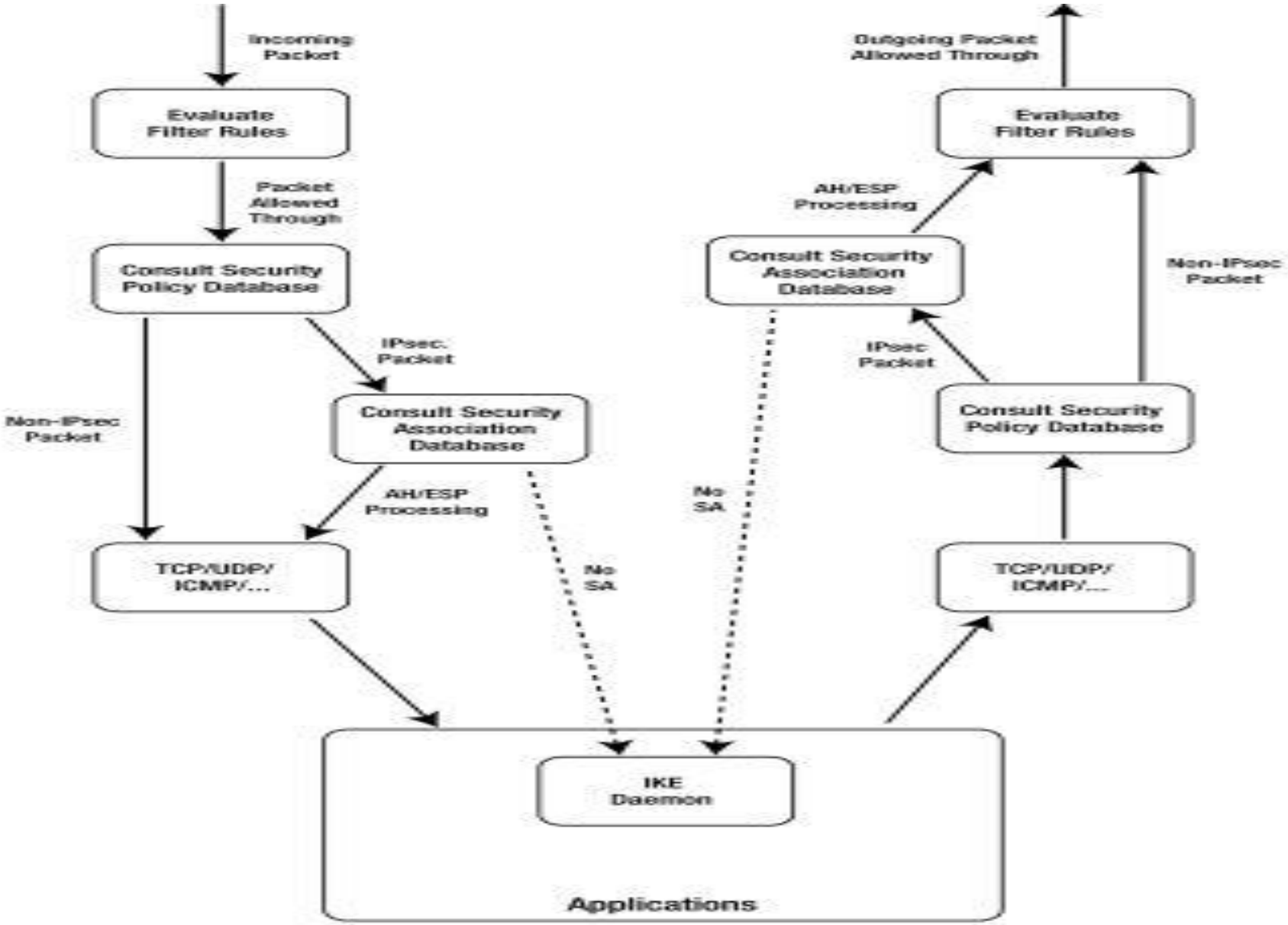
IPsec merupakan kumpulan mekanisme untuk melindungi lalu lintas IP dari penyadapan, modifikasi transit, dan banyak lagi. The "sec" bagian, yang harus ditulis dalam huruf kecil, singkatan dari "keamanan,“ tetapi menggunakan "keamanan IP" dengan mudah menyebabkan kebingungan dengan Option IPv4 Security, yang merupakan sesuatu yang sangatberbeda:Internet tua Protokol Keamanan Pilihan membawa informasi tentang keamanan tingkat data payload dalam pilihan IP. Ini memungkinkan untuk memastikan yang diklasifikasikan Data tidak bocor ke bagian non-diklasifikasikan dari jaringan.

IPsec Headers, Modes, and Algorithms

Ada dua header IPsec: Authentication Header (AH), yang (kejutan) menyediakan otentikasi, dan Payload Encapsulating Security (ESP) header, yang menyediakan baik otentikasi atau enkripsi, atau keduanya. Perbedaan antara AH dan otentikasi-satunya ESP adalah bahwa AH juga melindungi sebagian besar bidang di header IP, sementara ESP hanya dapat melindungi headerdan data sebagai berikutheader ESP.Kedua AH dan ESP adalahditerapkan di salah satu dari dua mode:mengangkutModus atau mode tunnel.Dimengangkutmode,AH atau ESP Header duduk antaraIPHeaderdan transportasiprotokolheader.Dimode tunnel,AH atau ESP mendahului sundulanituasliIP header,dan header IP baru diletakkan di depanAH atau ESP sundulan.IPsecmengangkutmodememungkinkan untuk mengimplementasikan IPsecdalam "keamanangerbang "agakdari dalam sumberatauhost tujuan itu sendiri. Pengaturan umum adalah salah satu di manadua keamanan sepert igateway menerapkan virtua lPribadi Jaringan(VPN) di atas Internet. Kapan host di satu lokasi ingin untuk berkomunikasi dengan host di lokasi lain, VPN gerbang di lokasi pertama menambahkan header ESP (AH tidak digunakan banyak) bersama dengan header IP baru dengan sumber alamat termasuk ke gateway itu sendiri dan alamat tujuan milik remote pintu gerbang.

Exchanging Keys and Security Associations

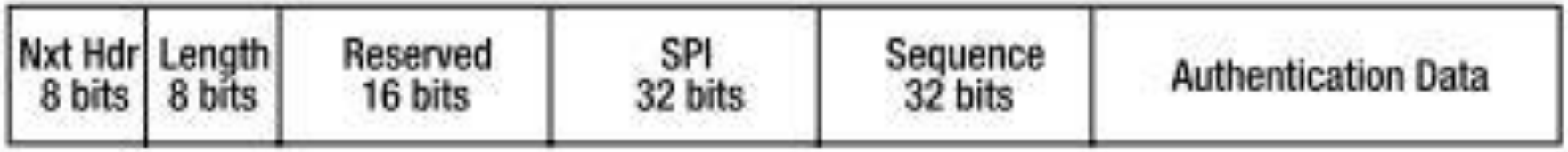
sebelumnya adalah banyak informasi untuk dimasukkan ke dalam satu set file konfigurasi. Tapi yg menentukan nyata adalah bahwa semua pengaturan ini harus sama pada keduapengirim dan sisi penerima untuk IPsec untuk kerja. Internet Key Exchange (IKE) protokol memungkinkan untuk menegosiasikan sebagian besar pengaturan antara dua host yang menerapkan IPsec. IKEsendiri dijahit bersama-sama dari beberapa bagian, termasuk Asosiasi Internet Security dan KeyManagement Protocol (ISAKMP) dan bagian dari Oakley Penentuan Key Protocol. IKE bekerjadalam dua tahap. Selama fase 1, IKE memeriksa identitas koresponden dan melakukan negosiasi saluran aman sehingga lebih komunikasi IKE bisa dienkripsi. Kemudian pada tahap 2,protokol negosiasi Security Asosiasi (DS) yang digunakan untuk melindungi paket dari aplikasi lain.


Incoming Packet
Evaluate Filter Rules
Packet Allowed Through
Consult Security Policy Database
IPsec. Packet
Non-IPsec Packet
Consult Security Association Database
AH/ESP Processing
No SA
TCP/UDP/ ICMP/...
IKE Daemon
Applications
Outgoing Packet Allowed Through
Evaluate Filter Rules
AH/ESP Processing
Consult Security Association Database
IPsec Packet
Non-IPsec Packet
Consult Security Policy Database
TCP/UDP/ ICMP/...

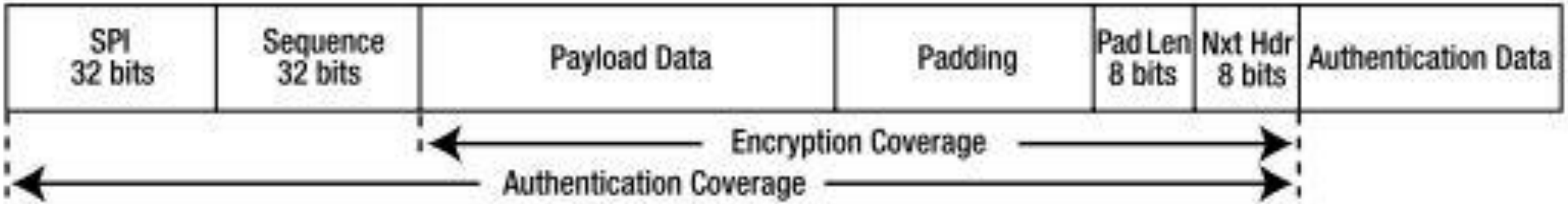
IPsec on the Wire

Ketika IPsec AH atau ESP paket mengalir melalui jaringan, mereka membawa satu dari header ditampilkan diGambar 9-3. Header AH hanya disisipkan di antara header IP dan tingkat yang lebih tinggi (TCP atauUDP) Header bersama dengan data pengguna. Untuk header ESP, hal mendapatkan sedikit lebih kompleks: yangheader-tingkat yang lebih tinggi dan pengguna data dimasukkan ke dalam bidang "payload data" dari header ESP. Jika ada enkripsi ESP, bidang ini dienkripsi, bersama dengan padding yang mengisi data payload untuk bahkan blok yang dibutuhkan oleh algoritma enkripsi, bersama dengan panjang padding dan selanjutnya field header. "Header Next" tidak benar-benar istilah yang tepat dalam hal ini, sebagai header "berikutnya“ (Isi dari field data payload) mendahului kolom header berikutnya. Ketika kedua enkripsi dan otentikasi diaktifkan di ESP, paket pertama dienkripsi, dan hanya kemudian adalah data otentikasi dihitung. Hal ini memungkinkan penerima untuk menolak paket palsu tanpa harus (mencoba) mendekripsi mereka terlebih dahulu. Jika otentikasi ESP tidak memungkinkan, "data otentikasi" lapangan tidak menyajikan. Perhatikan bahwa enkripsi tanpa otentikasi tidak aman karena penyerang dapat memodifikasi dienkripsi paket di cara berbahaya bahkan tanpa mengetahui kunci enkripsi. Keamanan Parameter Index (SPI) lapangan digunakan oleh penerima untuk memetakan paket yang dating Asosiasi Keamanan kanan di database SA, mirip dengan fungsi dari nomor port di TCP dan UDP. Bidang nomor urut berisi counter yang dapat digunakan untuk menggagalkan "Serangan replay." Ini adalah serangan dimana penyerang mencatat pertukaran paket yang sah dan kemudian mengirimkan salinan lain dari paket tersebut. Paket pertama untuk SA selalu memiliki nomor urut satu, dan nomor urut bertambah satu untuk setiap paket baru. Jika penerima ingin perlindungan replay, itu hanya menolak paket dengan nomor urut yang lebih rendah dari paket terakhir yang diterima dan lebih tinggi dari paket berikutnya mengharapkan, dengan margin kecil untuk memperhitungkan paket yang datang dalam rusak.

Authentication Header



Encapsulating Security Payload Header



The KAME IPsec Implementation

Bahkan lebih dari dengan mata pelajaran lain yang dibahas dalam buku ini, IPsec menyediakan banyak cara untuk menembak diri sendiri di kaki jika Anda tidak tahu apa yang Anda lakukan. Contoh di bawah ini hanyalah dimaksudkan untuk menunjukkan bahwa itu sebenarnya mungkin untuk menjalankan IPsec di BSD / Linux dalam praktek, sesuatu yang sulit untuk percaya pada awalnya karena kompleksitas dari protokol dan banyak, banyak cara di mana racoon IKE daemon dapat gagal jika hal-hal yang tidak diatur tepat. IPsec telah menjadi bagian dari upaya KAME dari awal. Mac OS datang dengan KAME IPsec built in, dan untuk FreeBSD, itu hanya sebuah kernel kompilasi pergi. Lihat FreeBSD pengguna sebagai disebutkan sebelumnya bab ini dan tambahkan baris berikut untuk konfigurasi kernel Anda:

options IPSEC

options IPSEC_ESP

IPsec Advantages and Limitations

Sejauh ini, IPsec belum dikerahkan sebagai enkripsi tujuan umum, end-to-end dan otentikasi mekanisme SSL cara yang sama memiliki. Tu terlalu buruk, karena Ipsec memiliki cukup keamanan keuntungan lebih SSL: SSL berjalan di atas TCP, sehingga mengganggguTCP sesi cukup untuk mengganggu itu komunikasi SSL terjadi lebih TCP sesi yang bersangkutan juga.PemecahanTCP sesi sepenuhnya sepele bagi penyerang yang dapat mengamati TCP lalu lintas dan sering bisa dilakukan dengan beberapa upaya untuk TCP berumur panjang sesi bahkan tanpa kemampuan untuk mengamati traffic . sesi SSL hanya blok komunikasi: itu tidak mengizinkan penyerang untuk menyuntikkan dipalsukan Data berhasil atau mendekripsi terenkripsi informasi.

IPsec, di sisi lain, dapat menolak paket mengganggu sebelum TCP, UDP, atau lainnya protokol layer yang lebih tinggi terlihat pada mereka. Ini juga tidak hanya terbatas pada TCP atau transportasi protokol yangmirip dengan TCP. Seorang penyerang yang tidak bisa mengamati lalu lintas bahkan tidak bisa membuat limbah tuan rumah menerima waktu mengeksekusi algoritma otentikasi karena paket-paket yang dipalsukan yang ditolak berdasarkan SPI dan memutar ulang kontra. Ini merupakan keuntungan penting atas mekanisme perlindungan BGP TCP MD5, yang rentan untuk "kripto DoS, “ dimana penyerang menyuntikkan paket palsu yang gagal otentikasi Tes untuk tujuan mengkonsumsi sumber daya CPU pada sistem korban. Namun, sebuah penting downside Ipsec adalah bahwa hal itu perlu membawaenkripsi dan rumah tangga otentikasi informasi di setiap paket individu, yang menambahkan sejumlah besar overheadNya juga umumnya lebih efisien untuk mengotentikasi atau mengenkripsi / mendekripsi besar jumlah data dalam satu pergi (seperti dengan SSL ketika mentransfer blok besar data) bukan daripada melakukan hal yang sama untuk sejumlah individu paket membawa jumlah data yang sama.