BAB 10 - TROUBLESHOOTING

Dalam penggunaan sehari-hari pengguna tidak dapat membedakan antara ipv4 dan ipv6. Untuk mengetahui kondisi tersebut kita perlu mengetahui aplikasi apa saja yang menggunakan ipv4 dan ipv6, bagaimana kondisi protokolnya (ada aplikasi yang tidak dapat dijalankan pada ip versi yang tidak sesuai dengan tipenya). Pada bagian awal bab ini menjelaskan mengenai tcpdump, yaitu alat yang populer untuk memeriksa paket yang mengalir melalui jaringan. Tcpdump akan memberitahu anda ip versi berapa yang sedang digunakan.

Tcpdump

Tcpdump adalah alat (tool) yang digunakan untuk menganalisa jaringan computer. Versi terbaru tcpdump untuk linux didapatkan secara gratis di http://www.tcpdump.org/, untuk mengetahui versi mana yang sedang digunakan maka dapat menggunakan syntak tcpdump-H. Sedangkan untuk os windows bernama "WinDump " tersedia di http://www.winpcap.org/windump/. Aplikasi tcpdump menyediakan user interface dengan dua mekanisme, yaitu dengan penyadapan dan menampilkan paket: Library Packet Capture (library pcap atau libpcap) dan Berkeley Packet Filter (BPF). Pada system BPF, interface jaringan beroperasi bersamaan dengan program pemfilteran, kemudian aplikasi tcpdump akan menerima Salinan dari paket yang mengalir melalui interface jaringan tertentu yang cocok dengan pemfilteran tersebut. Kode program filter ini menggunakan format kode assembly karena alasan efisiensi pada virtual CPU. Library pcap menyediakan interface tingkat tinggi namun pemogramannya menggunakan standar Bahasa C.

Tcpdump ICMPv6

Jika tidak ingin melakukan tcpdumping berulang kali namun anda berkebutuhan untuk selalu terkoneksi dengan system tcpdumping, maka dapat menggunakan fungsi ICMP6 yang berfungsi untuk memantau pencegahan pesan advertisement pada router.

Listing Program 10-1. *Intercepting a Router Advertisement Message*# tcpdump
tcpdump: listening on eth0
13:33:33.436664 fe80::204:27ff:fefe:249f > ff02::1: icmp6: router advertisement➥
[class 0xe0]
1 packets received by filter
0 packets dropped by kernel

Pada Listing Program 10-1 menggunakan system Linux Red Hat, dimana tcpdump dijalankan sebagai root, dan kondisinya user biasa tidak dapat membuka perangkat BPF hampir pada

keseluruhan system. Tcpdump secara otomatis memilih interface yang diinginkan: eth0. Daftar kemungkinan interface dapat ditampilkan dengan sintak –D.

Setiap baris tcpdump dimulai dengan timestamp, dengan format jam, menit, dan detik dengan diikuti oleh presisi mikrodetik. Perlu diperhatikan bahwa perangkat BPF menambahkan informasi pada timestamp ini, sehingga data pada timestamp ini tidak sepenuhnya akurat: bisa jadi terdapat delay saat paket tersebut telah diterima oleh hardware dan saat BPF memprosesnya.

Selanjutnyayang ditampilkan adalah alamat sumber dan alamat tujuan. Alamat sumber adalah alamat link-lokal dari router yang mengirimkan router advertisement, dan alamat tujuan adalah alamat multicast dari semua node. Selanjutnya tcpdump menyimpulkan bahwa paket ICMPv6 berisi router advertisement. Garis berakhir dengan isi bidang Kelas Traffic, karena mengandung nilai nondefault. Ternyata, Cisco memutuskan untuk menghormati RFC 791 (meskipun RFC khusus membahas tentang IPv4) dan menggunakan jenis layanan nilai yang menunjukkan "control internetwork " untuk router advertisement.

" packets received by filter " dan " packets dropped by kernel " berbeda pada masing-masing sistem, tetapi dalam banyak kasus jumlah paket pada library pcap dibaca dari perangkat BPF, sedangkan jumlah paket pada perangkat BPF tidak bisa memberikan nilai ke library pcap karena buffer telah full saat sebuah paket baru masuk, ini terjadi ketika aplikasi (tcpdump) tidak memproses paket cukup cepat pada traffic jaringan. Listing Program 10-2 menggunakan sintak –v untuk status aktif.

Listing Program 10-2. Advertisement Decoding Pesan Router

```
# tcpdump -v -s 0
tcpdump: listening on eth0
13:52:05.259531 fe80::204:27ff:fefe:249f > ff02::1: icmp6: router advertisement( ➥
chlim=64, pref=medium, router_ltime=1800, reachable_time=0, retrans_time=0)(src ➥
lladdr: 00:04:27:fe:24:9f)(mtu: mtu=1500)(prefix info: LA valid_ltime=2592000,pr ➥
eferred_ltime=604800,prefix=2001:db8:31:53:/64) [class 0xe0] (len 64, hlim 255)
```

Sintak –v untuk menunjukkan output verbose, -s 0 untuk memberitahu tcpdump agar menangkap paket yang dikirimkan secara keseluruhan.

Listing Program 10-3 menunjukkan interaksi antara dua host.

Listing 10-3. *Decoding a Neighbor Discovery Exchange*

```
# tcpdump -v -s 0 -e
tcpdump: listening on eth0
15:02:27.471601 0:a:95:f5:24:6e 33:33:ff:29:23:b6 ip6 86: host3.example.com > ➥
ff02::1:ff29:23b6: icmp6: neighbor sol: who has host5.example.com(src lladdr: ➥
00:0a:95:f5:24:6e) (len 32, hlim 255)
15:02:27.471708 0:1:2:29:23:b6 0:a:95:f5:24:6e ip6 86: host5.example.com > host3 ➥
.example.com: icmp6: neighbor adv: tgt is host5.example.com(SO)(tgt lladdr: 00:0 ➥
```

1:02:29:23:b6) (len 32, hlim 255)

Syntak -e memberitahu tcpdump untuk menampilkan informasi link-layer.

Tcpdumping UDP

Listing program 10-4 sampai 10-7 menunjukkan output yang berbeda dari tcpdump saat menangkap request DNS dan balasan dari UDP.

Listing Program 10-4. *Standard* tcpdump *Output*

```
# tcpdump
tcpdump: listening on eth0
13:12:33.935061 host5.example.com.32782 > ns.example.com.domain: 15025+ AAAA? ➥
ns.example.com. (32)
13:12:33.948362 host5.example.com.domain > host5.example.com.32782: 15025* 1/2/2 ➥
(148)
```

Untuk paket DNS seperti ini, tcpdump menampilkan request identifier, jenis query, dan nama yang informasi yang diminta. Nilai terakhir pada baris adalah panjang dari paket DNS, termasuk panjang IP dan header UDP.

Listing Program 10-5 menampilkan informasi selanjutnya.

Listing Program 10-5.*More Verbose* tcpdump *Output and Capturing Full Packets*

```
# tcpdump -v -s 0
13:06:07.693110 host5.example.com.32775 > ns.example.com.domain: [udp sum ok] ➥
65043+ AAAA? ns.example.com. (32) (len 40, hlim 64)
13:06:07.710238 ns.example.com.domain > host5.example.com.32775: [udp sum ok] ➥
65043* 1/2/2 ns.example.com. AAAA 2001:db8:31:53::53 (148) (len 156, hlim 60).
```

Dengan ekstra verbositas dan seluruh paket yang tersedia, tcpdump ganda memeriksa checksum UDP dan memberitahu pengguna apakah checksum dalam paket itu benar. Dengan DNS yang lengkap tersedia, maka bagian jawaban juga ditampilkan. Listing Program 10-6 menggunakan pengaturan verbosity yang lebih tinggi.

Listing 10-6. *Even More Verbose* tcpdump *Output When Capturing Full Packets*

```
# tcpdump -vv -s 0
tcpdump: listening on eth0
13:07:17.811560 host5.example.com.32778 > ns.example.com.domain: [udp sum ok] ➥
45697+ AAAA? ns.example.com. (32) (len 40, hlim 64)
13:07:17.827372 ns.example.com.domain > host5.example.com.32778: [udp sum ok] ➥
45697* q: AAAA? ns.example.com. 1/2/2 ns.example.com. AAAA 2001:db8:31:53::53 ns: ➥
example.com. NS ns.example.com., example.com. NS ns2.beispiel.de. ar: ➥
ns.example.com. A 192.0.2.80, ns.example.com. A6 0 2001:db8:31:53::53 ➥
```

(148) (len 156, hlim 60)

Listing Program 10-7 menggunakan sintak –X untuk mendapatkan paket hexdump.

Listing 10-7. *Displaying Packet Contents in a Hexdump with* tcpdump

```
# tcpdump -X -s 0
11:24:03.310136 host5.example.com.64875 > ns.example.com.domain: 29533+ AAAA? ➥
ns.example.com. (32)
0x0000: 6000 0000 0028 1140 2001 0db8 0002 0000 `....(.@.......
0x0010: 020a 95ff fef5 246e 2001 0db8 0031 0053 ......$n.....1.S
0x0020: 0000 0000 0000 0053 fd6b 0035 0028 938e .......S.k.5.(..
0x0030: 735d 0100 0001 0000 0000 0000 026e 7307 s].........ns.
0x0040: 6578 616d 706c 6503 636f 6d00 001c 0001 example.com.....
11:24:03.327441 ns.example.com.domain > host5.example.com.64875: 29533* 1/2/2 ➥
AAAA 2001:db8:31:53::53 (148)
0x0000: 6000 0000 009c 113f 2001 0db8 0031 0053 `......?.....1.S
0x0010: 0000 0000 0000 0053 2001 0db8 0002 0000 .......S........
0x0020: 020a 95ff fef5 246e 0035 fd6b 009c 7f6e ......$n.5.k...n
0x0030: 735d 8580 0001 0001 0002 0002 026e 7307 s].........ns.
0x0040: 6578 616d 706c 6503 636f 6d00 001c 0001 example.com.....
0x0050: c00c 001c 0001 0001 5180 0010 2001 0db8 ........Q.......
0x0060: 0031 0053 0000 0000 0000 0053 c00f 0002 .1.S.......S....
0x0070: 0001 0001 5180 0002 c00c c00f 0002 0001 ....Q...........
0x0080: 0001 5180 0011 036e 7332 0862 6569 7370 ..Q....ns2.beisp
0x0090: 6965 6c02 6465 00c0 0c00 0100 0100 0151 iel.de.........Q
0x00a0: 8000 04c0 0002 50c0 0c00 2600 0100 0151 ......P...&....Q
0x00b0: 8000 1100 2001 0db8 0031 0053 0000 0000 .........1.S....
0x00c0: 0000 0053 ...S
```

tcpdumping TCP

Output dari tcpdump sedikit sulit untuk diuraikan ketika menangkap paket TCP, hanya karena TCP merupakan protokol yang lebih kompleks dari pada UDP atau ICMPv6. Listing 10-8 menunjukkan permintaan DNS sama seperti Daftar 10-4 sampai 10-7 tetapi sekarang melalui TCP, dengan tidak ada flag yang berlaku untuk mengubah keluaran tcpdump itu.

Listing 10-8. tcpdump *of a DNS Request/Reply over TCP*

```
% sudo tcpdump
tcpdump: WARNING: en0: no IPv4 address assigned
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on en0, link-type EN10MB (Ethernet), capture size 96 bytes
14:32:35.468540 host3.example.com.58231 > ns.example.com.domain: S 2265400865: ➥
2265400865(0) win 65535 <mss 1440,nop,wscale 0,nop,nop,timestamp 631301731 0>
14:32:35.484974 ns.example.com.domain > host3.example.com.58231: S 3739752857: ➥
3739752857(0) ack 2265400866 win 57344 <mss 1220> [flowlabel 0x6c66e]
```

14:32:35.485197 host3.example.com.58231 > ns.example.com.domain: . ack 1 win 65535
14:32:35.485722 host3.example.com.58231 > ns.example.com.domain: P 1:35(34) ack 1 ➥
win 65535 45278+[|domain]
14:32:35.503456 ns.example.com.domain > host3.example.com.58231: P 1:151(150) ack ➥
35 win 58560 45278*[|domain] [flowlabel 0x6c6bd]
14:32:35.507729 host3.example.com.58231 > ns.example.com.domain: F 35:35(0) ack ➥
151 win 65535

Dalam contoh ini, tcpdump berjalan di bawah MacOS (jadi kita perlu sudo) pada interface saat IPv6-only. Program ini mengingatkan kita bahwa interface tidak memiliki alamat IPv4, tapi ini tidak ada hubungannya dengan operasinya. Karena tcpdump berisi protokol logika decode sendiri, tidak relevan, karena semua rasa Ethernet menggunakan link-layer header format yang sama. untuk interface keluar. Pilihan MSS memungkinkan TCP untuk menemukan MTU yang lebih kecil di ujung jauh sangat efisien, meskipun Path MTU Discovery masih diperlukan untuk menemukan mengurangi MTU.

NOP ("tidak ada operasi") opsi yang diperlukan untuk mengisi ruang pilihan dalam header TCP bahkan 32 bit. Skala jendela (wscale) dan opsi timestamp adalah bagian dari RFC 1323 TCP ekstensi untuk kinerja tinggi. Flag umum adalah P untuk PUSH dan F untuk FIN. Paket TCP dengan ada bendera khusus yang mendapatkan periode di tempat ini. Server menggunakan flow label untuk sesi ini, tapi klien tidak, dan tcpdump tidak menampilkan 0x0 nilai flow label dalam kasus itu. Ketiga paket pengakuan kembali dari klien ke server, menyelesaikan setup sesi. Tak lama setelah mendirikan sesi, klien mengirimkan pertama sebenarnya paket data: query DNS, yang adalah 34 byte panjang. Domain pesan menunjukkan bahwa decoding lanjut tidak mungkin di suatu tempat selama pengolahan protokol DNS. Hal yang sama terjadi untuk jawaban, yang bahkan lebih lama pada 150 byte. Jawabannya mengakui segmen yang berisi permintaan dengan menunjukkan bahwa byte berikutnya yang diharapkan dalam sesi adalah nomor 35 (relatif terhadap jumlah urutan awal). Setelah klien mengirimkan paket FIN dalam rangka untuk meruntuhkan sesi, ada beberapa

TCP HIGH PERFORMANCE EXTENSIONS

Pada 100 Mbps Ethernet, mungkin untuk mengirimkan paket ukuran penuh setiap 6,7 mikrodetik. Dengan TCP maksimum ukuran jendela dari 65.535 bytes, pengirim harus menghentikan pengiriman data setelah 43 paket (288 mikrodetik) dan menunggu penerima untuk mengakui paket pertama sebelum mengirim satu ke-44. Namun, pada link transatlantik, dibutuhkan sekitar 80 milidetik untuk paket pengakuan pertama yang kembali karena kecepatan cahaya penundaan. Jadi untuk 79,712 milidetik dari 80, pengirim hanya menunggu pengakuan tanpa mengirim
data apapun, sangat membatasi kinerja TCP. Bandwidth maksimum yang dapat menggunakan TCP merupakan salah satu ukuran jendela per round trip, atau sekitar 800 kilobyte per detik dalam contoh ini.

untuk mengaktifkan fitur ini. Sebuah Pilihan skala window nol berarti bahwa sistem pengiriman mendukung pilihan jika ujung yang lain keinginan untuk menggunakannya, tapi tidak akan menggunakannya sendiri untuk paket keluar selama sesi ini. Opsi timestamp memungkinkan untuk estimasi waktu pulang pergi yang jauh lebih baik, yang diperlukan untuk mencapai kinerja tinggi. Rata-rata ukuran paket di Internet adalah sekitar 500 byte, sehingga memiliki biaya overhead 12 byte ekstra dalam setiap paket mengurangi efisiensi bandwidth

sebesar 2%. Ini tambahan overhead adalah harga kecil untuk membayar jika itu berarti mampu menggunakan bandwidth yang tersedia penuh, tapi sayangnya, opsi timestamp digunakan dalam setiap paket, terlepas dari apakah pilihan skala jendela benar-benar diaktifkan (yaitu, ada yang sebenarnya skala jendela dua atau lebih tinggi).

Namun, sistem umumnya menggunakan "satu ukuran cocok untuk semua" default, dan sangat sedikit aplikasi yang ditetapkan mereka ukuran penyangga sendiri. Jadi sebagian besar waktu, dukungan RFC 1323 hanya limbah 12 byte per paket TCP. Anda dapat menonaktifkan kedua opsi skala jendela dan opsi timestamp pada FreeBSD dan sistem MacOS dengan variabel sysctl net.inet.tcp.rfc1323 ke nol. Linux menggunakan dua variabel sysctl untuk ini:

- net.ipv4.tcp_window_scaling dan net.ipv4.tcp_timestamps.
- Pengaturan mereka ke nol untuk berubah off perilaku, baik untuk IPv4 dan IPv6.
- Beberapa filter stateful melakukan pemeriksaan pada nomor urut TCP, tetapi tidak mendukung ekstensi RFC 1323. jelas, ini menyebabkan masalah ketika pilihan skala jendela 2 atau lebih tinggi yang berlaku. Misalnya, IPF menderita masalah ini. Lihat Bab 9 untuk informasi lebih lanjut tentang IPF.
-

Promiscuity

Secara default, tcpdump akan mencoba untuk menempatkan interface kedalam mode "promiscuous," sehingga menarik semua paket dari kawat, bukan hanya yang ditujukan pada alamat MAC sendiri, bersama dengan siaran biasa dan multicast yang menarik. Modus promiscuous, tentu saja, hanya berlaku untuk interface yang menggunakan alamat MAC, seperti Ethernet. Dengan proliferasi Ethernet switch, modus ini tidak berguna seperti dulu:

Filters

Sejauh ini, kami telah menjalankan tcpdump tanpa memberikan filter, sehingga ditampilkan semua paket yang melewati interface yang bersangkutan. Dalam kebanyakan kasus, bukan yang Anda inginkan, jika hanya karena informasi yang berlebihan membuat lebih sulit untuk menguraikan program output untuk paket yang relevan. Pcap library dan, dengan perluasan, program tcpdump tidak menggunakan sintaks penyaringan tetap. Filter sederhana hanya melihat alamat atau bidang-bidang seperti nomor port, mungkin disertai dengan identifier protokol. Filter lebih kompleks diciptakan dengan merangkai fragmen penyaring kecil dengan AND dan OR klausa dan tanda kurung. Contoh filter yang sederhana adalah sebagai berikut:
ip looks for IPv4 packets.
ip6 looks for IPv6 packets.
host 192.0.2.53 looks for packets with IPv4 address 192.0.2.53. This includes IPv4 packets to and from this address and also ARP packets for this address.
host 2001:db8:31:53::53 matches any packet to or from this IPv6 address.
host ns.example.com looks up the domain name ns.example.com and matches all the IP addresses that the DNS returns (both IPv4 and IPv6).
ip6 host ns.example.com looks up the domain name ns.example.com and matches all the IPv6 addresses that the DNS returns.
net 2002::/16 matches any packet to or from prefix 2002::/16 (6to4 address space).2
src net fe80::/16 matches packets with a link-local source address.
dst ff02::1 matches packets addressed to the all-hosts multicast address.
port 53 matches all TCP and UDP packets with port number 53 (DNS).
dst port 80 matches all TCP and UDP packets with a destination port 80.

tcp matches TCP packets.
udp matches UDP packets.
icmp6 matches ICMPv6 packets. **ether host 0:3:93:e0:ea:2** looks for packets with the specified MAC address.
ether proto 0x86dd matches packets with ethertype 0x86dd (IPv6). The ethertype may also be supplied in decimal.
ip6 proto 58 matches IPv6 packets with a Next Header value of 58 (ICMPv6). You can also use protocol names present in /etc/protocols, but they must be escaped with a backslash if they're also tcpdump filter keywords.
ip6 protochain ipv6-icmp matches ICMPv6 packets with possible intermediate headers between the IPv6 and ICMPv6 headers. ("ipv6-icmp" is the name for ICMPv6 in /etc/protocols.)
"protochain \tcp" looks for IPv4 or IPv6 packets with a TCP payload, possibly at the end of a protocol chain. The backslash escapes the TCP keyword, so the entry named "tcp" in the file /etc/protocols provides the required protocol number. Without the slash, tcpdump generates a parse error. The quotation marks are necessary to keep the shell from interpreting the backslash
Anda dapat membuat filter lebih kompleks dengan menggabungkan beberapa klausa dengan "dan," "atau," dan "tidak." Jangan lupa untuk menggunakan tanda kurung bila diperlukan, dan gunakan tanda kutip untuk menjaga shell dari menafsirkan kurung dalam kasus ini. Dan jika ini tidak memenuhi kebutuhan Anda penyaringan, kita lihat pada halaman manual tcpdump untuk informasi tentang cara membuat filter lebih canggih.

IPv6 Connectivity

Seperti yang kita lihat dalam Bab 8, hal-hal aneh bisa terjadi ketika host memiliki alamat IPv6 tapi bukan rute default, atau rute default IPv6 tapi tidak ada alamat unicast global. Anda dapat mengetahui apakah sistem memiliki rute default IPv6 dengan daftar seluruh tabel routing IPv6 dengan netstat-r-A inet6 di Linux atau netstat-r-f inet6 bawah FreeBSD atau MacOS.

Address Availability and DAD Failures

Listing 10-9. *Listing Addresses Under Windows,Uncovering a DAD Failure*
```
netsh interface ipv6>show address
Querying active state...
Interface 5: Local Area Connection 3
```

Listing 10-10 menunjukkan baris dari syslog dan output dari perintah ifconfig di bawah MacOS ketika terjadi kegagalan DAD. FreeBSD output hampir sama.

Listing 10-10. *Syslog and* ifconfig *Output Under MacOS/FreeBSD After a DAD Failure*
```
% tail /var/log/system.log
May 4 17:16:40 localhost kernel: en1: DAD detected duplicate IPv6 address 2001 ➥
:0db8:0031:0002:0204:27ff:fefe:249f: NS in/out=0/1, NA in=1
May 4 17:16:40 localhost kernel: en1: DAD complete for 2001:0db8:0031:0002:0204 ➥
:27ff:fefe:249f - duplicate found
```

May 4 17:16:40 localhost kernel: en1: manual intervention required
% ifconfig en1
en1: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
inet6 2001:db8:31:2:201:2ff:fe29:23b6 prefixlen 64 duplicated
inet6 fe80::230:65ff:fe24:f106 prefixlen 64 scopeid 0x5
ether 00:30:65:24:f1:06
media: autoselect status: active
supported media: autoselect

Pada router cisco, ada kemungkinan terjadinya kegagalan saat proses login seperti terlihat pada listing 10-11

Listing 10-11. *DAD Failure on a Cisco Router*
3w0d: %IPV6-4-DUPLICATE: Duplicate address 2001:DB8:31:2:204:27FF:FEFE:249F on ➥ Ethernet0
Cisco#show ipv6 interface ethernet 0
Ethernet0 is up, line protocol is up
IPv6 is enabled, link-local address is FE80::204:27FF:FEFE:249F
Global unicast address(es):
2001:DB8:31:2:204:27FF:FEFE:249F, subnet is 2001:DB8:31:2::/64 [EUI/DUP]
3FFE:FFFF:310:3:204:27FF:FEFE:249F, subnet is 3FFE:FFFF:310:3::/64 [EUI]
Joined group address(es):
FF02::1
FF02::2
FF02::9
FF02::1:FFFE:249F
MTU is 1500 bytes
ICMP error messages limited to one every 100 milliseconds
ICMP redirects are enabled
ND DAD is enabled, number of DAD attempts: 1
ND reachable time is 30000 milliseconds
ND advertised reachable time is 0 milliseconds
ND advertised retransmit interval is 0 milliseconds
ND router advertisements are sent every 200 seconds
ND router advertisements live for 7200 seconds
Hosts use stateless autoconfig for addresses.

Hal tersebut terjadi karena router cisco memiliki 2 alamat unicast global sehingga terjadi crash. Dengan demikian maka alamat unicast pertama tidak dapat digunakan lagi.

Meskipun, tampaknya, di bawah Windows, Linux, dan MacOS, sistem kembali alamat IPv4 ke aplikasi yang nama lookup ketika tidak ada alamat IPv6 global yang, tuan rumah, menggali, dan nslookup masih akan menunjukkan alamat IPv6, sebagai alat tersebut memotong penyelesai perpustakaan sistem dan query DNS server secara langsung.

Ndp

Ndp digunakan untuk menampilkan informasi router yang lain termasuk konfigurasi pada BSD dan sistem MacOS. NDP-a menampilkan daftar tetangga bersama dengan alamat link-layer mereka, mirip dengan arp-a dengan IPv4. Lihat Listing 10-12.

Listing 10-12. *Displaying the Neighbor Cache on FreeBSD or MacOS with* ndp

```
% ndp -an
Neighbor Linklayer Address Netif Expire St Flgs Prbs
::1 (incomplete) lo0 permanent R
2001:1af8:6::20a:95ff:fef5:246e 0:a:95:f5:24:6e en1 permanent R
fe80::1%lo0 (incomplete) lo0 permanent R
fe80::204:27ff:fefe:249f%en1 0:4:27:fe:24:9f en1 23h57m56s S R
fe80::20a:95ff:fef5:246e%en1 0:a:95:f5:24:6e en1 permanent R
```

Seperti biasa, n-bendera menekan pencarian alamat-ke-nama. Menggunakan bendera ini cukup banyak diperlukan bila menggunakan NDP pada MacOS, karena sistem sebaliknya akan daftar nama sendiri setiap kali nama pencarian tidak berhasil. Rupanya, ini adalah hasil dari bug. Listing 10-13 menggunakan NDP untuk daftar informasi antarmuka khusus.

Listing 10-13. *Listing Information for an Interface with* ndp

```
% ndp -i xl0
linkmtu=1500, curhlim=64, basereachable=30s0ms, reachable=40s, retrans=1s0ms
Flags: nud accept_rtadv
```

Traceroute6

Jika sistem memiliki kedua rute default IPv6 dan alamat unicast global yang digunakan, Anda dapat melakukan traceroute6 (tracert6 bawah Windows, dan tracert juga mendukung IPv6) untuk menentukan apakah benar-benar ada konektivitas IPv6. Sebuah traceroute 2002 :: sering merupakan pilihan yang baik jika Anda tidak yakin apakah DNS sudah berfungsi. Karena ini adalah alamat 6to4 untuk alamat IPv4

0.0.0.0, jejak tidak akan menyelesaikan, tetapi akan kios di gateway 6to4 terdekat. Jika Anda melihat setidaknya beberapa hop bekerja di traceroute6 output, router lokal dan yang berikutnya bekerja. Anda bisa, tentu saja, juga melakukan traceroute ke arah host IPv6 yang valid, tapi untuk ini, Anda perlu mengetahui alamat IPv6 host atau DNS harus bekerja. Pada semua sistem, netstat-n akan daftar koneksi TCP yang aktif dan menampilkan berakhir lokal dan remote alamat, sehingga mudah untuk memberitahu versi IP untuk sesi. Lihat Listing 10-14.

Listing 10-14. *Determining a Session's IP Version with* netstat

```
% netstat -n | more
Active Internet connections
Proto Recv-Q Send-Q Local Address Foreign Address (state)
tcp6 0 0 2001:db8:31::20a.55858 3ffe:ffff:2310:2.993 ESTABLISHED
```

tcp4 0 0 192.0.2.6.55672 192.0.2.225.22 ESTABLISHED
tcp6 0 0 2001:db8:31::20a.52731 2001:db8:2:5::2.80 CLOSE_WAIT
udp6 0 0 *.5353 *.*
udp4 0 0 *.5353 *.*

Pipa output melalui lebih memastikan informasi tidak gulir di layar segera. Sesi pertama adalah koneksi TCP melalui IPv6 menuju Aman server IMAP (port 993). Baris kedua adalah

sesi SSH IPv4 (port 22), dan koneksi TCP lalu, yang telah ditutup, adalah sesi HTTP. Dua baris terakhir menunjukkan bahwa sistem mendengarkan untuk paket UDP yang masuk pada port 53533 untuk kedua IPv6 dan IPv4 pada soket terpisah.