IPv6 End-node Firewall

Shin Shirahata <shin@clara.ad.jp>
Clara Online, Inc.

Agenda

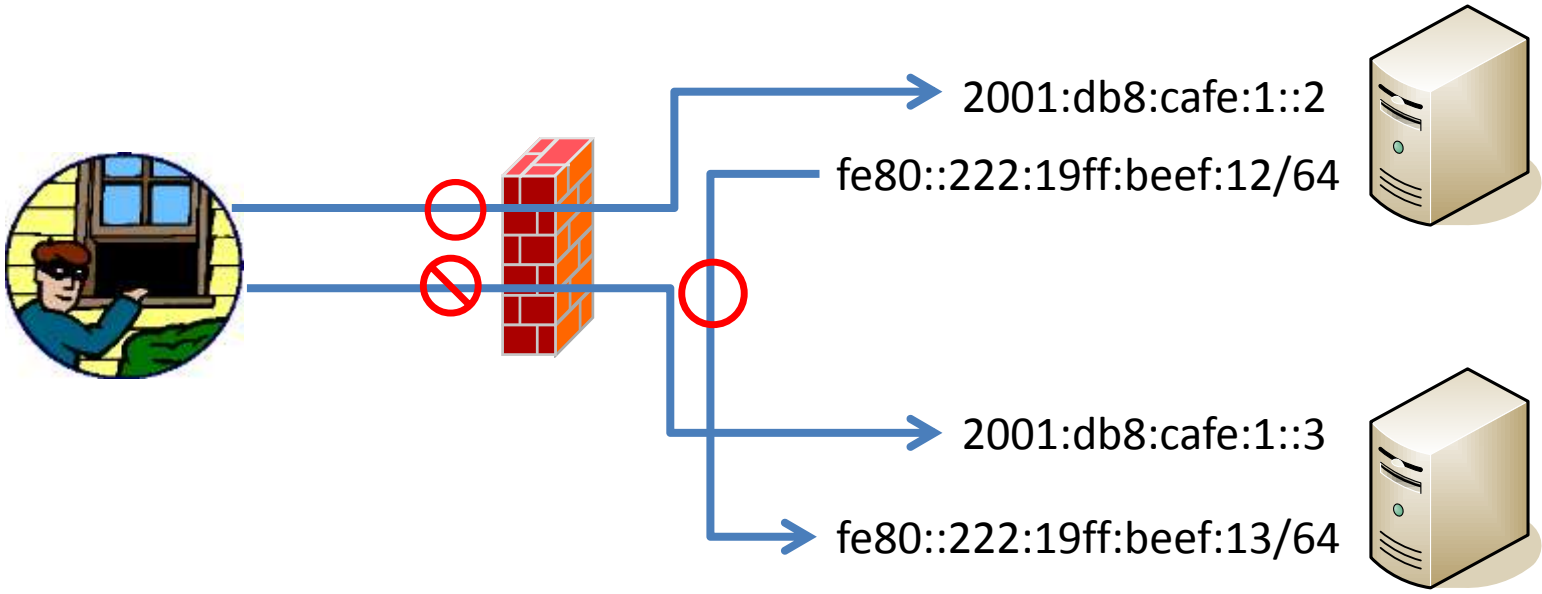
- Packet filtering on IPv6
- TCP_Wrappers
- Automatic tunnel
- Middlebox security

Introduction

- Why End-node Firewall is important?
  - Network firewall could not cover all security issues.
    - access from a link-local source address can be handled by server side application access control
  - Tunnel or Encryption (e.g. SSL or VPN) makes difficult to control traffic at a Firewall.
- This slide explain about end-node firewall other than network firewall

Link Local Address

- Link Local Address could be used as a backdoor.



If you don't need to use link-local address for servers, it’s recommended to filter the packets:
“DENY ALL by default” rules might be useful.

ICMPV6:

TO BE FILTER, OR NOT TO BE FILTER : THAT IS THE QUESTION

Firewall on IPv6

- What is the difference from IPv4 firewall?
  - TCP and UDP: no difference
  - ICMPv6: plays more important role in compare to ICMPv4.
    - “Deny All” approach doesn’t work on IPv6.
    - E.g: IPv6 nodes SHOULD implement Path MTU Discovery (RFC1981)

... that overly aggressive filtering of ICMPv6 by firewalls may have a detrimental effect on the establishment and maintenance of IPv6 communications. On the other hand, allowing indiscriminate passage of all ICMPv6 messages can be a major security risk.

"RFC4890: Recommendations for Filtering ICMPv6 Messages in Firewalls"

General strategy of ICMPv6 traffic filtering

- Permit necessary ICMPv6 traffic
  - Traffic That Must Not Be Dropped
  - Traffic That Normally Should Not Be Dropped
- Drop other ICMPv6 traffic
  - Traffic That Will Be Dropped Anyway -- No Special Attention Needed
  - Traffic for Which a Policy Should Be Defined
  - Traffic That Should Be Dropped Unless a Good Case Can Be Made

Categorization by RFC4890 - This slide explains about traffic filtering on end-node. For filtering on network –based firewall, refer to section 4.3 of RFC4890

ip6tables

Note: This slide assumes ip6tables on CentOS6

- iptables command is used for IPv4 packet filtering. ip6tables command is used for IPv6.
  - ip6tables: syntax is same as iptables. different configuration file:
    - iptables: /etc/sysconfig/iptables
    - ip6tables: /etc/sysconfig/ip6tables
- Stateful packet inspection (SPI)
  - SPI feature (--state NEW and ESTABLISHED, RELATED) feature was introduced since Linux kernel 2.6.20 and iptables v1.3.5
    - Older distribution (e.g. CentOS 5, kernel 2.6.18) doesn't support **state** module on IPv6. We suggest to use other distribution which come with Kernel 2.6.20 or later for packet filtering (e.g. CentOS 6).

Note: icmpv6 options on ip6tables

```
$ ip6tables -p ipv6-icmp -h
(snip)
Valid ICMPv6 Types:
```

- destination-unreachable
  - no-route
  - communication-prohibited
  - address-unreachable
  - port-unreachable
- packet-too-big
- time-exceeded (ttl-exceeded)
  - ttl-zero-during-transit
  - ttl-zero-during-reassembly
- parameter-problem
  - bad-header
  - unknown-header-type
  - unknown-option
- echo-request (ping)
- echo-reply (pong)
- router-solicitation
- router-advertisement
- neighbour-solicitation (neighbor-solicitation)
- neighbour-advertisement (neighbor-advertisement)
- redirect

Traffic That Must Not Be Dropped: Destination Unreachable (Type 1)

- Destination Unreachable (Type 1) - All codes
  - Useful for debugging
  - Important to speed up cycling through possible addresses, as they can avoid the need to wait through timeout
    - **e.g. Fallback to IPv4.**

```
ip6tables -A INPUT -p icmpv6 -m state --state ESTABLISHED,RELATED ¥
 --icmpv6-type destination-unreachable -j ACCEPT
```

Traffic That Must Not Be Dropped: Packet Too Big (Type 2)

- Packet Too Big (Type 2)
  - Vital to the correct functioning of Path MTU Discovery
    - Since routers are not allowed to fragment packets in IPv6, informing sources of the need to fragment large packets is more important than for IPv4.
  - Parts of the Internet will become inaccessible
    - If these messages are not generated when appropriate, hosts will continue to send packets that are too large or may assume that the route is congested.

```
-A INPUT -p icmpv6 -m state --state ESTABLISHED,RELATED ¥
 --icmpv6-type packet-too-big -j ACCEPT
```

..or set MTU as 1280 (Guaranteed Minimum MTU)

Traffic That Must Not Be Dropped: Time Exceeded (Type 3)

- Time Exceeded Error Message
- Code 0
  - Generated at any node on the path being taken by the packet and sent, any-to-end between unicast addresses, if the Hop Limit value is decremented to zero at that node.
  - Code 0 messages can be needed if the path to a particular destination requires an unusually large number of hops.

```
-A INPUT -p icmpv6 -m state --state ESTABLISHED,RELATED ¥
 --icmpv6-type ttl-zero-during-transmit -j ACCEPT
```

Traffic That Must Not Be Dropped: Parameter Problem (Type 4)

- Parameter Problem (Type 4) - Codes 1 and 2 only
  - Parameter Problem Code 1 (Unrecognized Next Header) and Code 2 (Unrecognized IPv6 Option) messages may result if a node on the path (usually the destination) is unable to process a correctly formed extension header or option.
  - If these messages are not returned to the source, communication cannot be established, as the source would need to adapt its choice of options probably because the destination does not implement these capabilities.
  - Hence, these messages need to be generated and allowed for effective IPv6 communications.

```
ip6tables -A INPUT -p icmpv6 -m state --state ESTABLISHED,RELATED ¥
 --icmpv6-type unknown-header-type -j ACCEPT
ip6tables -A INPUT -p icmpv6 -m state --state ESTABLISHED,RELATED ¥
 --icmpv6-type unknown-option -j ACCEPT
```

Traffic That Must Not Be Dropped: ICMPv6 Echo Request and Echo Response

- ICMPv6 Echo Request and Echo Response
  - Echo Request (Type 128)
  - Echo Response (Type 129)
    - Dropping connectivity checking messages will prevent the firewall being the destination of a Teredo tunnel and it is not considered necessary to disable connectivity checking in IPv6 networks because port scanning is less of a security risk.

```
ip6tables -A INPUT -p icmpv6 --icmpv6-type echo-request -j ACCEPT
ip6tables -A INPUT -p icmpv6 -m state --state ESTABLISHED,RELATED ¥
 --icmpv6-type echo-reply -j ACCEPT
```

Address Configuration and Router Selection messages (1/4)

- Address Configuration and Router Selection messages
  - There are a number of other sets of messages that play a role in configuring the node and maintaining unicast and multicast communications through the interfaces of a node.
  - **These messages must not be dropped if the node is to successfully participate in an IPv6 network.**

Address Configuration and Router Selection messages (2/4)

- Address Configuration and Router Selection messages:
  - Router Solicitation (Type 133)
  - Router Advertisement (Type 134)
  - Neighbor Solicitation (Type 135)
  - Neighbor Advertisement (Type 136)
  - Inverse Neighbor Discovery Solicitation (Type 141)
  - Inverse Neighbor Discovery Advertisement (Type 142)

```
ip6tables -A INPUT -p icmpv6 -m icmp6 --icmpv6-type router-solicitation -m hl ¥
 --hl-eq 255 -j ACCEPT
ip6tables -A INPUT -p icmpv6 -m icmp6 --icmpv6-type router-advertisement -m hl ¥
 --hl-eq 255 -j ACCEPT
ip6tables -A INPUT -p icmpv6 -m icmp6 --icmpv6-type neighbour-solicitation -m hl ¥
 --hl-eq 255 -j ACCEPT
ip6tables -A INPUT -p icmpv6 -m icmp6 --icmpv6-type neighbour-advertisement  -m hl ¥
 --hl-eq 255 -j ACCEPT
ip6tables -A INPUT -p icmpv6 -m icmp6 --icmpv6-type redirect -m hl --hl-eq 255 -j ACCEPT
ip6tables -A INPUT -p icmpv6 -m icmp6 --icmpv6-type 141 -m hl --hl-eq 255  -j ACCEPT
ip6tables -A INPUT -p icmpv6 -m icmp6 --icmpv6-type 142 -m hl --hl-eq 255  -j ACCEPT
```

Address Configuration and Router Selection messages (3/4)

- Link-Local Multicast Receiver Notification messages:
  - Listener Query (Type 130)
  - Listener Report (Type 131)
  - Listener Done (Type 132)
  - Listener Report v2 (Type 143)

```
ip6tables -A INPUT -p icmpv6 -s fe80::/10 -m icmp6 --icmpv6-type 130 -j ACCEPT
ip6tables -A INPUT -p icmpv6 -s fe80::/10 -m icmp6 --icmpv6-type 131 -j ACCEPT
ip6tables -A INPUT -p icmpv6 -s fe80::/10 -m icmp6 --icmpv6-type 132 -j ACCEPT
ip6tables -A INPUT -p icmpv6 -s fe80::/10 -m icmp6 --icmpv6-type 143 -j ACCEPT
```

Address Configuration and Router Selection messages (4/4)

- SEND Certificate Path Notification messages:
  - Certificate Path Solicitation (Type 148)
  - Certificate Path Advertisement (Type 149)

```
ip6tables -A INPUT -p icmpv6 -m icmp6 --icmpv6-type 148 -m hl --hl-eq 255 -j ACCEPT
ip6tables -A INPUT -p icmpv6 -m icmp6 --icmpv6-type 149 -m hl --hl-eq 255 -j ACCEPT
```

- Multicast Router Discovery messages:
  - Multicast Router Advertisement (Type 151)
  - Multicast Router Solicitation (Type 152)
  - Multicast Router Termination (Type 153)

```
ip6tables -A INPUT -p icmpv6 -s fe80::/10 -m icmp6 --icmpv6-type 151 -m hl ¥
 --hl-eq 1 -j ACCEPT
ip6tables -A INPUT -p icmpv6 -s fe80::/10 -m icmp6 --icmpv6-type 152 -m hl ¥
--hl-eq 1 -j ACCEPT
ip6tables -A INPUT -p icmpv6 -s fe80::/10 -m icmp6 --icmpv6-type 153 -m hl ¥
--hl-eq 1 -j ACCEPT
```

Traffic That Normally Should Not Be Dropped

- Time Exceeded (Type 3) - Code 1

```
ip6tables -A INPUT -p icmpv6 -m state --state ESTABLISHED,RELATED ¥
 --icmpv6-type ttl-zero-during-reassembly -j ACCEPT
```

- Parameter Problem (Type 4) - Code 0

```
ip6tables -A INPUT -p icmpv6 -m state --state ESTABLISHED,RELATED ¥
 --icmpv6-type bad-header -j ACCEPT
```

Traffic That Will Be Dropped Anyway -- No Special Attention Needed

- Router Renumbering messages must be authenticated using IPsec, so it is not essential to filter these messages even if they are not allowed at the firewall/router:
  - Router Renumbering (Type 138)

```
ip6tables -A INPUT -p icmpv6 -m icmp6 --icmpv6-type 138 –j DROP
```

- Mobile IPv6 messages that are needed to assist mobility:
  - Home Agent Address Discovery Request (Type 144)
  - Home Agent Address Discovery Reply (Type 145)
  - Mobile Prefix Solicitation (Type 146)
  - Mobile Prefix Advertisement (Type 147)
- It may be desirable to drop these messages, especially on public interfaces, if the firewall is not also providing mobile home agent services, but they will be ignored otherwise.
- The message used by the experimental Seamoby protocols may be dropped but will be ignored if the service is not implemented:
  - o Seamoby Experimental (Type 150)

Traffic for Which a Policy Should Be Defined

- Traffic for Which a Policy Should Be Defined Redirect messages provide a significant security risk, and administrators should take a case-by-case approach to whether firewalls, routers in general, and other nodes should accept these messages:
  - Redirect (Type 137)

```
ip6tables -A INPUT -p icmpv6 -m icmp6 --icmpv6-type 137 –j DROP
```

    - Conformant nodes must provide configuration controls that allow nodes to control their behavior with respect to Redirect messages so that it should only be necessary to install specific filtering rules under special circumstances, such as if Redirect messages are accepted on private interfaces but not public ones.

Traffic for Which a Policy Should Be Defined

- If a node implements the experimental Node Information service, the administrator needs to make an explicit decision as to whether the node should respond to or accept Node Information messages on each interface:
  - Node Information Query (Type 139)
  - Node Information Response (Type 140)
- It may be possible to disable the service on the node if it is not wanted, in which case these messages will be ignored and no filtering is necessary. Error messages not currently defined by IANA:
  - Unallocated Error messages (Types 5-99 inclusive and 102-126 inclusive)

By defining “Drop by default” policy, these ICMPv6 traffic will be dropped

Traffic That Should Be Dropped Unless a Good Case Can Be Made

- Traffic That Should Be Dropped Unless a Good Case Can Be Made Messages with types in the experimental allocations:
  - Types 100, 101, 200, and 201.
- Messages using the extension type numbers until such time as ICMPv6 needs to use such extensions:
  - Types 127 and 255. A
- All informational messages with types not explicitly assigned by IANA, currently:
  - Types 154-199 inclusive and 202-254 inclusive. Note that the base ICMPv6 specification requires that received informational messages with unknown types must be silently discarded.

By defining “Drop by default” policy, these ICMPv6 traffic will be dropped

Implementing rate-limit

- Permitting ICMPv6 traffic is a necessary in IPv6, but it still vulnerable to DoS attack.
  - By using SPI, it could allow incoming ICMPv6 messages only for existing sessions
  - If SPI could not use, rate-limit is effective for mitigating DoS attack
- Example: Permit up to “10” ping message per

```
ip6tables -A INPUT -p icmpv6 --icmpv6-type echo-request -m limit --limit 10/minute ¥
-j ACCEPT
```

Note: As for my test, if “ip6tables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT” policy is defined, ICMPv6 rate-limit will not work in CentOS6.
Please use “ip6tables -A INPUT -p tcp -m state --state ESTABLISHED,RELATED -j ACCEPT” and “ip6tables -A INPUT -p udp -m state --state ESTABLISHED,RELATED -j ACCEPT” instead.

TCP_WRAPPERS

hosts.allow/hosts.deny

- Make sure that TCP_wrappers configurations are “dual stack”ed

- /etc/hosts.deny

```
ALL: ALL
```

For deny access from everywhere as default

- /etc/hosts.allow

```
sshd: 192.0.2.0/255.255.255.0
sshd: [2001:db8:1771:1::]/64
```

Grant access from 192.0.2.0/24 and [2001:db8:1771:1::]/64 for SSH

Reverse DNS and ACL on IPv4

- Reverse DNS based ACL works on most of IPv4 network.
  - In IPv4 network, network administrator configures reverse DNS entry for each IPv4 address (at least in Japan, majority of ISPs does this).
    - Example: ppp-hcm-123-45-67.example.net.
  - It's common to use domain name to implement ACL.
    - Easy to maintain: No configuration update is needed when additional IPv4 address prefixes are installed on the network and reverse DNS entries are configured.
  - Hosts.allow

```
sshd: .example.net
```

Reverse DNS and ACL on IPv6

- Reverse DNS based ACL does NOT works on IPv6
  - In IPv6 network, it's very difficult to configure reverse DNS entry for **ALL** IPv6 address since IPv6 has huge address space. It cause domain name based ACL will not work on IPv6.
- Use IPv6 address block instead of domain name
  - Network administrator can use IPv6 address block for configuring ACL. It's relatively very small number of address blocks are used.
  - Examples from global routing table:
    - IIJ (AS2497) originates 147 IPv4 prefixes and 5 IPv6 prefixes.
      - Several routes are advertised behalf of their customer.
    - BIGLOBE (AS2518) originates 32 IPv4 prefixes and 1 IPv6 prefix.
    - Clara Online (AS23661) originates 11 IPv4 prefix and 1 IPv6 prefix.

AUTOMATIC TUNNELS

Automatic Tunnel

- Automatic tunneling technologies (e.g. 6to4 and Teredo) could be possible backdoor on IPv4 side.
  - How automatic tunnel protocols are works?
  - How could be a security issue?

What is 6to4?

- A technique to send IPv6 packets over IPv4 network
  - User could get IPv6 Internet connectivity from IPv4 only Internet environment
    - Require a global IPv4 address on 6to4 node:
      Cannot be used through a NAT
  - Designed for IPv6 tradition mechanism
  - No explicit tunnel configuration required
- Difference from protocol translator
  - 6to4 does facilitate interoperation between *6to4 enabled* IPv4 nodes and IPv6 only nodes. It does *not* for IPv4 only nodes and IPv6 only nodes.

33

6to4 Address format

- 2002::/16
  - Special Address Prefix [RFC3056] is used for 6to4
- Maps an IPv4 address to an IPv6 address
  - IPv6 /48 will be given. Multiple /64 network can be used.

In a case of 192.0.2.42:

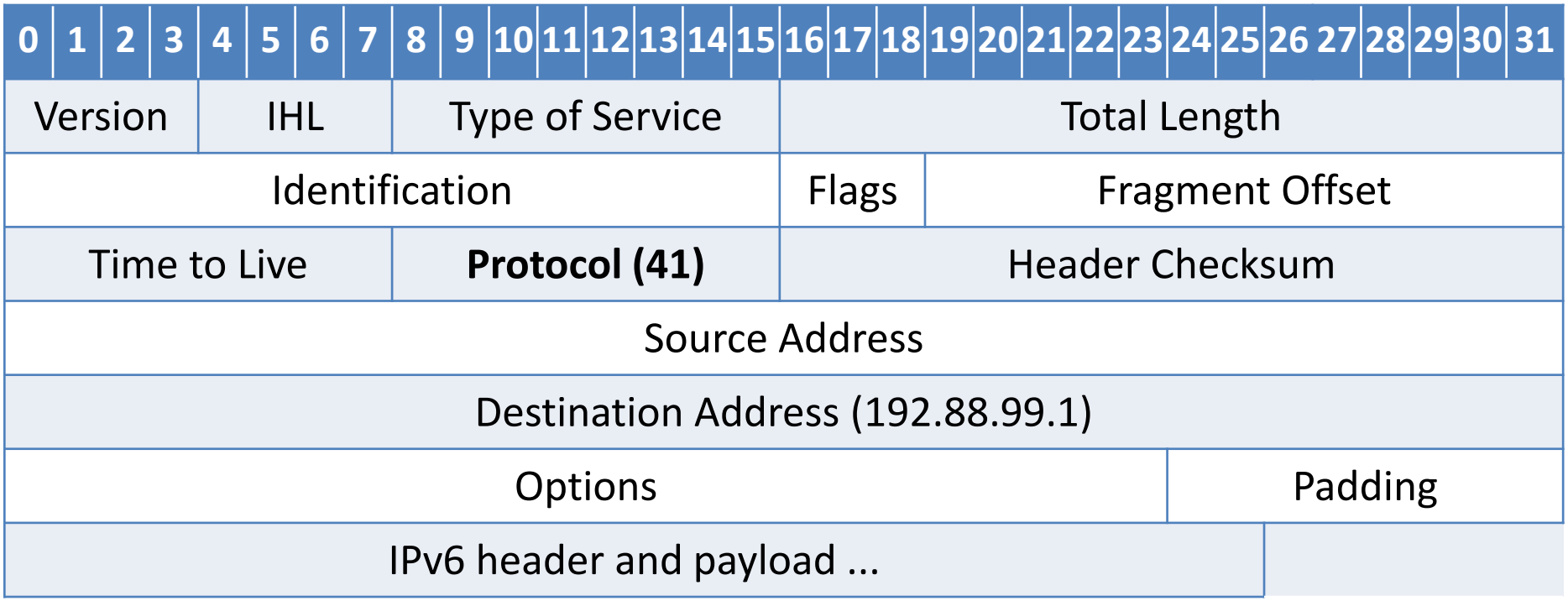
0~15	16-47	48-63	64-127
16bits	**32bits**	**16bits**	**64bits**
6to4 prefix	**IPv4 Address**	**SLA ID**	**Interface ID**
2002	**c000:022a**	**::c000:022a**	

192 0 2 42

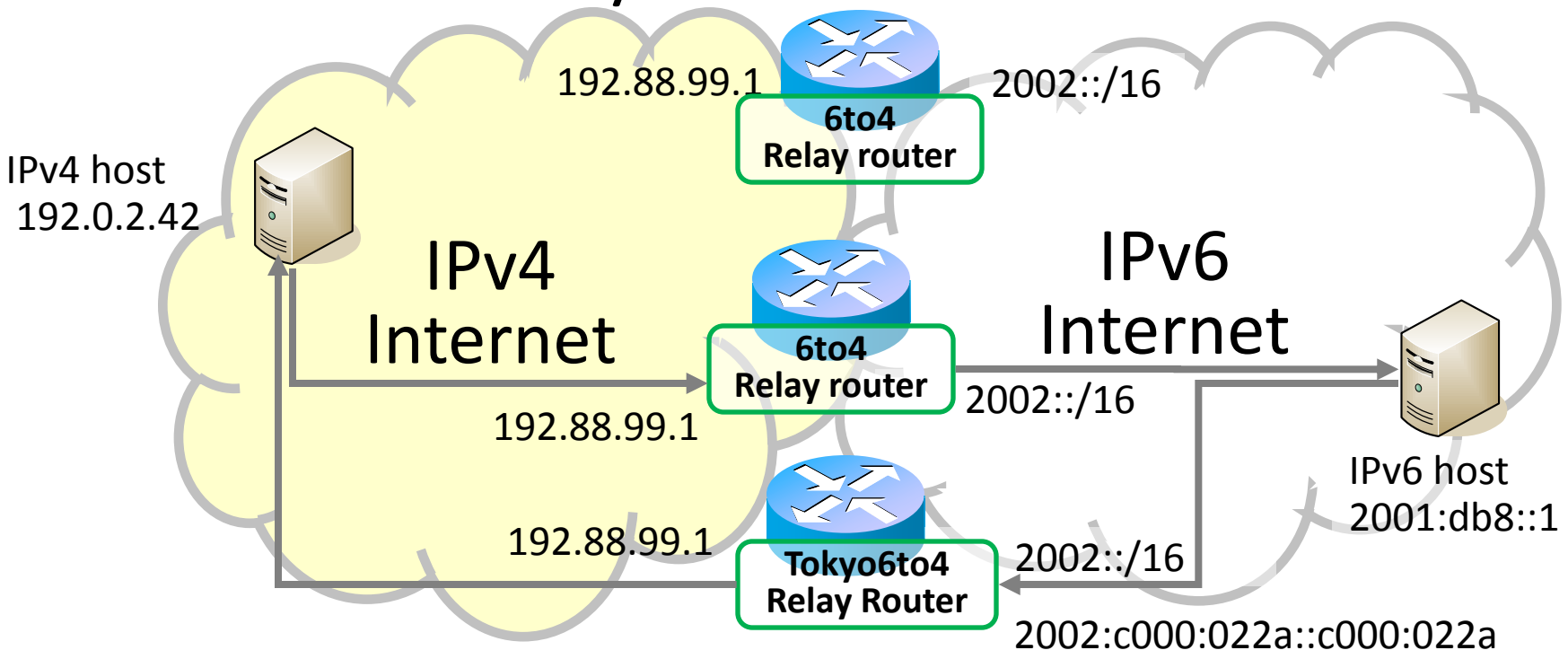
Usage of lower 80bit depends on implementation:

- **Windows –** "IPv4 address in HEX" (shown as example)
- **Linux –** "::1"

6to4 Packet Header Format



6to4 Relay Router and the IPv4/IPv6 Internet

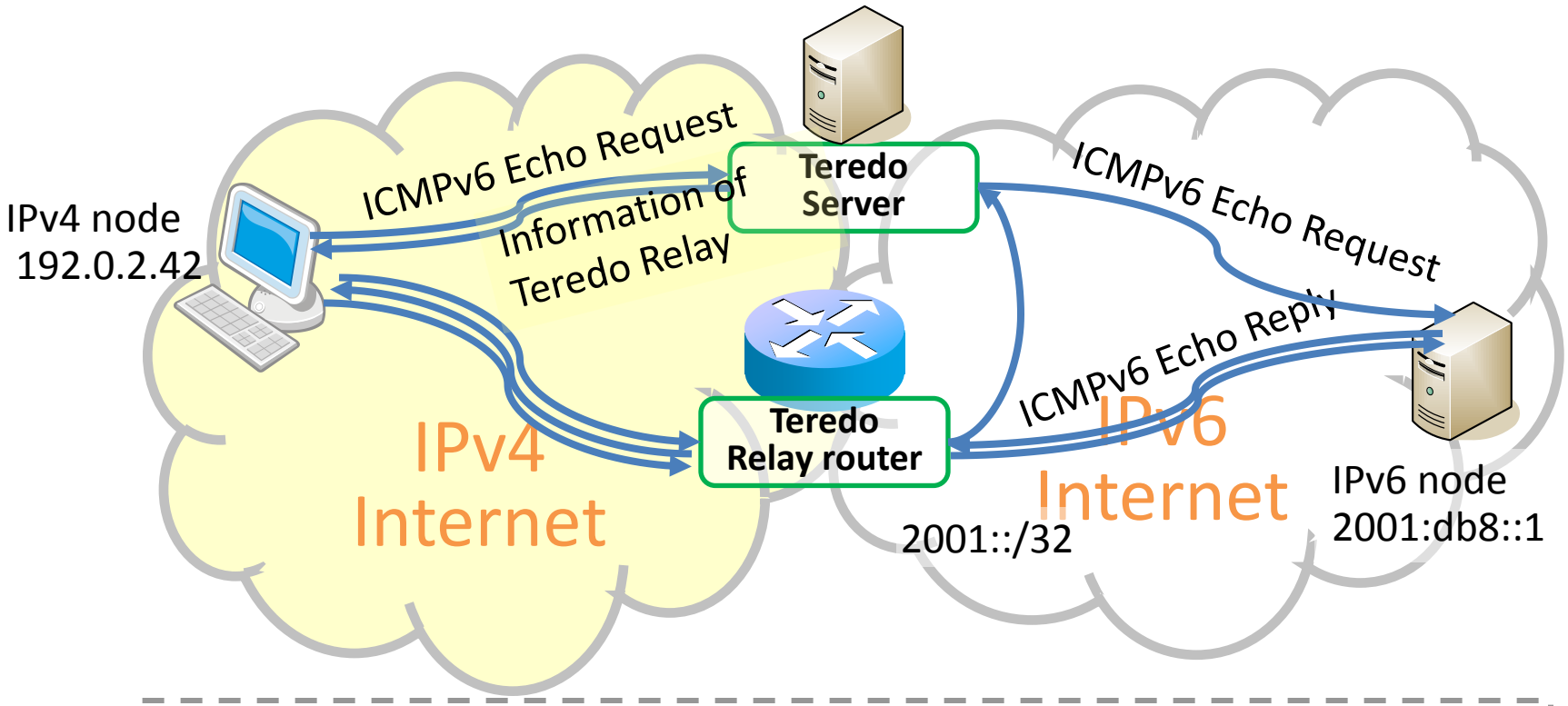


192.88.99.0/24 and 2002::/16 are Anycasted
Asymmetric routing results frequently

What is Teredo?

- One of IPv4 Automatic tunnel Technology
  - IPv6 packet is encapsulated to IPv4 UDP packet
  - /128 address will be allocated on Client
- Teredo could establish tunnel over various NAT implementation
  - It support Cone NAT, Restricted NAT
- Developed by Microsoft
  - Available on Windows XP or later
  - Open source implementation (Miredo) is also available on Mac OS X, Linux, *BSD
- Component of Teredo
  - Teredo Relay
    - Role: Translate IPv4 packet and IPv6 packet
    - Operated by various network operators
  - Server
    - Role: Session management
    - Microsoft's server is widely used
  - Client

Communication flow of teredo (In case of Restricted NAT)



TeredoでNAT越えに利用するパケットを「バブル」と呼ぶ
通信が確立されるまで数秒程度かかる

Behavior of Teredo (In case of Restricted NAT)

Preparation

1. Identify a NAT type
   - Teredo client send the packets to Teredo server **from inside NAT**
2. Teredo client sends the packets to IPv6 hosts via Teredo server
3. IPv6 hosts send the packets to Teredo relay. Teredo relay send the packet to the client **via Teredo server.**
   - At that time, client will gets Teredo relay router information
4. Teredo client starts communication with Teredo relay

IPv6 communication by Teredo

5. Teredo relay start communication with client

Teredo Address format

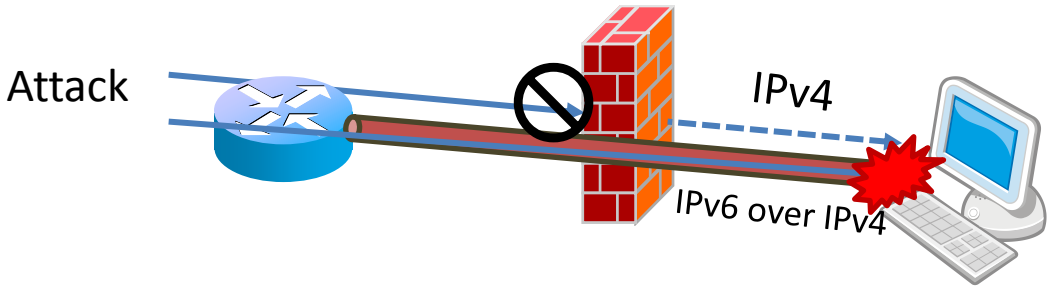
- 2001::/32
  - Special prefix for Teredo [RFC4380]
- Address of Teredo server
- Type of NAT Implementation (cone NAT or other)
- UDP Port Number and IPv4 address of NAT box
  - Hided by XOR
    Example: 3ffffdd2 xor ffffffff = c000022d. 192.0.2.45 in Decimal.

	0~31	32~63	64~79	80~95	96~127
length	32bits	32bits	16bits	16bits	32bits
Description	**Teredo Prefix**	**Teredo server IPv4 address**	**Flag**	**UDP Port**	**Client Public IPv4 Address**
Partial	**2001:0000**	**4136:e378**	**8000**	**63bf**	**3fff:fdd2**
Restore		65.54.227.120	cone NAT	40000	192.0.2.45

Source: Wikipedia, http://en.wikipedia.org/wiki/Teredo_tunneling

How could be a security issue?

- Example: Access control is applied on IPv4 connection, but not applied for IPv6 over IPv4 traffic (tunnel)
  - Firewall could be evadable if traffic is terminated at end-node
    - → External host can reach to an internal network: end-node will be exposed by tunnel
    - → Risk of backdoor



- If a tunnel is unnecessary, traffic filtering is recommended.

How to disable IPv6 automated tunnel: Windows

- Windows: 6to4, Teredo and ISATAP are enabled by default
  - How to disable Run “Command prompt” as Administrator.

    • Disabling 6to4
    netsh interface 6to4 set state disabled
    • Disabling Teredo
    netsh interface teredo set state disabled
    • Disabling ISATAP
    netsh interface isatap set state disabled

  - Confirm the 6to4/Teredo/ISATAP configuration

    • Show a 6to4 configuration
    netsh interface 6to4 show state
    • Show a Teredo configuration
    netsh interface teredo show state
    • Show an ISATAP configuration
    netsh interface isatap show state

- More details can be found at Microsoft’s "Netsh Commands for Interface (IPv4 and IPv6)" http://technet.microsoft.com/en-us/library/cc770948(v=ws.10).aspx

How to disable IPv6 automated tunnel: Windows (cont'd)

- You can disable IPv6 on all tunnel interfaces by "Microsoft Fix it 50412"
  - http://go.microsoft.com/?linkid=9728872
- Also, you can do the same settings by change the value in the Registry (not recommended, but may be useful in enterprise environment)

the **DisabledComponents** registry value in **HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Services¥Tcpip6¥Parameters ¥** registry sub key

- Type 0 to enable all IPv6 components. (Windows default setting)

…

- Type 0x01 to disable IPv6 on all tunnel interfaces. These include Intra-Site Automatic Tunnel Addressing Protocol (ISATAP), 6to4, and Teredo.

- More details can be found at "How to disable IP version 6 or its specific components in Windows"
  http://support.microsoft.com/kb/929852/en

How to filter 6to4 and Teredo traffic on the firewall

- 6to4
  - Block *protocol type 41* for both direction
    - Same as 6in4 manually configured tunnel.
- Teredo
  - Block outbound access to *UDP port 3544* for Teredo server
- ISATAP
  - Block *protocol type 41* for both direction
    - Or blocks DNS lookup for potential routers list (PRL) – typically named as an “isatap”. This way is less than perfect since relay router can be configured as IP address.

MIDDLEBOX SECURITY

Two method to support IPv6 on service

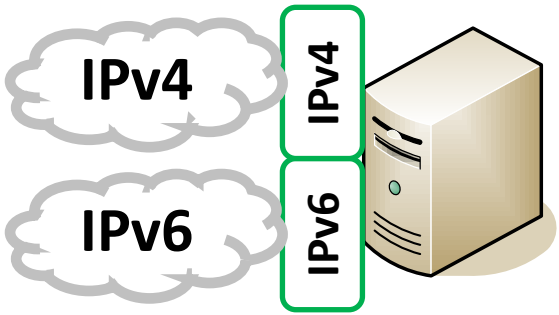
Server runs IPv4/IPv6

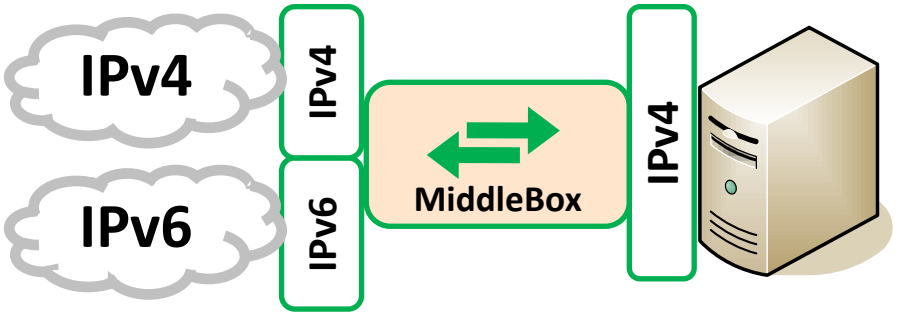
Dual Stack

- No IPv4/IPv6 translation

Server runs IPv4 (or IPv6) single stack only. Middle box is translating IPv4 and IPv6

- Middle box: Reverse proxy, load balancer, protocol translator, CDN etc





* Assumption: Network is running on IPv4 and IPv6 dual stack

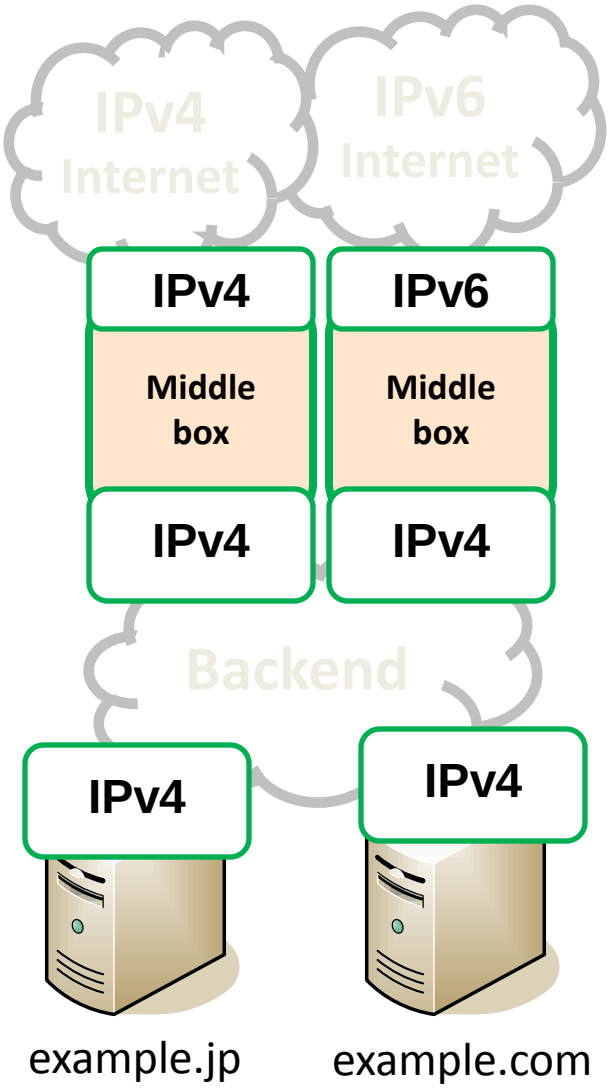
What is the middlebox

- Definition in RFC3234

  *"A middlebox is defined as **any intermediary device** performing functions other than the normal, standard functions of an **IP router** on the datagram path between a source host and destination host."*

- Network equipments that supports IPv4/IPv6 protocol translation
  - Load balancer with IPv4/v6 translation feature
  - Firewall that with IPv4/v6 translation feature
  - Protocol Translator
- Application level gateway
  - Reverse Proxy
  - CDN (Contents Delivery Network)

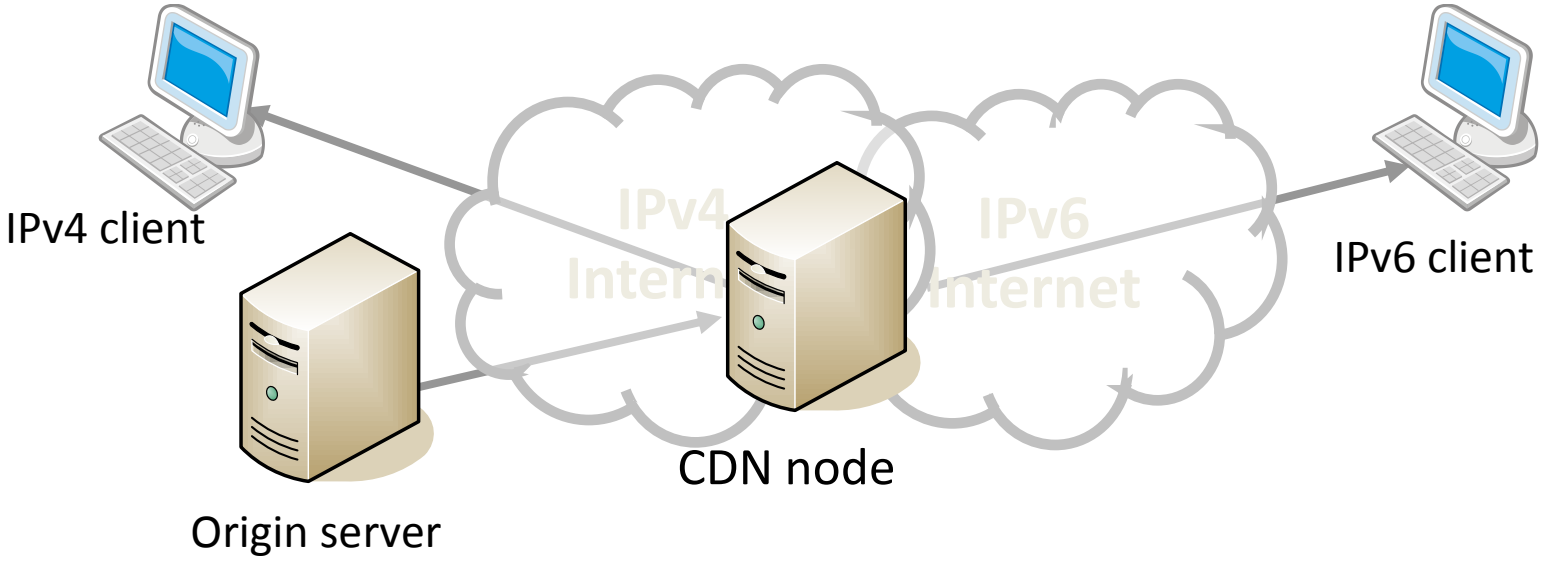
Middle box: Typical IPv4/IPv6 translation configuration



- Reverse Proxy
  - Reverse proxy server that running on front-end run IPv4 and IPv6 (or IPv6 only)
- Load balancer
  - Use IPv4 and IPv6 protocol translation feature
- Advantage
  - Dual stack operation is not necessary on server. Backend server will continue to use IPv4 only.
    - theoretically, IPv6 single stack server is possible.
  - No need to care about MTU of IPv4 side or IPv6 side since communications are terminated by application
- Disadvantage
  - Limitation of application layer protocol
  - Flexibility (e.g. setup a new domain name)

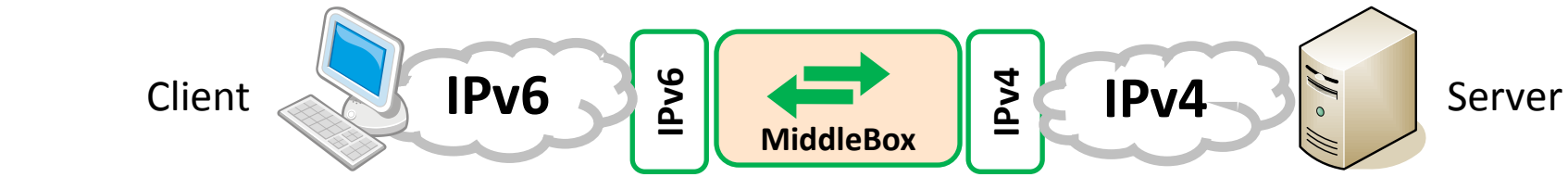
CDN as Middlebox

- Contents Delivery Network
  - Direct to nearest node of CDN network by using DNS or BGP Anycast
  - Kind of reverse proxies which is deployed over the world
    - If CDN provider supports IPv6, no IPv6 support is needed on origin server.



Logging Issue of middle box

- It is impossible to obtain source address of a connection which is come through middle box at server side.

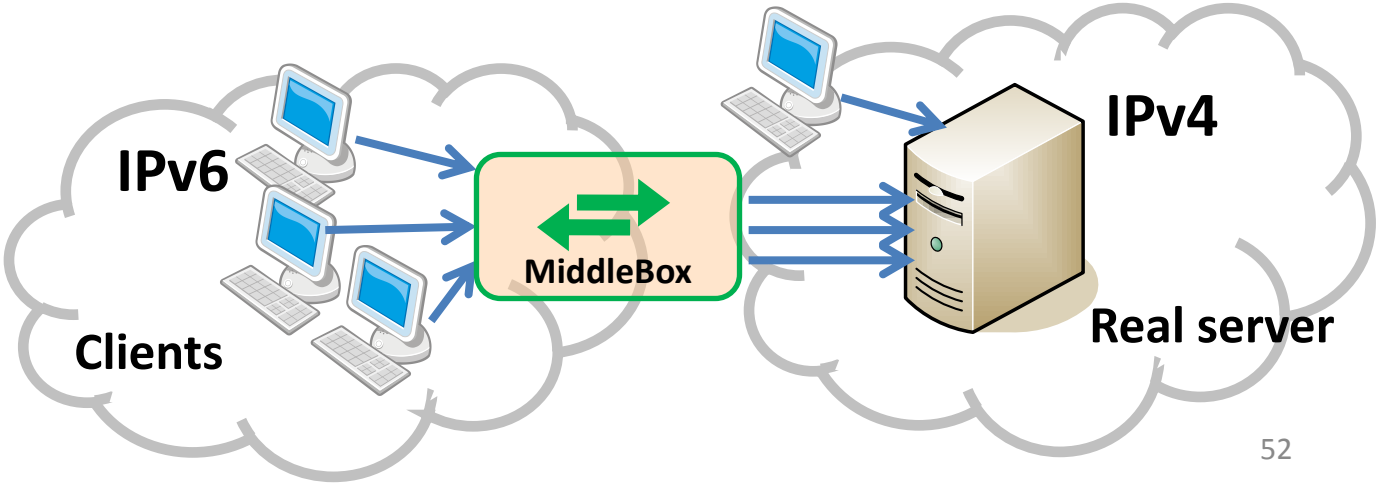


- Workaround:
  - When using HTTP Reverse Proxy or load balancer, configure special HTTP header for obtain original source IP address
  - X-Forwarded-For header:
    - Apache's "mod_remoteip "module
      - *Replaces the original client IP address for the connection with the useragent IP address list presented by a proxies or a load balancer via the request headers.*
      - https://httpd.apache.org/docs/trunk/mod/mod_remoteip.html
- In other cases, prep for log entry matching
  - Configure to obtain client's port number (actually, it's a port number of middle box) at web server side.

Issue on rate-limit of middle box

- It seems to be huge number of connections are coming from middle box on the real server
  - Seems like a DoS attack
  - Make sure that DoS protection or rate-limit feature are not activated against access from middle box

  →implement DoS protection or rate-limit feature on middlebox
- Access control on real server
  - ACL which is rely on client's IP address or domai name should be changed



Operation of middle box

- Enable logging for protocol translation
  - In case of IPv4 to IPv6 translation, take a port number of the traffic in translator and server side.
- Configure ACL to serve only the scope
    - Don’t make open proxy ☺

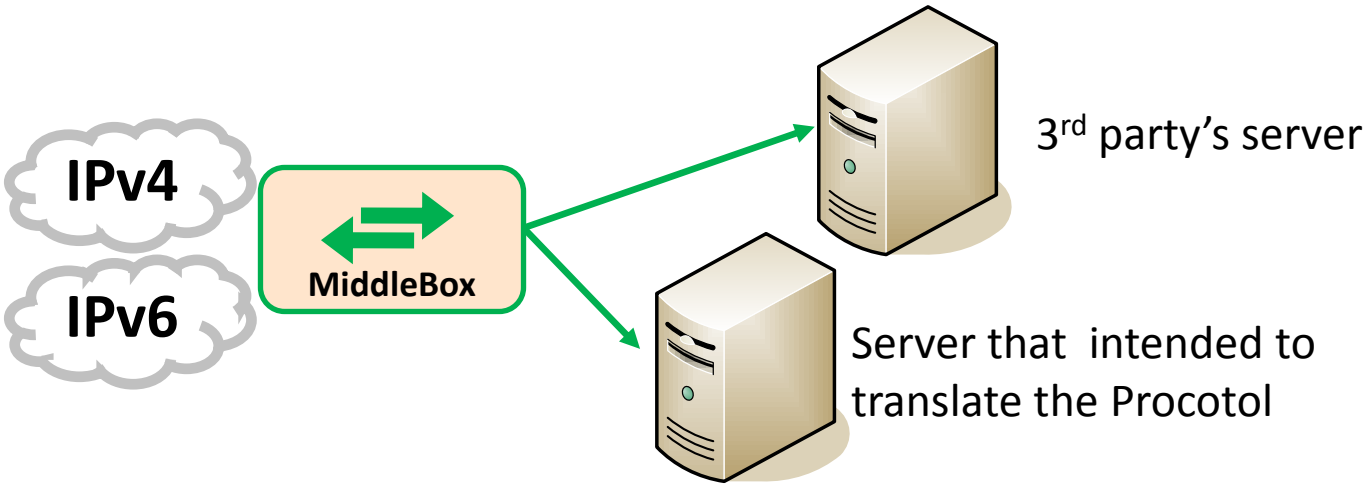
Location of a middlebox	Scope of service	Accessible server via middle box
Client-side (Protocol Translator)	Client’s network only	The Internet
Server-side (Reverse Proxy, Load Balancer, Protocol Translator etc)	The Internet	Servers under middlebox

How to identify the issue

- Connect to middle box from external, try to access a server which is outside the service scope.
  - Same security of reverse proxy is required.

```
$ telnet 2001:db8:beef::1 80
GET http://[$IPV4_ADDRESS_OF_3RD_PARTY_WEB_SERVER]/ HTTP/1.0
```

```
$ telnet 2001:db8:beef::1 80
GET http://[$IPV6_ADDRESS_OF_3RD_PARTY_WEB_SERVER]/ HTTP/1.0
```



Test your firewall

- Running Port scanner from a different network is a useful for testing firewall;

  E.g. Internal and external (Public Internet)

  – If Intended port is found open, need to reconfigure.

  – Don't scan against 3rd party network ☺

- Several Port scanner supports IPv6

  – Nmap http://nmap.org

```
>nmap -6 -sT mail

Starting Nmap 5.00 ( http://nmap.org ) at 2013-04-29 13:55 JST
Interesting ports on mail.example.jp (2001:db8:beef::1):
Not shown: 989 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
25/tcp  open  smtp
53/tcp  open  domain
80/tcp  open  http
110/tcp open  pop3
```

Test you firewall

- If you don’t have a IPv6 access other than own network, you can use online scanner.
  - Tim's Free Online IPv6 Port Scanner (Firewall Tester)
    - http://ipv6.chappell-family.com/ipv6tcptest/
    - Source code is also available
      - https://github.com/timsgit/ipscan
- It will test your firewall from the public internet

Verifying SMTP service

- In some case, IPv6 transport is not used even SMTP server of the recipient has a AAAA record.
- IPv6 email reflector (http://veznat.com/)
  - Test that SMTP delivery is running correctly over IPv6
  - By sending blank e-mail to ipv6@test-ipv6.veznat.com , the results including Received headers, MX record configuration will be retuned.

Subject: IPv6 Email Reflector Results
From: IPv6 Email Reflector <ipv6@test-ipv6.veznat.com>
To: <XXX@example.jp>

Thanks for using the test-ipv6 email reflector.

Your message was received over IPv6.

Here are the Received headers:

Summary

- Packet filtering
  - Some ICMPv6 messages should not be filtered.
    - SPI and Rate-limit are useful for protecting attack by ICMPv6
  - Better to filter Automatic tunnel in dual stack environment.
- Reverse DNS
  - We couldn't rely on Reverse DNS lookup on IPv6
- IPv6 address syntax and format
  - May vary by implementation
    - Don’t forget it when you’re doing “grep” :)
- Middlebox deployment
  - Need to care about source address.