

IP version 6 security considerations

Jun-ichiro itojun Hagino
KAME/WIDE Project
Research Laboratory, Internet Initiative Japan
`itojun@{iijlab,kame}.net`

Outline

- Transition to IPv6 internet (or IPv4/v6 dual stack internet) is imminent
- What will change wrt internet security with IPv6?
- What kind of caveats are there?

- IPv6 features and IPv6 transition stages
- Good stuffs and bad stuffs
- More details on bad stuffs

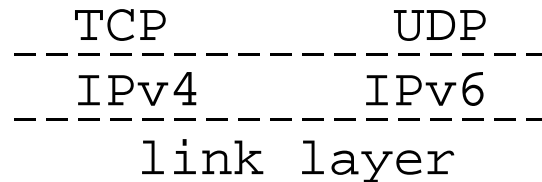
IPv6 protocol stack

□ IPv6

- Implemented independently from IPv4 stack

□ TCP/UDP

- Code is shared in many cases among IPv4/v6 (esp. TCP)
- RFC does not specify how to implement
 - there are implementations w/ TCP6 and TCP4
- There are implementation differences due to spec ambiguities (later)



Key differences between IPv6 and IPv4

- IPv6 address is written in hexadecimal format
 - 2001:240:5ff:0:220:e0ff:fe8d:3a8c
- Subnet mask is always /64
 - No variable subnet masks like /28, /29, and stuff
 - (you will see /n routes in routing table, of course)
- It is normal to associate multiple address to an interface
 - With IPv4, it was normally assumed that a node has single IPv4 address
- Scoped IPv6 addresses
 - Link-local address - 128bit is not enough to disambiguate destination
 - fe80::1%interface1

```
fe80::1 ethernet A          ethernet B
----- my machine ----- fe80::1
interface 1          interface 2
```

IPv6 protocol stack and API

- AF_INET6: similar to AF_INET
- `sizeof(sockaddr_in6) > sizeof(sockaddr_in) = sizeof(sockaddr)`

- RFC2553/3493 basic API
 - TCP/UDP: socket, bind, connect - same
 - address family independent hostname lookup (getaddrinfo, getnameinfo)
- RFC2292/3542 advanced API
 - raw/ICMPv6, extension headers handling (ping, traceroute)
 - (only very few apps need this)

- IPv6 support = rewrite apps to use RFC2553/3493-based API as needed

```
struct sockaddr_in6 {  
    u_int8_t  sin6_len;  
    u_int8_t  sin6_family;  
    u_int16_t sin6_port;  
    u_int32_t sin6_flowinfo;  
    struct in6_addr sin6_addr;  
    u_int32_t sin6_scope_id;  
};
```

Transitioning to IPv6

- There will be no flag day - we will transition gradually

- "Dual stack"
 - Every IPv6 nodes in the early stage will be capable of speaking IPv4
 - IPv4/v6 nodes will use IPv6 to communicate with IPv6 nodes
 - IPv4/v6 nodes will use IPv4 to communicate with IPv4 nodes
 - The use of IPv6 will gradually increase, and become dominant

- "Tunnelling"
 - IPv6 networks may be at distant locations, not directly adjacent to each other
 - Encapsulate IPv6 packet into IPv4 to deliver packets through IPv4 networks
 - Just like mbone/VPN tunnels

Good stuff

- Huge address space
 - No need for NAT and other wacky hacks
 - Much harder to portscan
- Simpler administration
 - No need for stateful DHCP server
 - No need for renumber when subnet gets filled up
- IPsec available by default
 - Key management is still a big headache, still it's good to have ESP/AH available everywhere
- No NAT -> management is easier
- No NAT -> more p2p apps -> efficient use of network
- Extension header chain -> extensibility
- Multicast -> efficient use of network

Bad stuff (need to be cautious)

- IPv4 mapped address
- Twists in new API (getnameinfo)
- Heavy use of tunnelling
- No NAT -> firewalling/fire suit is important
- No NAT -> more p2p apps -> difficult to filter/police
- Extension header chain -> difficult to filter
- Multicast -> difficult to track down
- Additional Complexity due to multiple protocol families
- Specification unclarity/implementation incompatibility

IPv4 mapped address (RFC2553/3493)

- `::ffff:127.0.0.1`
 - Representation of 127.0.0.1 on top of AF_INET6 socket
 - (real peer is IPv4, 127.0.0.1)

- Not distinguishable between the following two:
 - IPv4 packet goes into AF_INET6 socket (kernel translates the address)
 - Real IPv6 packet with `::ffff:127.0.0.1`

- Ambiguity leads to Security hole
 - Malicious party could circumvent access control
 - Malicious party could make you generate unexpected IPv4 packet
 - Services that flip src/dst - DNS server, udp echo, ...

- Solution
 - Do not use IPv4 mapped address by `setsockopt(IPV6_V6ONLY, 1)`, open AF_INET socket for IPv4
 - Specification picked insecure behavior as default!
 - Some OSes (OpenBSD, NetBSD, FreeBSD) are cautious and picked safer default

getnameinfo (RFC2553/3493)

□ Scenario

- getnameinfo may return FQDN (DNS PTR query) or numeric address string
- Bad guy can configure malicious PTR record
 - 1.1.1.10.in-addr.arpa. IN PTR 127.0.0.1
- getnameinfo will return "127.0.0.1" when 10.1.1.1 sockaddr_in is passed
- Caller cannot know if it is the result of PTR lookup, or inet_ntop(3)

□ Solution

- If you use getnameinfo for access control, use the following construct (next page)

Security: getnameinfo

```
error = getnameinfo(sa, salen, addr, sizeof(addr),
    NULL, 0, NI_NAMEREQD);
if (error == 0) {
    /* could be malicious PTR record */
    memset(&hints, 0, sizeof(hints));
    hints.ai_socktype = SOCK_DGRAM; /*dummy*/
    hints.ai_flags = AI_NUMERICHOST;
    if (getaddrinfo(addr, "0", &hints, &res) == 0) {
        /* malicious PTR record */
        freeaddrinfo(res);
        printf("bogus PTR record\n");
        return -1;
    }
    /* addr is FQDN as a result of PTR lookup */
} else {
    /* addr is numeric string */
    error = getnameinfo(sa, salen, addr, sizeof(addr),
        NULL, 0, NI_NUMERICHOST);
}
```

Heavy use of tunnelling

- There are numerous RFC for IPv6 transition technologies that use IPv6-over-IPv4 tunnel
 - RFC1933/2893 configured tunnel
 - RFC1933/2893 automatic tunnel
 - RFC2401 IPsec tunnel
 - RFC2473 IPv6 generic packet tunnelling
 - RFC2529 6over4 tunnel
 - RFC3056 6to4 tunnel
 - isatap tunnel
 - mobile-ip6 (uses RFC2473)
- Difficult to filter/police
- Solution: construct simple IPv6 network
 - RFC1933/2893 configured tunnel will suffice
 - Other transition technologies are frills
 - If native IPv6 network exists already, there's no need for transition technologies

Importance of firewalling/fire suit

- There will be no NAT
 - (NAT does not provide security, btw)

- Absence of NAT will promote p2p applications
 - Packet filtering at the edge will become very difficult
 - "Fire suit" instead of firewalling - OS vendors must take a security stance

- Extension header chain makes filtering difficult
 - Need to dig deeper into packet to know the actual protocol being used
 - DoS possibility if the IPv6 stack implemented without care
 - (there's no upper limit to number of extension headers specwise)

```
[IPv6, next=routing] [routing, next=TCP] [TCP] [TCP payload]
```

Multicast support

- IPv6 uses on-link multicast heavily for MAC address resolution and autoconfiguration
 - Make sure to get a good driver/card/switch
 - This does not constitute problems

- IPv6 hosts are capable of multicast
- With IPv6, multicast people wants to deploy Internet-wide multicast

- Routing protocol: PIM-SM, maybe with MBGP

- Problem areas: How to filter/police? Tracking down problems?

Additional Complexity due to multiple protocol families

- OS kernel support and applications support are usually separate
- Need to ship a single binary that works on IPv6-enabled and IPv6-disabled kernel
 - AF_INET6 support may not be there
- A support nightmare
 - IPv4 kernel + IPv4 userland
 - IPv4 kernel + IPv4/v6 dual stack userland
 - IPv4/v6 kernel + IPv4/v6 dual stack userland
 - IPv4 kernel + IPv4/v6 dual stack userland
 - "IPv6 code is present but not configured" case
 - no IPv6 router, for instance
- Solution: an application should handle both IPv4 and IPv6 without problem even without kernel support
 - Remove any code that assumes AF_INET
 - Do not add any code that hardcodes AF_INET6

Specification unclarity/incompatibility

□ Port number space

- What is the relationship of port number space between TCP over v4 versus TCP over v6?
- Does bind(2) ordering matter?
- Solution: Take the safest side, do not assume kernel behavior
 - getaddrinfo(AI_PASSIVE)
 - bind(2) to all addresses returned
 - Die only if all bind(2) fails

□ IPv6 wildcard bind...

- may receive IPv4 traffic as IPv4 mapped addr (RFC2553/3493)
- Leads to mistakes/vulnerabilities, low portability, don't use it
- Use setsockopt(IPV6_V6ONLY, 1) to avoid mistake
 - IPV6_V6ONLY is just recently introduced, so platforms may not support this

Other tips and tricks

- Never use colon as database separator (awk/perl), config separator and such
 - need to hold IPv6 address eventually
 - if you really need to, use [addr]:port (RFC2732)

- Parsing IPv6 address string
 - it is IMPOSSIBLE to write an regex! simply use getaddrinfo(3) to test.

Summary

- Transition to IPv6 internet is imminent

- What will change wrt internet security with IPv6?
- What kind of caveats are there?

- IPv6 features and IPv6 transition stages
- Good stuffs and bad stuffs
- More details on bad stuffs

- Be prepared, it's already out there waiting for you
 - WinXP and MacOS X have it already!