

IPv6 for IPv4 users

Bill Stearns, Active Countermeasures

Welcome!

- ▷ **Bill Stearns**
 - Active Countermeasures
- ▷ **Prerequisites**
 - Helpful if you're familiar with IPv4

What we'll cover

- ▷ **IPv6**
 - How it compares to IPv4
 - Addresses
 - Multicast
 - OS/App support

Addresses

- ▷ 128 bits = 16 bytes = 32 hex characters
 - (IPv4: 32 bits = 4 bytes = 8 hex characters)
 - Colons instead of periods
 - fe80::da7f:fd91:ee44:5297
 - Umm, "::"?
 - Replaces single longest block of all 0's
 - = fe80:0000:0000:0000:da7f:fd91:ee44:5297
 - Can leave off leading 0's
 - = fe80:0:0:0:da7f:fd91:ee44:5297

Wireshark IPv4 SSH- packet

No.	Time	Source	Destination	Protocol	Length	Info
4	0.056977	167.71.97.235	10.0.0.111	SSHv2	108	Server: Protocol (SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.10)
5	0.057083	10.0.0.111	167.71.97.235	TCP	66	50997 → 22 [ACK] Seq=1 Ack=43 Win=131712 Len=0 TSval=20

Frame 4: 108 bytes on wire (864 bits), 108 bytes captured (864 bits)

- Ethernet II, Src: ARRISGro_bd:3b:dc (f4:0e:83:bd:3b:dc), Dst: Apple_11:79:bc (40:6c:8f:11:79:bc)
- Internet Protocol Version 4, Src: 167.71.97.235, Dst: 10.0.0.111
 - 0100 = Version: 4
 - 0101 = Header Length: 20 bytes (5)
 - Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
 - Total Length: 94
 - Identification: 0xc7ba (51130)
 - Flags: 0x40, Don't fragment
 - Fragment Offset: 0
 - Time to Live: 50
 - Protocol: TCP (6)
 - Header Checksum: 0x6d3e [validation disabled]
 - [Header checksum status: Unverified]
 - Source Address: 167.71.97.235
 - Destination Address: 10.0.0.111
- Transmission Control Protocol, Src Port: 22, Dst Port: 50997, Seq: 1, Ack: 1, Len: 42
- SSH Protocol

```
0000 40 6c 8f 11 79 bc f4 0e 83 bd 3b dc 08 00 45 00 @l..y... ..;...E.
0010 00 5e c7 ba 40 00 32 06 6d 3e a7 47 61 eb 0a 00 .^...@.2. m>.Ga...
0020 00 6f 00 16 c7 35 f7 44 8c ff 1f d3 4c 0e 80 18 .o...S.D ...L...
0030 00 e3 3f 67 00 00 01 01 08 0a 0f f1 ec d8 7a 90 ..?g.... ..z...
0040 cd 7e 53 53 48 2d 32 2e 30 2d 4f 70 65 6e 53 53 ~SSH-2. 0-OpenSS
0050 48 5f 37 2e 32 70 32 20 55 62 75 6e 74 75 2d 34 H_7.2p2 Ubuntu-4
0060 75 62 75 6e 74 75 32 2e 31 30 0d 0a ubuntu2. 10..
```

Frame (frame), 108 bytes

Packets: 31 · Displayed: 31 (100.0%)

Profile: Default

Wireshark IPv6 SSH- packet

Wireshark packet capture showing an IPv6 SSH packet. The packet list displays three packets:

- 3 0.029062 2601:18c:4202:65f0::... 2604:a880:800:c1::... TCP 86 50978 → 22 [ACK] Seq=1 Ack=1 Win=131360 Len=0 TSval=20...
- 4 0.060612 2604:a880:800:c1::... 2601:18c:4202:65f0::... SSHv2 128 Server: Protocol (SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu...
- 5 0.060772 2601:18c:4202:65f0::... 2604:a880:800:c1::... TCP 86 50978 → 22 [ACK] Seq=1 Ack=43 Win=131328 Len=0 TSval=2...

The packet details for Frame 4 (SSHv2) are expanded, showing:

- Frame 4: 128 bytes on wire (1024 bits), 128 bytes captured (1024 bits)
- Ethernet II, Src: ARRISGro_bd:3b:dc (f4:0e:83:bd:3b:dc), Dst: Apple_11:79:bc (40:6c:8f:11:79:bc)
- Internet Protocol Version 6, Src: 2604:a880:800:c1::221:c001, Dst: 2601:18c:4202:65f0:a079:8c1b:5e3e:491c
 - 0110 = Version: 6
 - 0000 0000 = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
 - 0101 1110 0010 0101 0110 = Flow Label: 0x5e256
 - Payload Length: 74
 - Next Header: TCP (6)
 - Hop Limit: 50
 - Source Address: 2604:a880:800:c1::221:c001
 - Destination Address: 2601:18c:4202:65f0:a079:8c1b:5e3e:491c
- Transmission Control Protocol, Src Port: 22, Dst Port: 50978, Seq: 1, Ack: 1, Len: 42
- SSH Protocol

The packet bytes pane shows the raw data in hexadecimal and ASCII:

```
0000 40 6c 8f 11 79 bc f4 0e 83 bd 3b dc 86 dd 60 05 @l.y...:..`
0010 e2 56 00 4a 06 32 26 04 a8 80 08 00 00 c1 00 00 ·V·J·2&.....
0020 00 00 02 21 c0 01 26 01 01 8c 42 02 65 f0 a0 79 ...!...&...B.e.y
0030 8c 1b 5e 3e 49 1c 00 16 c7 22 88 a6 ba ff 6b c7 ..^>I..."....k.
0040 1f cb 80 18 00 e0 d7 fa 00 00 01 01 08 0a 0f f1 .....
0050 32 56 7a 8d e7 3d 53 53 48 2d 32 2e 30 2d 4f 70 2Vz...=SS H-2.0-Op
0060 65 6e 53 53 48 5f 37 2e 32 70 32 20 55 62 75 6e enSSH_7. 2p2 Ubu
```

Packet capture file: ipv6-ssh.pcap | Packets: 49 · Displayed: 49 (100.0%) | Profile: Default

TCP? UDP? ICMP?

- ▷ **TCP**
 - Same layer as IPv4
- ▷ **UDP**
 - Same layer as IPv4
- ▷ **ICMP**
 - Different types and codes

Ports

- ▷ Colon gets confusing, so
 - [http://\[fe80::da7f:fd91:ee44:5297\]:80](http://[fe80::da7f:fd91:ee44:5297]:80)
 - Easier to use hostname:
 - <http://www.example.com:80>



Private/reserved addresses

- ▷ **IPv4**
 - 10.x.y.z, 192.168.y.z, 172.16.y.z-172.31.y.z
- ▷ **IPv6**
 - fe80::/112
- ▷ **Loopback**
 - IPv4 127.0.0.1 -> IPv6 ::1
- ▷ **Unspecified**
 - IPv4 0.0.0.0 -> IPv6 ::

Can I have both?

▷ **Absolutely!**

- A single network interface can have multiple IPv4 and multiple IPv6 addresses
- Almost always have an fe80:... address to start
- If an IPv6 router shows up, will locally create a global IPv6

Ifconfig eth0 (and lo)

```
lo          Link encap:Local Loopback
            inet addr:127.0.0.1  Mask:255.0.0.0
            inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING  MTU:65536  Metric:1
```

```
eth0       Link encap:Ethernet  HWaddr 5a:42:63:68:45:7f
            inet addr:167.71.106.69  Bcast:167.71.111.255  Mask:255.255.240.0
            inet6 addr: 2604:a880:800:c1::221:b001/64 Scope:Global
            inet6 addr: fe80:: 5842:63ff:fe68:457f/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```

ip addr show dev eth0 (and lo)

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever

2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group
default qlen 1000
    link/ether 5a:42:63:68:45:7f brd ff:ff:ff:ff:ff:ff
    inet 167.71.106.69/20 brd 167.71.111.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet 10.17.0.17/16 brd 10.17.255.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 2604:a880:800:c1::221:b001/64 scope global
        valid_lft forever preferred_lft forever
    inet6 fe80::5842:63ff:fe68:457f/64 scope link
        valid_lft forever preferred_lft forever
```

Types of addresses

- ▷ Run `ifconfig`
 - `::1` Scope:Host
 - Localhost, assigned to `lo`
 - `fe80:...` Scope:Link
 - Only communicates on this network segment
 - (other) Scope:Global
 - Used to talk with the rest of the world
- ▷ Also: `"ip addr"`
- ▷ If you only have `fe80`'s, no Internet

IPv6 allocations

- ▷ Regional Internet Registry: /12 ($\sim 10^{34}$ *)
to /23 ($\sim 10^{32}$)
 - Local IR: /19 (10^{33}) to /32 (10^{29})
- ▷ ISP: /32 (10^{29})
- ▷ Customer: /48 (10^{24}) to /56 (10^{22})
 - Internal networks: /64 (10^{19}) or smaller

* Approximate # of IPv6 addresses in each

https://en.wikipedia.org/wiki/IPv6_address#General_allocation

Broadcast/multicast

▷ IPv4

- 255.255.255.255
- *Net.work*.255.255 (Network + all 1's)
- 224.x.y.z-239.x.y.z Multicast

▷ IPv6

- No broadcast
- Multicast
 - FF00::/8
 - FF02:0:0:0:0:0:0:1 All Nodes Address
 - FF02:0:0:0:0:0:0:2 All Routers Address...
 - See reference

Where do I get an address?

▷ IPv4

- DHCP server (or manually assigned)

▷ IPv6

- Router advertisement
- Interface creates its own address
- DHCPV6?

NAT/address sharing?

- ▷ IPv4
 - Yes
- ▷ IPv6
 - No

Is it live?

- ▷ Ping google public DNS address
- ▷ IPv4

```
ping -c 3 8.8.8.8
```

- ▷ IPv6

```
ping6 -c 3 2001:4860:4860::8888
```

Can I mix?

- ▷ IPv4 talking to IPv6
 - No, but
 - Can use IPv6 to "::IP.v4.add.ress"
 - ::8.8.8.8 = 0:0:0:0:0:0:8.8.8.8
 - Still technically IPv6
- ▷ Use a proxy in between

Application support

- ▷ **Very strong now**
 - Hard to find apps that can't use IPv6
 - Code level: listen on ":::", can accept both IPv4 and IPv6 connections
- ▷ **Occasionally parallel tools**
 - ping/ping6
 - traceroute/traceroute6
- ▷ **DOD mandated IPv6**

Listening on IPv6 with python

https://github.com/activecm/save_json_stream/blob/main/save_json_stream.py

```
def create_server(listening_port, max_connections):
    """Create the initial listening server socket."""

    try:
        server_h = socket.socket(socket.AF_INET6, socket.SOCK_STREAM)
    except OSError:
        server_h = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    try:
        server_h.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        server_h.bind(('', int(listening_port)))
        server_h.listen(max_connections)
    except PermissionError:
        fail('Unable to listen on port ' + str(listening_port))
    Debug('Listening on TCP port ' + str(listening_port), True)

    sel_objs.register(server_h, selectors.EVENT_READ, handle_accept)
```

#We try to open an IPv6 listener (which also accepts IPv4). If this fails
#...we retry with IPv4 only.

Capturing packets

- ▷ Packet capture tools can save both
 - Pcap format supports both
- ▷ Some?/Many?/Most? have IPv6 decoders
- ▷ Tell the difference?
 - First 4 bits after the physical header
 - "Version" field in both IPv4 and IPv6
 - 0x4 = 4 for IPv4
 - 0x6 = 6 for IPv6
 - (5 was experimental: https://en.wikipedia.org/wiki/Internet_Stream_Protocol)
 - Wireshark displays

IPv4 and IPv6: which is used?

- ▷ Linux prefers IPv6 now
 - Try IPv6 first, then fall back to IPv4
- ▷ Windows ($\geq$ Vista) prefers IPv6
 - To change:
 - <https://docs.microsoft.com/en-us/troubleshoot/windows-server/networking/configure-ipv6-in-windows>
- ▷ MacOS (prefers IPv6 if A + AAAA returned)
 - To disable IPv6:
 - <https://choibles.com/revert-mac-to-ipv4/>

Packets

- ▷ BPFs
 - IPv4: use 'ip'
 - IPv6: use 'ip6'
- ▷ <https://www.networkacademy.io/ccna/ipv6/ipv4-vs-ipv6>
- ▷ TCP and UDP headers
 - No change between IPv4 and IPv6
- ▷ ICMP vs ICMPv6
- ▷ Bare Ipv6 header
 - Additional stuff moved to other headers

Firewall/IDS/IPS/Network monitor

- ▷ **Totally separate for IPv4 and IPv6**
 - Need to implement both!

DNS

▷ IPv4

- A records: hostname->IPv4 address
- PTR records: IPv4 address-> hostname

▷ IPv6

- AAAA records: hostname->IPv6 address
- PTR records: IPv6 address-> hostname

Hostname for an IP address

▷ IPv4

```
dig +short -x 8.8.4.4
```

```
Dns.google.
```

```
○ 4.4.8.8.in-addr.arpa. 86400 IN PTR dns.google.
```

▷ IPv6

```
dig +short -x 2001:4860:4860::8844
```

```
Dns.google.
```

```
○ 4.4.8.8.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.6.8.4.0.6.8.4.1.0.0.2.ip6.arpa. 86400 IN PTR dns.google.
```

Mac address: privacy

- ▷ IPv6 EUI-64
 - Obsoleted
- ▷ 6 byte mac address ("11:22:33:44:55:66")
- ▷ -> 8byte value, with ff:fe in the middle
 - 1122:33ff:fe44:5566
 - **And** invert bit 7 from the left:
 - 1322:33ff:fe44:5566
- ▷ Can tell who made the NIC from address
- ▷ Replaced by hash

References

▷ Google DNS

- <https://developers.google.com/speed/public-dns/docs/using>

▷ EUI-64

- <https://packetlife.net/blog/2008/aug/04/eui-64-ipv6/>
- -> <https://www.rfc-editor.org/in-notes/rfc4941.txt>

▷ IPv6 Multicast addresses

- <https://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.xhtml>

- ▷ https://en.wikipedia.org/wiki/Comparison_of_IPv6_support_in_operating_systems

More references

- ▷ <https://ipv6int.net/systems/>
- ▷ <https://www.activecountermeasures.com/malware-of-the-day-ipv6-address-aliasing/>
- ▷ https://en.wikipedia.org/wiki/IPv6_address
- ▷

Next Fireside Friday

- ▷ 3/28's presentation
 - Packet crafting!
 - Please install ncat and hping3
 - RPM-based systems

```
sudo yum -y install nmap-ncat hping3
```

- Deb-based systems

```
sudo apt -y install ncat hping3
```

- ▷ Only if above are not available
 - Alternate names:
 - nmap-ncat, netcat

Questions?



...