Project Number:	**IST-2001-32603**
Project Title:	**6NET**
CEC Deliverable Number:	**32603/GRNET/DS/351/A1**
Contractual Date of Delivery to the CEC:	December 31st 2002
Title of Deliverable:	Secure IPv6 Operation: Lessons learned from 6NET
Work package contributing to Deliverable:	WP3
Type of Deliverable*:	R-Report
Deliverable Security Class**:	PU-Public
Editors:	G. Koutepas
Contributors:	C. Friacas, G. Koutepas, A. Liakopoulos, J. Mohacsi, E. Rosti, E. Vyncke, D. Zacharopoulos
Reviewers	J. Mohacsi , J. Ferreira

* Type: P - Prototype, R - Report, D - Demonstrator, O - Other

** Security Class: PU- Public, PP – Restricted to other programme participants (including the Commission), RE – Restricted to a group defined by the consortium (including the Commission), CO – Confidential, only for members of the consortium (including the Commission)

Abstract:

This Deliverable describes IPv6 security issues revealed and studied through the 6NET operation. The document, as part of Activity 3.5 (Network Security) examines security measures suitable for the 6NET network infrastructure, and general IPv6 deployments.

Keywords: IPv6, Security, Threats, Best Practices, Firewall

Executive Summary

The main objective of Deliverable 3.5.1 is to provide an overview of the current security issues in an IPv6 based networking environment and suggest a number of helpful security "guidelines". This deliverable tries to identify the problem, i.e. possible security threats, and examines the most suitable counter-measures. It is based on what is known on security from the operation of the 6NET network and the 6NET user community. Actions suggested in this document are derived from tests performed within 6NET and knowledge accumulated during its operation; it is thus a valuable summary of the actual practical experience on IPv6 environments gained from the operation of 6NET.

In the context of WP3, there was considerable effort on networking services (most notably multicast) implementation, testing, and debugging that recently resulted in a stable networking environment. The next logical step was to undertake other activities to complete the functionality of the 6NET infrastructure, such as security with IPv6. The latter is an activity that could be examined and configured effectively only after the network architecture and running services had been finalized.

A number of challenges had to be overcome in order to complete this deliverable. Final results on the operation of 6NET could only be obtained after the main part of the development work was over. Unfortunately, many real life security tests had to be skipped due to the unavailability of testing tools operating in IPv6. Also, no big-scale security events were observed during 6NET's operation that would help to access the effectiveness of some of the measures. However, some of the IPv6 threats were envisaged and corresponding measures were taken during the phases of design and setup of the 6NET infrastructure. The experience gained from its day-to-day operation and the actual technical problems that had to be addressed helped built a solid technical base and a deeper understanding of the main security issues. This document will hopefully provide some help and guidelines to all parties interested in building a secure and functional IPv6 infrastructure.

Table of Contents

1. Introduction

1.1. Usage and Audience of this Document

This document analyzes the security issues for native IPv6 deployment at an enterprise or ISP/NSP network. It draws from the experience gained by the 6NET project that implemented a high speed interconnection network offering a variety of services to the end users.

IPv6 is regarded worldwide as a solution for a number of the networking problems the most important of which being IP address depletion. It also provides additional features that can enhance the network interconnection characteristics and security. Many of these new features however, have been untested in a real networking environment and could present novel challenges and threats.

6NET started as a network for providing core IPv6 connectivity between European academic institutions. Since its successful implementation the focus has shifted from the basic (core) networking services to user connectivity and services. It's these services that provide the real value of the network and usefulness to the end users.

Although in 6NET the assets at risk are academic networks and the operation of experimental services, security problems can escalate to the universities' internal networks, possibly creating problems there, but also may disrupt experiments and other evaluation processes of 6NET itself. Furthermore, since 6NET is also an important feasibility demonstration project, security will guarantee the public's trust in the value and especially reliability of the new technology.

One of the characteristics of 6NET that adds importance to security issues are its high speed network connections. Any high bandwidth line offers a very efficient infrastructure for the malicious users that will manage to utilize it to attack any of its nodes or targets outside. Specifically in IPv6, 6to4 facilities provide the possibility of an attack "spill-over" to more critical IPv4 production and even commercial networks. Policies, methodologies, configurations and the general security lessons learnt from 6NET will, hopefully, prove to be a useful and substantial contribution to the IPv6 community and future deployments.

The target audience for this document is network managers and administrators that control medium to large size networks and want to transition to IPv6, want to understand the threats and requirements of the new environment and want to perform a transition to IPv6 without compromising their security level. This document offers them some guidelines on the security issues they may encounter. Complete and system specific technical instruction however they will get from the 6NET cookbooks. Some prior knowledge of the IPv6 protocol is required.

The document may also be useful to:

- Organization managers that want to get a better understanding of the possible threats and problems that they may encounter when transitioning to IPv6. The document will also be useful for adapting the security policies or developing new ones.
- Members of the general public that want to be informed on the security aspects of the new protocol and the results that the 6NET project has had in the area.

1.2. Positioning of this document relative to other 6NET Deliverables

This document provides an overview of the major security issues and the best security practices for a "6NET like" IPv6 user environment. This network design follows the high-level management guidelines defined in documents D6.3.1 (6NET IPv6 Network Management Cookbook) [9] and D6.3.2 (Final Report on IPv6 management and monitoring architecture, design, tools and operational procedures - Recommendations) [10]. The current document, D3.5.1, however, does not go to a low enough level to provide specific configuration examples and/or code samples. Such detailed accounts are left to the "cookbooks" created by other 6NET actions, more specifically D.3.1.2 (IPv6 cookbook for routing, DNS, intra-domain multicast, and security) [11].

In summary a network administrator reading this document should (a) assess the similarity of his architecture and services to 6NET's, (b) get informed on the threats and issues of the specific configuration by the present document (D.3.5.1) and (c) use the technical guidelines of D.3.5.1 as well as specific technical details, corresponding to his configuration, in the Cookbooks.

2. Security aspects of IPv6

2.1. Intro: New threats in security with IPv6

Most of the people talking about IPv6 security focus on its two main features that provide authentication and encryption, which is provided by IPSec, a mandated feature of IPv6:

- The "Authentication Header", provides authentication and integrity (without confidentiality) to the IPv6 datagrams. It is an extension to the IP header and offers support for many different authentication techniques.
- The "Encapsulating Security Header" provides integrity and confidentiality to the IPv6 datagrams. In fact IPsec was initially proposed as a feature of IPv6 and was transferred to IPv4.

Security in IPv6 however has many more aspects. The new protocol creates a completely new operational environment where existing threats could be automatically mitigated or made worst. And new, unpredicted threats that utilize the protocols unique characteristics could also appear. A quick summary:

- A wide range of addresses makes host and port scanning difficult, but it may also be a very effective in concealing malicious users and their operation within a LAN.

- Intrusion detection systems and especially NIDS have to be adjusted to the new type of IP packets. Although a number of security products already offer this capability, IPv6 poses extra difficulties for any security assessment based on traffic analysis; there is extra CPU overload associated with the analysis of the bigger IPv6 packets; new, special types of packets need to be incorporated in the security policy. Additionally, in many cases IPv4/IPv6 transition mechanisms "conceal" the real IPv6 packets and prevent their examination by routing them though tunnels.

- The Routing Header feature of IPv6, as well as other Extension Header types, can also be used for (indirect) attacks since it may be used to conceal the real destination of an IP datagram.

- Another security issue is created by the automatic IPv6 address configuration schemes. Specifically, automatic address allocation when misconfigured may reveal important assets of the network due to pattern repetition. The same technique may also be initiated by malicious users in order to reserve specific IPs and prevent legitimate address allocation or "hijack" connections.

Currently, the IPv6 protocol is supported on most platforms and operating systems. It often requires just a single command to establish a connection with the IPv6 network via auto-configuration mechanisms without the system administrators being aware. Therefore, IPv6 presents a security factor that has to be considered and carefully planned for in the enterprise network. Users as well as system administrators should not overlook the dangers, which derive from IPv6 connectivity.

2.2. Information Gathering

The big number of potential hosts in a typical IPv6 LAN setting makes host and services identification ("fingerprinting", port scanning) quite difficult if not impossible. A number of issues however could simplify the process easier as well as put important systems in danger:

- The practice of system administrators to use specific, predictable, numbering schemes for important systems (e.g. routers, servers, etc.). Care from the side of the administrators when choosing a numbering pattern for their systems should probably help with this problem.
- The possibility of information gathering on existing systems from poorly secured routers, gateways, DHCPv6 servers[1] or other network devices. This problem is rather a system's security one and the solution do not differ under IPv6: careful and timely security management, ensuring that the system is adequately protected from current threats.
- The need for particular ICMPv6 messages to be allowed in the network for the IPv6 protocol to operate correctly. As in IPv4, these packets can be used for information gathering therefore the security policy should be appropriately adjusted to cope with the new protocol features, allowing through only the necessary types of messages (more in section 2.6.3).
- Some IPv6 multicast addresses are used to group devices of the same type for convenience, e.g. all routers, all NTP servers etc. An attacker able to access these addresses could acquire access to the corresponding devices and perform attacks against them (e.g. DoS). Careful border filtering should prevent the particular addresses from being announced or accessed outside the network's administrative borders.

Finally a practice well known and proven valuable under IPv4, the filtering of unneeded services at the network's access points, can be equally useful under IPv6 for mitigating reconnaissance threats.

[1] DHCPv6 is not necessary for IPv6 address allocation but may be used in the future to provide additional info on services.

2.3. Firewalls

2.3.1. The firewall architectures

The firewalls became in the 90s the building block of each IP network. The recent growth of IPv6 usage needed analysing whether the new protocol can provide enough security without the use of IPSec. This analysis is also important since the application of IPSec on the Internet is relatively scarce and probably will be limited due to deployment difficulties of the public key infrastructure, and in spite the fact that IPSec itself provides a good, modular framework. This section tries to analyse what is available and what is missing for effective IPv6 firewalling.

The Internet firewall is a system that implements and enforces the security policy between two networks: usually protects an internal private network (Intranet) from external Internet threats. Sometime firewalls are also implemented with more than two network interfaces, where the third, fourth interfaces are used for special purposes like DMZs (DeMilitarised Zones), etc.

The firewalls usually can be operated at different levels in the networking hierarchy:

IP level	Packet filtering firewalls
Transport level	Circuit oriented firewalls
Application level	Application level proxies. There is a higher level of support in the application level proxies, e.g. transparent proxies and modularisation

The most important principle of firewalls, however, is helping to enforce the security policy (administrative rules) that will protect certain assets. The majority of modern firewalls employ a mix of protective methods at different levels.

In IPv6 the levels are not changed, therefore we can expect that firewalls should support IPv6 at any level. A good firewall implementation should be IP version agnostic at transport or application levels.

We will focus our discussion to packet filtering firewalls for two reasons.

- These types of firewalls are the basic elements for the more advanced firewalls. They have become necessary components due to the very large number of existing protocols on the Internet (e.g. a wide variety of H.323 related standards, instant messaging protocols, even FTP) that prevents the operation of proxy services for every one of them
- Currently there are only very few application level firewalling solutions available on the market that offer IPv6 capabilities.

2.3.2. Location of the Firewalls

Traditionally the firewalls are installed next to the interconnecting device (usually routers) in order to choke the unwanted traffic as close to the originating point as possible. Nowadays the firewalls (usually more then one at each network) are installed in front of the device or network, which must be protected. What are the implications of enabling IPv6 on these firewalls [18][19]?

- The firewalls should support Neighbour-Discovery and Neighbour-Advertisement ICMPv6 message processing – This issue is rarely discussed with IPv4 firewalls: The IPv4 firewalls must support ARP protocol. The Neighbour Discovery Protocol (RFC 2461) is an extension of ARP for IPv6, therefore IPv6 firewalls must support Neighbor Discovery Protocol filtering "out of the box".
- The IPv6 firewalls should not filter out packets with proper fragmentation header. A common practice in IPv4 firewalls, to guard against the tear-drop attack or other cases of heavily fragmented packets, is to reassemble the IP fragments at the firewalls themselves and send the complete and sanitised resulting packets to the end systems. Unfortunately this is not possible in IPv6, since fragmentation and reassembly can happen only on the originating and destination node. However, some protection which might be possible in IPv6 is discussed later.
- IPv6 firewalls must support extension headers.

The rest of the requirements are depending on the location of the firewall boxes and routers.

Internet-router-firewall-protected network architecture

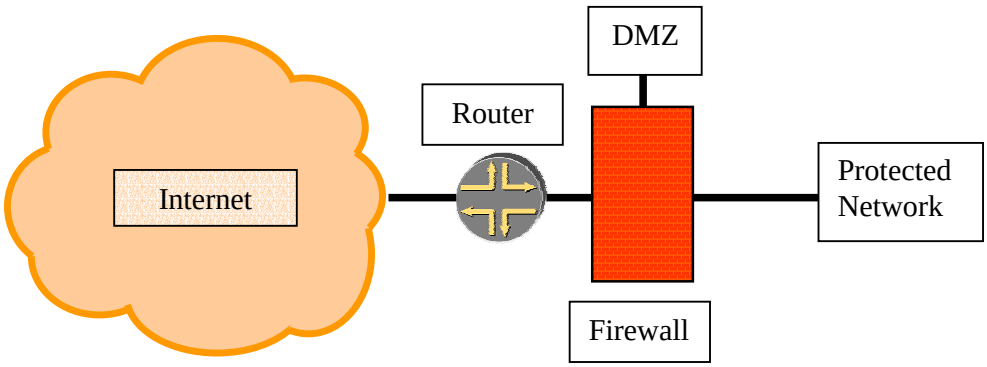

Figure 1: Internet-router-firewall-protected network setup

Additional requirements:

- In this setup the Firewall must support Stateless Address Autoconfiguration mechanisms (RFC 2462) if the Autoconfiguration option is used on the Protected Network. If the Firewall is operated transparently to the IP layer, then it should allow the Router-Solicitation messages coming from hosts and their respective answer coming from the Router. It should also allow periodic Router-Advertisement messages to go from the Router to the Protected Network. If the Firewall is operated non-transparently to the IP Layer, then it should be able to answer Router-Solicitation messages and periodically announce Router-Advertisement messages. These settings are also important if the network is operated with DHCPv6 (or other Statefull Address

Assignment methods), since the Stateless RA messages will inform nodes on the network about the configuration method that is to be followed.

- If IPv6 multicast is implemented in the Protected Network, then the Firewall must support the Multicast Listener Discovery Protocol in order to keep track of the interested nodes in the Protected Network for a particular multicast group.

Internet-firewall-router-protected network architecture

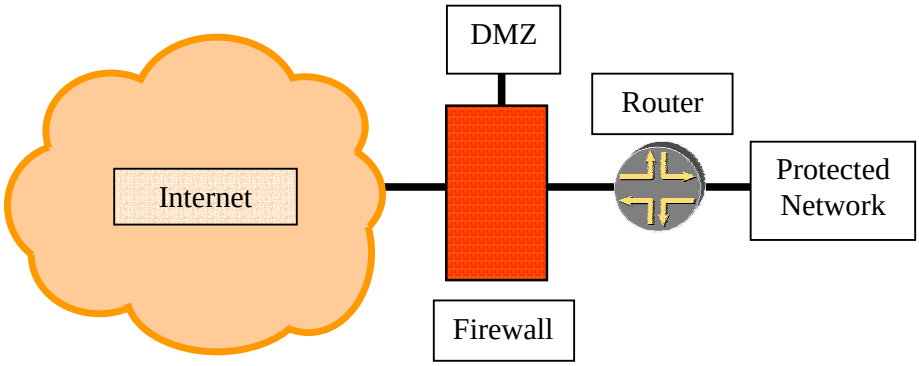

Figure 2: Internet-firewall-router-protected network setup

Additional requirements:

- The Firewall must support the dynamic routing protocol filtering, that is used by the access router (Router) and Internet Service Provider (e.g. OSPFv3, IS-IS, RIPng, or BGP). This might be challenging if IPSec is used for securing the routing protocols. As a general rule we recommend to use either static routing or BGP for such a setup, since BGP is using MD5 hash and TTL hack for securing routing updates that are IP version agnostic.

This setup might be inconvenient, since the Firewall should support a number of different access technologies, therefore it may need to support a wide variety of interfaces. This problem is expected to be less common in the future since many providers prefer handing-over the Internet service over Ethernet media.

Internet-firewall/router(edge device)-protected network architecture

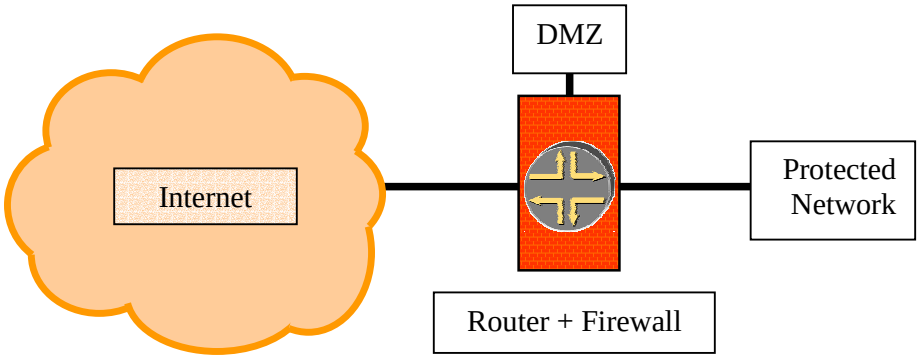

Figure 3: Internet-Edge-protected network setup

Additional requirements:

- Must both support what is necessary for the previous two architecture (Router-Solicitation, Router-Announcement, and Dynamic routing filtering)

This is a rather powerful architecture, since it allows concentrating both the routing and the security policy in one device; however this concentration makes that particular architecture susceptible to the security problems:

- More functionality should be integrated into one device. That makes it more complex and opens the possibility of more security problems
- Since it is only one device the principle of security: protect your network/service with more than one asset, cannot be fulfilled.

This setup is very common in small-office/home-office environments, where a single xDSL, or cable router provides connectivity and in the same time enforces the security policy defined by the network administrator.

2.4. Address usage/assignment in firewalled IPv6 environments

The current operational practice in an IPv4 environment is to somehow keep track of which machine is using which IPv4 address at a certain point in time either by statically allocating IPs (v4) or by using DHCP and keeping lease logs. It is also very common to identify machines in the enterprise network management systems by their MAC addresses. It is thus crucial for the secure and efficient operation of IPv4 networks to log the IP address, MAC address and L2 port combinations. There are also some tools, implemented in L2 switches, to prevent DHCP abuse and ARP poisoning called DHCP snooping and ARP inspection respectively. Let's see what is possible in IPv6, and what countermeasures are possible to prevent abuse. There are different possible ways to assign IPv6 addresses:

2.4.1. Using Stateless Address Autoconfiguration

In this method the globally aggregatable unicast address is derived from the prefix advertised by the routers and the IEEE EUI-64 identifier (RFC 2462). Since the EUI-64 identifier is generated from the MAC address if it was available, then mapping the IP address to a MAC address is very easy to do. Only MAC addresses and L2 port mapping should be implemented. Enforcing the usage of the EUI-64 identifiers as part of IPv6 addresses could be easily enforced by firewalls. Some firewalls already allow checks to MAC address and EUI-64 address consistency of the outgoing packets. This way the accountability of the outgoing communications can be easily provided. However, you should configure carefully your firewall rules if you also use statically assigned addresses.

2.4.2. Using Privacy Extensions for Stateless Address Autoconfiguration

Addresses of this type were developed due to concerns that the same Interface identifier could be used anytime in multiple communication contexts. In this case it becomes possible for that identifier to be used to correlate seemingly unrelated activity. But privacy extended addresses are considered harmful [7] for several reasons:

- They complicate debugging, troubleshooting
- They require frequent updates on the reverse DNS entries
- They allow easier in-prefix address spoofing

- In the current form temporary and forged addresses cannot be distinguished
- They do not improve the prefix privacy

Therefore we do not recommend using privacy extended address as defined in RFC 3041. The updated standard addresses [17] solve some of the problems above. There is also a new IPv6 feature called Cryptographically Generated Addresses (CGA) [8], which generates a random interface identifier based on the public key of the node. The goal of CGA is to prove ownership of an address and to prevent spoofing and stealing of existing IPv6 addresses.

To prevent using RFC3041 type of addresses you can use the filtering technique described in the previous section.

2.4.3. Using DHCPv6

DHCPv6 is the "statefull address autoconfiguration protocol" and the "statefull autoconfiguration protocol" referred to in "IPv6 Stateless Address Autoconfiguration" (RFC2461).

DHCP can provide a device with addresses assigned by a DHCP server and other configuration information, which are carried in options.

DHCPv6 (RFC 3315) servers use DUIDs (DHCP Unique Identifier) to identify clients for the selection of configuration parameters and in association with IA clients (Identity association - a collection of addresses assigned to a client). DHCP clients use DUIDs to identify a server in messages where a server needs to be identified.

The DUID can be generated from several different sources:
1. DUID Based on Link-layer Address Plus Time [DUID-LLT]
2. DUID Assigned by Vendor Based on Enterprise Number [DUID-EN]
3. DUID Based on Link-layer Address [DUID-LL]

In the case of DUID-LLT and DUID-LL, the association between the IPv6 address and Link-layer address (usually MAC) still exist in the state information of the DHCPv6 server, so accountability is still possible. In the case of DUID-EN it is the responsibility of the administrator to build such a pairing.

If the addresses are assigned from a well identifiable sub-range in /64, the firewalls can ensure that only hosts using DHCPv6 for address configuration can connect outside of the protected network.

Unfortunately, currently there is no similar technique available on the market that will allow only real DHCPv6 servers to assign addresses to the requester hosts.

2.4.4. Static Address Assignment

The static address assignment is very similar to IPv4 static assignment therefore similar pitfalls might possible if it is used.

2.4.5. Prevention techniques

A technique similar to the one that prevents ARP cache poisoning (in IPv6 ND cache poisoning) is possible but it requires DHCPv6 snooping. Firewalls can enforce the DHCPv6 usage and make the DHCPv6 address assignment the default method, thus making DHCPv6 snooping easier to implement. Currently no DHCPv6 snooping support is available for any networking device.

IPv6 can provide an option to prevent ND cache poisoning in the case of stateless autoconfiguration via snooping the Neighbour Solicitation and Neighbour Advertisement messages: Neighbour Solicitation messages contain an informational pair [source_IPv6, source_MAC] that can be stored, while Neighbour Advertisement messages contain two informational pairs: [source_IPv6, source_MAC] and [destination_IPv6, destination_MAC] which can be also stored. Any case of a mismatch can be diagnosed from the previously stored ND entry and the switch can disable the abusing port. A "light version" of the above protocol can be implemented in Firewalls: detect and report ND entry changes i.e. different IP address with same MAC address etc.

2.5. IPv6 Extension Headers and Fragmentation Issues (IPv6 extension header processing)

In IPv6 the IPv4 options are replaced by extension headers. It is common practice in IPv4, to drop IPv4 packets which contain IPv4 options. Generally source routing is considered harmful since many times it is used to attack end-systems and hide the true attacker. In contrast, in IPv6, the extension headers DO have legitimate usage.

2.5.1. Hop-by-hop option extension headers

- This extension header is used for padding, "jumbograms" and router-alert options (including MLD (RFC-2710) or RSVP messages (RFC- 2205)). It is recommended that you should allow through the firewalls of the hop-by-hop options if you need them for certain operations like multicast MLD join messages. Otherwise it is recommended to log and discard such packets.

2.5.2. Routing option extension headers

This extension header implements, what is called in IPv4 "loose source routing". The general recommendation, if you are not supporting Mobile IPv6, is that you should log and discard packets containing routing options headers. In the case that you do support Mobile IPv6 you must enable routing options to Mobile IPv6 Home Agents.

2.5.3. Destination option extension headers

This extension header is used for padding, so it is recommended to log and discard such packets

2.5.4. Fragmentation option extension headers

Fragmentation and reassembly is only allowed at the originating and destination nodes in IPv6. Intermediate nodes (e.g. Routers, firewalls) are forbidden to initiate fragmentation or undertake reassembly procedures. Firewalls, however, can make some sanity checking of packets with fragmentation extension headers.

1. All but the last fragment should be bigger than the minimal MTU size defined by IPv6 (RFC 2460) to be 1280 octets.
2. The payload size of all but the last fragment of the message should be a multitude of 8 bytes, since the fragments are generated by 8 byte boundaries.
3. The zero offset fragments should not be last. In the case that they are last, the originator did not have to apply fragmentation: either there is a wrong host implementation or the result of an attack.

4. You might drop fragment packets that are destined to routers, firewalls, or other internetworking devices.
5. The number and content of the headers preceding the Fragmentation header of different fragments of the same original packet are allowed to differ. Only those headers in the Offset zero fragment packet are retained in the reassembled packet. This can be problematic since an attacker might generate some extra header to confuse the firewalls and end hosts.
6. The Next Header values in the Fragment headers of different fragments of the same original packet may differ. Only the value from the Offset zero fragment packet is used for reassembly. This can be problematical since attacker might generate some extra header to confuse the firewalls and end hosts.
7. The Identification must be different than the one used on any other fragmented packets sent recently with the same Source Address and Destination Address. The Firewall might implement some clever algorithms to rate limit the fragmented packets sent from a certain source if it is considered harmful. Currently there is no exploiting tool available to perform an IPv6 fragmentation attack, but it is worth mitigating the possibilities.

For better fragment identification Firewalls might implement more stringent rules by requiring the same number and content for all the preceding headers and same Next-Header value. However this requires further investigation. (See rules 5. and 6.)

Header chaining and fragmentation can possibly obstruct the determination of the transport layer from the first fragment (due to excessive header chaining). The firewall should be able to detect such an extreme situation. This case is fundamentally different from IPv4. An Intrusion detection system should also be aware of such a situation.

2.5.5. Authentication header

This header should be processed according to the Security policy

2.5.6. Encapsulated Security Payload header

This header should be processed according to the Security policy

2.6. Filtering

2.6.1. Ingress and Egress filtering

Most of the occurrences of various Denial of Service (DoS) attacks which have employed forged or spoofed source addresses have proven to be a troublesome issue for Internet Service Providers and the Internet community overall. RFC 2827 recommends a simple, effective, and straightforward method for using ingress traffic filtering to prohibit DoS attacks which use forged IP addresses to be propagated from 'behind' an Internet Service Provider's (ISP) aggregation point. The method, called "ingress filtering" can only prevent spoofing of the source address. An important benefit of implementing ingress filtering is that it enables the originator to be easily traced to its true source, since the attacker would have to use a valid, and legitimately reachable, source address.

The ingress filtering is usually implemented at ISP edge routers with various methods, either via firewall filters or by enforcing the uRPF (unicast reverse path forwarding) check. The behaviour of the ingress filtering is the following:

```
 IF     (packet's source address is from network residing behind the interface
where the packet comes from) {
    forward as appropriate
 } ELSE IF    (packet's source address is anything else) {
    deny packet
}
```

A similar technique can be implemented by the end-user of an ISP to prevent sending packets that do not belong to their network, usually called egress filtering.

These techniques can also be implemented in IPv6. IPv6 can make the ingress filtering easier, since only one prefix should be configured for the ingress filter, due to the hierarchical aggregation of IPv6 addresses. Usually only one /48 has to be configured, if you cannot setup automatically the antispoofing or uRPF (unicast Reverse Path Forwarding) check.

The egress filtering configuration is very similar to ingress filter configuration, the difference being that it is configured at the user's equipment.

We should note, that ingress and egress filtering might be more complex, albeit not impossible if multihoming and multiple prefixes are employed at the user site. In this case the multiple address prefix should be appropriately configured.

2.6.2. Address filtering in a firewalled IPv6 environment

In the IPv4 firewalls it is a common practice to somehow hide the protected network with the use of private addresses (RFC1918) and delegate the dirty task of address translation to firewalls. This is considered a solution for many problems which is not quite correct. The weakness of filtering is that it cannot be hidden by a network address translator. In IPv4 a very similar architecture can be build without using any private addresses and hiding the internal protected network by split DNS.

With IPv6 it's no longer necessary to use NAT, since the end-sites have plenty of subnets and addresses to be assigned to the nodes. Furthermore private addresses no longer exist as a concept therefore network administrators should configure their firewalls along this principle. They have to be aware, that potentially each device connected to the network can be reachable from outside. Of course, this does not cause big problems since the usual default firewall rules should deny any connection at all. It should be considered as a good security practice to deny everything on each IPv6 firewall, whenever is enabled for first time (an exception might be to enable the neighbour-discovery protocols). Let's consider the security implications of the different type of addresses that might be used for communication in IPv6.

2.6.2.1 Globally Aggregatable Unicast Addresses

This type of address is used for global communication, therefore you should filter them (allow or deny a particular address) according to your security policy. You should start as discussed earlier with rules denying everything rules and enable only the necessary communications to specific global addresses without configuring any address translation. Most of the time you should enable IPv6 packet forwarding on the IPv6 firewalls.

2.6.2.2 Link-local addresses

The link local address is used only in communications "on-link", since it only has meaning on a certain link. You should assess, if you need to setup link local filtering.

Let's consider you have a malicious host connected to your link. This host can scan or attack other machines on the same link or subnet, using the link-local address as a destination. The firewall cannot help in this situation, since it only acts as a gateway to other links or subnets. The switch where the devices are connected may prevent malicious host scanning on the IPv6 link.

If you can expect malicious hosts inside your subnet you should setup "personal" or "host-based" firewalls on each and every host on the subnet, and then you can setup filtering of the link-local-addresses. This problem is not new; we can see similar cases in the IPv4 Internet. But we have to emphasise that a personal" or "host-based" firewall should be able to filter link-local addresses too.

The usual firewalls however should also work properly in terms of link-local addresses: they must not forward packets having as destination a link-local address. Additionally, a firewall should not forward packets with a link-local address as its source, since it has no meaning on the other subnet or link. However, you should not disable completely the processing of packets with link-local addresses, since it is required by the neighbour-discovery process and the autoconfiguration procedure.

2.6.2.3 Site-local addresses

The site-local addresses are deprecated in RFC 3879. Router and firewall implementations SHOULD be configured to prevent routing of a site's local addresses and packets with this prefix by default.

2.6.2.4 Unique Local IPv6 Unicast Addresses

The Unique Local IPv6 Unicast Address Format (FC00::/8 and FD00::/8) [6] defines globally unique addresses which are intended for local communications, usually inside of a site. They are not expected to be routable on the global Internet. The recommendation concerning these addresses depends on the location of the firewall.

If the firewall is protecting the site from the outside Internet, then you should simply prevent Unique Local IPv6 Unicast Addresses from traversing the firewall.

If the firewall is installed inside the site, then you should follow the security policy guide and apply similar rules, with those used for globally aggregatable addresses. You may use different rules if you have less stringent or different policies for the communications in the internal part of the site.

2.6.2.5 Compatible addresses

IPv6 compatible IPv4 addresses (for auto-tunnelling) (::0.0.0.0/96) are no longer used so they should not appear on any links and must be filtered.

IPv6 mapped IPv4 addresses (which are used to determine the IPv4 transport) (::ffff:0.0.0.0/96) should not appear on any links except in the case of some transition methods (6PE – implementation dependent, SIIT). They should be filtered.

2.6.2.6 Multicast addresses

Multicast addresses cannot appear as the source of any packet. Therefore any packet that has a multicast address as source address can be safely dropped.

If you don't support multicast you may consider filtering out multicast packets at the site exit firewalls. Note that internal communication might require multicast for OSPFv3, DHCPv6 or NTP.

2.6.3. ICMP Filtering

It is very common (although questionable) practice to filter completely the ICMP messages in Ipv4. This is no longer possible with IPv6. As the name that it stands for suggests, Internet Control Message Protocol for IPv6 (RFC 2463) is the control and foundation protocol for the operation of IPv6, not an auxiliary protocol that can be easily omitted. Our recommendation is the following:

ICMPv6-echo-request and reply (Types 128 and 129):

- You should consider enabling at least outgoing ICMPv6-echo-request and their answers, the ICMPv6-echo-reply packets to facilitate debugging. Of course, it is wise to rate limit ICMPv6 debugging packets to a certain level.
- You may consider enable incoming ICMPv6-echo-request packets and their answers to your well know IPv6 service machines. You should be sure, however that your IPv6 service machine can handle ICMPv6 requests over a certain rate. Of course, it is wise to rate limit ICMPv6 debugging packets to a certain level.

ICMPv6-destination-unreachable (Type 1):

- You should consider enabling incoming ICMPv6 destination unreachable messages as answers, to outgoing IPv6 packets that have been sent for debugging purposes.
- You may generate proper ICMPv6 destination unreachable messages for all filtered packets. This is useful for debugging. It is a common practice in IPv4, to refrain from generating ICMPv6-destination-unreachable messages to hide the networking/service structure. You can apply the same rule to IPv6. If you generate ICMPv6-destination-unreachable messages, however, do it properly, setting the right reason code: no route to destination, administratively prohibited, beyond scope of source address, address unreachable, port unreachable.

ICMPv6 packet too big (Type 2):

- You must enable incoming ICMPv6-packet-too-big messages as answers to outgoing IPv6 packets for the Path-MTU-discovery to operate properly.
- You must generate ICMPv6-packet-too-big messages properly if your MTU is different anywhere within your network from the MTU on the link between you and your provider. So be prepared, to forward ICMPv6-packet-too-big messages at the firewall.

ICMPv6-time-exceeded (Type 3)

- You must/should enable incoming ICMPv6-time-exceeded messages to be able discover destination systems not reachable due to a low TTL value in the outgoing packets.
- You must generate correct ICMPv6-time-exceeded messages since they are essential for proper operation of Internet.

ICMPv6-parameter-problem (Type 4):

- You should consider enabling incoming ICMPv6-parameter-problem messages as answers to outgoing IPv6 packets for debugging purpose.
- You must generate correct ICMPv6-parameter-problem messages since they are essential for proper operation of Internet.

ICMPv6-Neighbour-Solicitation and Neighbour-Advertisement (Type 135 and 136):

- You must enable incoming and outgoing ICMPv6 Neighbour-Solicitation, Neighbour-Advertisement packets, with proper link-local addresses or multicast addresses for the Neighbour

Discovery function to operate properly.

ICMPv6-Router-Solicitation and Router-Advertisement (Type 133 and 134):

- If the Stateless Address Autoconfiguration function is used, you must enable outgoing ICMPv6 Router-Advertisement packets, with proper link-local addresses and multicast addresses (All node multicast addresses should be ff02::1).
- If the Stateless Address Autoconfiguration function is used, you must enable incoming ICMPv6-Router-Solicitation packets, with proper link-local addresses and multicast addresses (All router multicast addresses should be ff02::2).

ICMPv6-redirect (Type 137)

- You may disallow ICMPv6-router-redirect messages passing, if you have only one exit router. However, router redundancy might be implemented by router-redirect. It is important to know that redirect has link-local meaning only.

ICMPv6 MLD listener query, listener report and listener done (Type 130, 131 and 132):

- You should enable incoming and outgoing ICMPv6 MLD messages, with proper link-local addresses or multicast addresses if you want to use IPv6 multicast on a bigger scope than link-local. This is required if the "internet-router-firewall-protected network" architecture is used. In this case your firewall should act as an MLD router.

ICMPv6-renumbering (Type 138)

- You may disallow ICMPv6 router renumbering messages passing, since router renumbering is not widely adopted.

ICMPv6 node information query and reply (Type 139 and 140)

- You may disallow ICMPv6 node information query and reply processing, since node information query/reply is not widely adopted.

We summarise the ICMPv6 recommendations in the following table [19]:

ICMPv6	***Usage***
Echo request/reply	Debugging
Destination unreachable	Debugging – better indicators
TTL Exceeded	Error report
Parameter problem	Error report
NS/NA	Important for IPv6 operation - except if you use Static ND entry
RS/RA	For Stateless Address Autoconfiguration
Packet too big	Important for PATH MTU discovery
MLD messages	Required for Multicast operations

Note: Each IPv6 specific feature is marked with Blue, Each required feature marked with Red.

2.7. Fake router advertisements

Router Advertisements are one of the well-known differences between IPv6 and IPv4. IPv4's common method to supply an address for a (default) gateway is either through DHCP or static configuration. In IPv6, geographic network routers that are connected to the same link may use the Neighbour Discovery protocol for a variety of purposes, such as discover each other's presence, determine each other's link-layer addresses, learn parameter values necessary for communicating and exchange information about prefixes they know about. However, such mechanism has a cost in terms of risk from the security viewpoint. The potential range of attacks that one could make taking the place of a network segment's default gateway is considerable.

Routers consider authoritative the information carried in router advertisements sent by other on-link routers, even though such information is not cryptographically secured (e.g., digitally signed or key-MACed or encrypted). Therefore, routers update the affected communication parameters accordingly, without any verification. In the absence of any verification of the received information, malicious nodes may inject bogus values for optional fields of the ICMPv6 extension header, such as the advertised prefix, link layer address or MTU. Since legal router advertisements do not necessarily carry values for all of the possible options defined by the actual state of the Neighbour Discovery protocol, there are good chances that option values proposed by malicious router advertisements are not be corrected by successive legitimate router advertisements. As an example, if a malicious router advertisement announces a maximum transfer unit of 17 bytes and legal router advertisements do not specify the MTU option, the MTU value will remain 17 until a later router advertisement, either legal or fake, announces a different value.

A similar case applies to fields such as current hop limit and reachable time, which can be exploited since they allow the sender to leave their value "unspecified". In this case the receiver continues using its current values for those parameters. Thus, if the current value had been set via a fake router advertisement message, followed by a sequence of legitimate router advertisements that did not specify any value for the parameters, the bogus values would be used continuously until an explicit change occurs, if ever. Since parameters such as retransmission time, current hop limit and reachable time are seldom changed once they have been set, so an attacker can easily poison the network. IPv6 Service could thus degrade (by generation of extra-hops) or become inaccessible. Network administrators should be aware of this phenomenon, avoiding configurations where such advertisements are configured by default. The use of DHCPv6 systems may also assist in preventing such rogue configurations. While this problem doesn't represent a major threat, it can reduce end-user confidence about IPv6 services.

When a "fake router" starts to divert traffic, it will probably operate as an "evil proxy", modifying contents of outbound packets, or acting as the end-node on a communication stream. These two types of attacks can be mitigated using the IPSec protocol, whenever possible. Without knowing the keys of a specific end-to-end communication, there is no point in diverting it or intercepting it, except for DoS purposes.

But IPSec may not be an option if one end of the communication is not known in advance, if there are a big number of peers, or they are located in a different management domain. Once again, using DHCPv6 may provide the extra level of control needed to reduce advertisement problems.

A possible counter-measure that system/network administrators can deploy could be a mechanism that queries ff02::2 constantly in order to identify any "alien router" on the network segment. This type of solution is not an ideal one because it can only warn about an anomaly, not really being able to prevent or correct it. But correctly diagnosing a problem is half way to solve it.

Another (weak) solution would be to set up the "real router(s)'" advertisements settings in such a way that they force themselves as preferred paths on the end-nodes. However, any serious attempt intended to hack a network segment will certainly have this possibility also embedded in its design.

Of course, all the types of attacks (hijacking, DoS, DDoS, etc.) using fake router advertisements will only be possible after an intruder compromises one node on the same segment their other targets are located.

The results of attacks aimed at disrupting the network operations by announcing bogus information via fake router advertisement messages are presented in the Appendix of this document. In particular, there was analysis of attacks on the following parameters:

a) Prefixes
b) MTU
c) Current hop limit
d) Link layer address
e) Router lifetime

As inconsistencies in router advertisements are logged passively, in some cases the attack may not be blocked immediately, thus leading to a Denial of Service.

2.8. Connection hijacking (man-in-the-middle)

When deploying an IPv6 network, whether we think of it in terms of a server/services network, or in terms of a client/end-user network, the well-known connection hijacking issue is not in any way reduced or increased by having larger addresses, and a larger address space to play with.

If we keep in mind that IPSec is strongly tied to IPv6's birth, its usage alone would be enough to avoid any problems regarding connection hijacking attempts. Unfortunately the dominant practice we see today in terms of IPv6 applications already in place is that they don't make any use of IPSec. The use of certificates can also provide the needed end-to-end authentication at the application level (e.g. Web servers). Without such end-to-end security mechanisms, a man-in-the-middle hijack is a possibility.

DNS also plays an important role in redirecting traffic to false servers. Thus, special care should be taken regarding the security of DNS servers and caches. DNSSEC is one solution for this. The corresponding standard was revised in 2003 and tested with success over IPv6, confirming its independence from the IP version used. It is expected that this new standard will be soon adopted by the community at large.

Following the present IPv4 networks' paradigm, firewalls will also have an important role on IPv6 networks' security in the coming years. It is important to stress out however, that simple firewall implementations will not guarantee strong defences against anyone taking over connections from legitimate network activity/users.

One of the key new benefits that IPv6 can bring to the general use Internet is *multihoming*. With this method service is guaranteed to an end-user by more than one Internet Service Provider. At this point *multihoming* is still a work in progress, nevertheless we can already draw some conclusions about its usage. *Multihomed* sites will use multiple addresses. If the transport layer permits an address change after a connection is established, one of the two communicating parts could mislead to receive a new, fake, address as a valid one for its peer. Existing connections may be thus redirected to wrong destinations (to places defined by the malicious user who sent the address change in the first place). In today's Internet the same problem exists on a different layer of the communication (rewriting URLs). There are already some solutions for this problem on those layers (mainly using cookie related mechanisms) that could/should also be used to mitigate the issue at the transport layer too. There is also the possibility of using STCP in the context of some applications, as described in [4].

Concerning available tools under IPv6, we are increasingly observing diagnostic and hacking tools being available under the new protocol. Ethereal [5], a tool already widely used by system/network administrators in various platforms and networks is such an example. Other similar tools sporting IPv6 capabilities and not limited to "passive sniffing" of the physical layer, are soon expected to appear, if they do not exist already. The conclusion is that this type of attack presents a real threat for services over IPv6. The operational risks should be carefully examined before deployment and appropriate protective measures should be taken.

2.9. Level-2 Attacks

Most of the previous attacks were targeting the network layer. It must not be forgotten that the network layer relies on services provided by the data link layer. Any security vulnerability in the data link layer could be exploited by attackers.

ARP spoofing is a well documented attack for IPv4 which obviously is not applicable to IPv6. Playing tricks with ND can lead to similar effects for IPv6.

There are other attacks against data link layer, especially on Ethernet:

- Bypassing the IEEE 802.1q VLAN tag to access a different VLAN (like an out-of-band management VLAN)

- Fooling the learning bridge mode of IEEE 802.1d in order to flood all Ethernet frames on all bridge/switch interfaces, an easy way to be able to sniff some frames even if the sniffing entity is neither the source nor the recipient of the Ethernet frame

- Fooling around the Spanning Tree Protocol of IEEE 802.1d in order to change a topology or run a denial of services against bridges (could lead to traffic disruption)

- Fooling with other layer 2 protocols like Cisco CDP (Cisco Discovery Protocol), etc.

This section will describe those attacks in more detail and will also provide some counter-measures.

This section does not address layer 2 related attacks like attacks against Neighbour Discovery or DHCP.

2.9.1. Bypassing IEEE 802.1q VLAN Isolation

Description

An attacker host in the red VLAN can send a frame to a victim host in the blue VLAN without passing by a network device. This is only applicable when there are at least two switches between the attacker and the victim hosts. It also requires that the attacker VLAN is the native VLAN of the trunk between the switches.

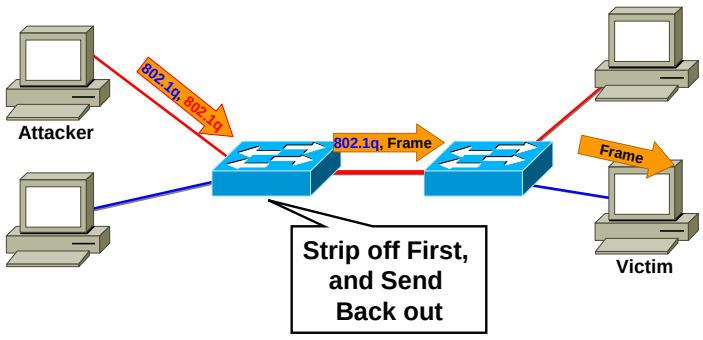

Figure 4: Specially crafted frames passing to the victim

As described on the picture above, the attacker needs to craft a specific frame with 2 IEEE 802.1q labels, the external one being the normal red VLAN tag and the internal one being the victim blue VLAN tag.

The first switch will only look in the first tag and:

- Will accept the frame since it is tagged with the right VLAN tag
- Will forward/flood the frame to the trunk
- Finally, it will remove the red VLAN tag since the frame is forwarded on an interface whose native VLAN is red (native VLAN means that all frames without IEEE 802.1q tag are assumed to be in the red VLAN)

The second switch will see a frame with an IEEE 802.1q tag whose value is blue and will forward this frame to the victim in the blue VLAN since the switch trusts all tags coming from the first switch. Before sending the frame to the blue victim (connected to a port whose native VLAN is blue), it will remove the blue VLAN tag.

Issues

Issues are related to the possibility of an attacker sending frames to a victim in another VLAN. As VLAN are a convenient way to isolate different security domains, this is a security vulnerability (even if the attacker cannot receive any frame back from the victim):

- The attacker could run a DoS attack (like SYN flooding);

- The attacker could run a one way exploit against the victim in order to inject a Trojan, a virus, etc. E.g. the MS-SQL slammer worm required a single UDP packet to break into a MS-SQL server.

Counter-Measures

The easiest counter-measure is to ensure by **administrative action** that the native VLAN of all trunks are not used for hosts.

NB: some Cisco switches can also block all IEEE 802.1q packets coming from one host (this obviously prevents the attack but also prevents any layer 2 QoS) and can also enforce to send an IEEE 802.1q tag even when transmitting a frame belonging to the native VLAN.

2.9.2. Fooling IEEE 802.1d Learning Bridge

Description

This attack is also known as a CAM overflow or MAC flooding.

An Ethernet switch (as well as Ethernet bridges) is called a '*learning bridge'*. This means that the switch dynamically learns on which port is a specific MAC address. It learns the binding <port, MAC address> by looking at the source MAC address of all received frames. The bindings are usually implemented in hardware through the help of a device called CAM (content addressable memory) for speed.

When a switch receives a frame it looks in the CAM to see on which port is located the destination MAC address and only forwards the frame to the right port. If the destination MAC is unknown or is broadcast, the frame is flooded to all ports in the same VLAN.

The attack is pretty crude: the attacker sends enough MAC frames with a random source MAC address in order to fill completely the CAM table with bogus information. The switch has then no more free CAM entries in order to install new bindings for valid MAC address. This translates into the flooding of each and every valid frame.

Issues

There are two issues:

- Denial of services: as all frames will be flooded, the switch actually performs like a repeater/hub which could lead to a severe performance degradation

- Lack of confidentiality: it is usually wrongly assumed that it is impossible to sniff on a switch as usually a port received frames only for hosts directly connected to it. This is not the case when the CAM is overflown… All hosts in a VLAN will receive all frames sent to any host in this VLAN.

The attack is even worse since the CAM is usually shared by all VLAN, so, an attacker in VLAN red will overflow the CAM and all VLAN (blue, green, …) will start flooding.

Counter-Measures

The counter-measure is a **technical** measure. Some switches (notably Cisco Catalyst) have a feature called *port security* which is a means to enforce a specific number of source MAC address for all frames received on this port. There is no performance hit when enabling port security.

If a port configured with a maximum of 1 MAC address, the CAM will insert only the first MAC address in the CAM and will simply ignore all other MAC addresses (either by dropping the offending frame or by disabling the offending port).

The usual recommendation is to configure a maximum of 2 MAC addresses to allow for a MAC address change (like replacing a network device or for roaming laptops).

Note: recent switches are also never overwriting an existing CAM entry, a <port, MAC address> binding is only removed when the switch did not see any traffic for a specific period (typically 5 minutes). This means that routers and other verbose hosts (typically those running a server or running Microsoft Windows) will always stay in the CAM and the attacker cannot erase their bindings.

Note2: when the spanning tree protocol, SPT, detects a change of topology (see next attack), a side effect is to completely erase the CAM (normal since the topology changed). This makes a combined attack CAM+SPT very dangerous. The port security is the only good way to handle CAM overflow.

2.9.3. Fooling IEEE 802.1d Spanning Tree Protocol

Description

The IEEE 802.1d Spanning Tree Protocol, SPT, is used to build a forwarding topology without loop. It is a kind of substitute for the lack of ‘routing protocols’ in an Ethernet environment. Loop avoidance is critical since Ethernet frame can loop forever (there is no equivalent to the hop count field).

The SPT is essentially based on an election process to get the root switch followed by a tree building and ended by putting some links in either forwarding or blocking states.

SPT PDU are transported in Bridge PDU, BPDU. BPDU has not been designed to allow for authentication (unlike BGP, OSPF, …) this leads the path to interesting attacks as any host can send BPDU which will be accepted by all switches.

Issues

There are multiple:

- A specific BPDU called Topology Change Notification will cause both a re-election process (typically done by CPU => risk of denial of service) and will clear the CAM (increasing the exposure to CAM overflow attacks);

- By fooling with bridge priority (used in the root election process), the forwarding topology can be changed by the attacker. This could lead to denial of services (sub-optimal use of the Ethernet infrastructure) as well as lack of confidentiality (all traffic could be rerouted through a host running a sniffer).

Counter-Measures

Little can be done to protect a switch from attacks launched from another switch. This is a case where **physical security** of the switches is important as well as good security practices for switch management.

On the other side, it is usually trivial to block all BPDU frames received on ports where there should be no switch (like a port connected to a DNS server). This is a **technical measure** which should be implemented on all non trunking ports.

2.9.4. Fooling other Layer 2 Protocols

Description

There are a couple of less critical layer 2 protocols, most of them are ancillary. Either standardized like power over Ethernet, IEEE 802.1x user/device authentication, Cisco own Cisco Discovery Protocol, etc.

Issues

Most of these protocols (except 802.1x) are not authenticated and can be spoofed.

Issues are:

- Some denial of services against the switch CPU, like for CDP or 80.1x

- Some physical denial of services by claiming a lot of electrical power (so that other devices cannot get enough power)

Counter-Measures

The easiest way to prevent attacks is to enable those protocols only when there is a real requirement.

IEEE 802.1x should be used either for mutual authentication of switches or at access layer switches to authenticate end users.

CDP should never be implemented towards un-trusted devices (like end user hosts or switches/routers in another security domain).

Power over Ethernet should only be enabled on ports requiring it. This should not be the case for 6NET.

2.9.5. Final Words on Layer 2 Attacks

Layer 2 attacks are real; there exist multiple tools (from the old *dsniff* to *ettercap*). Attacks have been seen in some enterprises as well as in some Service Provider networks. These attacks apply independently to the layer 3 protocol, so, it is also a concern for IPv6

The good news is:

- They can only launched from a layer 2 adjacent node

- There are administrative or technical counter-measures to prevent all those attacks. Even better, the counter-measures have little to no performance impact.

2.10. The DNS Service

DNS service is a critical part of a network operational infrastructure. The version information (version directive in the options section of named.conf -- in BIND) should be made stealth. This may prevent revealing the version information that will allow potential attackers to exploit well-known bugs of older software versions. Also allowing zone transfers from unidentified places should be considered a bad practice.

Initially, 6NET used IP based authentication for managing zone transfers for the 6NET.org and sixnet.org zones. Slave name servers, for example, could get updated zone information only if their IPv6 address matched the addresses stated within the master DNS configuration files (in the allow-transfer section). Currently zone transfers use symmetric keys for authentication and authorization, namely TSIG authentication (RFC2845). This improved significantly the overall security of the zone transfer procedure. The exchange of the keys was done securely using encryption provided by pgp.

The next security step is the implementation of DNSSec which is described in RFC 2535. As a minimum, DNSSEC will be applied to the inverse address mapping of the 6NET address space 2001:0798::/40. If time permits, this will be extended to the 6NET.org zone, which is more interesting as it contains more non-trivial delegations (i.e. delegations to different organizations). In order for these features to work, BIND must be compiled with ssl support (the option is `configure --with-openssl`).

It is preferred for security reasons, to keep the master name server of a zone outside the NS records (stealth) even though it is an "official" server for the zone, and keep it inaccessible. Zone transfers should be allowed only between the master and the other "official" slave Name Servers that are listed in the NS records of the zone. This type of configuration is called "hidden primary" configuration.

2.11. Multicast

UCL's Mbone tools are commonly used for multicasting purposes over IPv6. The security features that exist in the IPv4 version of these tools are also supported in the IPv6 version. There is the authentication feature where a session should not be public and can only be accessed with an authentication password that is set by the session creator. The password can be transmitted securely to the participating party using PGP. In addition there is the encryption feature; a session's entire traffic is encrypted before transmitted. The encryption algorithm is usually DES (or PGP as described in the SDR program). A session created either with authentication or encryption is called "Private Session". It must be noted that the multicast protocol does not provide any security mechanisms of it own as it is used for experimental purposes. The security features described above, involve the applications using the underlying multicast protocol and not the actual protocol.

2.12. Mobility issues

Issues concerning Authentication, Security and Privacy of IPv6 mobility are addressed in the Deliverable D4.2.2v2 (Framework for the Support of IPv6 Wireless LANs ver.2) [15]

2.13. Transition Mechanisms

Security issues of IPv6 transition mechanisms are addressed in the Deliverable D6.2.2 (Operational procedures for secured management with transition mechanisms) [16]

3. Bibliography

[1] S. Convery, D Miller, IPv6 and IPv4 Threat Comparison and Best-Practice Evaluation (v1.0)", presentation at the 17th NANOG, May 24, 2004

[2] Internet Security Systems, "Security Implications of IPv6", white paper, http://documents.iss.net/whitepapers/IPv6.pdf

[3] Eric Marin, "IPv6 Security", presentation at 6NET Zagreb, May '03

[4] http://research.microsoft.com/users/tuomaura/Publications/aura-nikander-camarillo-ssp04.pdf

[5] Ethereal tool web pages, http://www.ethereal.com

[6] R. Hinden, B. Haberman, Centrally Assigned Unique Local IPv6 Unicast Addresses, work-in-progress, June 23, 2004, <draft-ietf-ipv6-ula-central-00.txt>

[7] F. Dupont, P. Savola, RFC 3041 Considered Harmful, work-in-progress, June 2004, <draft-dupont-ipv6-rfc3041harmful-05.txt>

[8] T. Aura, Cryptographically Generated Addresses (CGA), work-in-progress, April 16, 2004, <draft-ietf-send-cga-06.txt>

[9] D6.3.1: 6NET IPv6 Network Management Cookbook

[10] D6.3.2: Final Report on IPv6 management and monitoring architecture, design, tools and operational procedures. Recommendations

[11] D3.1.2: IPv6 cookbook for routing, DNS, intra-domain multicast, and security

[12] Michael Behringer, "Tracing DoS Attacks," Hi Tech 2002 Workshop, Limmerick, IE, June 2002

[13] Internet Security Systems, "Security Implications of IPv6", white paper, http://documents.iss.net/whitepapers/IPv6.pdf

[14] Eric Marin, "IPv6 Security", presentation at 6NET Zagreb, May '03

[15] D4.2.2: Framework for the Support of IPv6 Wireless LANs ver.2

[16] D6.2.2: Operational procedures for secured management with transition mechanisms

[17] T. Narten, R. Draves, S. Krishnan, Privacy Extensions for Stateless Address Autoconfiguration in IPv6, work-in-progress, Dec. 23, 2004, <draft-ietf-ipv6-privacy-addrs-v2-02>

[18] J. Mohacsi, "IPv6 firewalls", presentation on the 5th TF-NGN meeting, October 2001 available at http:///skye.ki.iif.hu/~mohacsi/athens_tf_ngn_ipv6_firewalls.pdf

[19] J.Mohacsi, "Security of IPv6 from firewalls point of view", presentation on TNC2004 conference, June 2004, available at http://www.terena.nl/conferences/tnc2004/programme/presentations/show.php?pres_id=115

4. Appendix: Experiments performed at GARR on Fake Router Advertisements

4.1. Target system

The target network is connected to the 6NET network, whose topology is illustrated in Figure 1. Routers connected to it forward packets towards other routers. No simple hosts are present on the subnetwork considered.

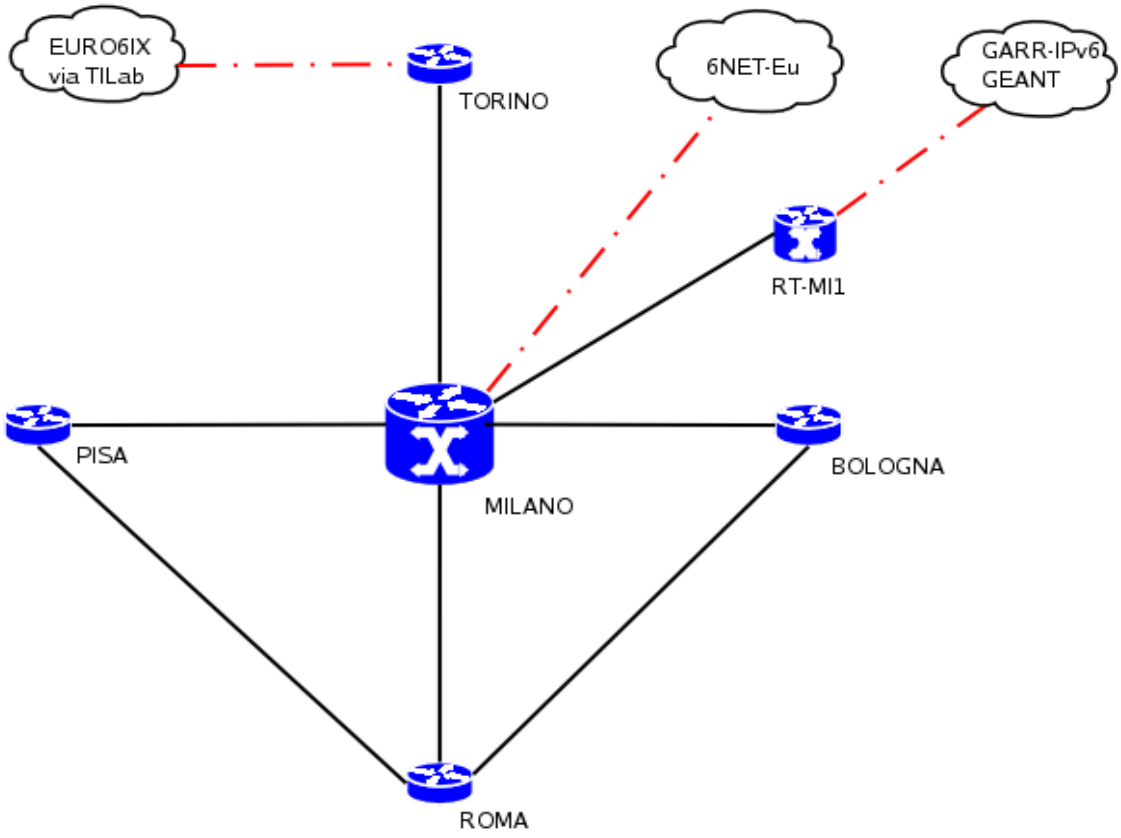

Figure 5: The 6NET topology in Italy, as of September 2004.

Because of the type of connections among the routers in Figure 1, and in order not to interfere with production traffic, tests were conducted in an experimental setting that reproduces the network configuration of a WAN. The network setting is depicted in Figure 2, where the Milano node of Figure 1 is expanded to illustrate the Universita` di Milano sub network. The three routers of the Universita` di Milano (henceforth, UNIMI) shown in Figure 2 are used to carry out the planned attacks. They are PC's with Pentium 4 Intel processors, 512MB of RAM, running the Linux operating system in the Redhat Fedora Core 9.1 distribution. Tests have been performed with both kernel version 2.4 and the latest stable version of 2.6. In order to identify possible distribution specific behaviours, we tried the same attacks against the same machines running Debian Linux V.T.

The network connection between the three routers is a multi-access inter-router one, where all routers exchange router advertisements with each other according to RFC 2461 and run IPv6 natively. Router B is also the designated router of one of UNIMI's laboratories (Laser) LAN sub network. Router C is the default router towards the rest of 6NET, and in particular towards the router of DIVTLC (router X).

In all distributions, the Linux operating system stores the information regarding the various IPv6 network parameters in corresponding files in the directory `/proc/sys/net/IPv6/conf/`. When new values are announced via a router advertisement message, the value currently stored in the corresponding file for the given parameter is updated.

In general, we will use router A as the attacker and routers B and C as the direct target or simply one of the (unaware) actors in the attacks. As such, we expect them to react to messages they see on the network, or change their state because of them, according to the network protocols they execute. The underlying assumption for the correct operation of the Neighbour Discovery protocol is that routers exchange honest and true information with each other. In case the behaviour of one or more of them deviates from the expected one, consequences of various degrees of severity are expected.

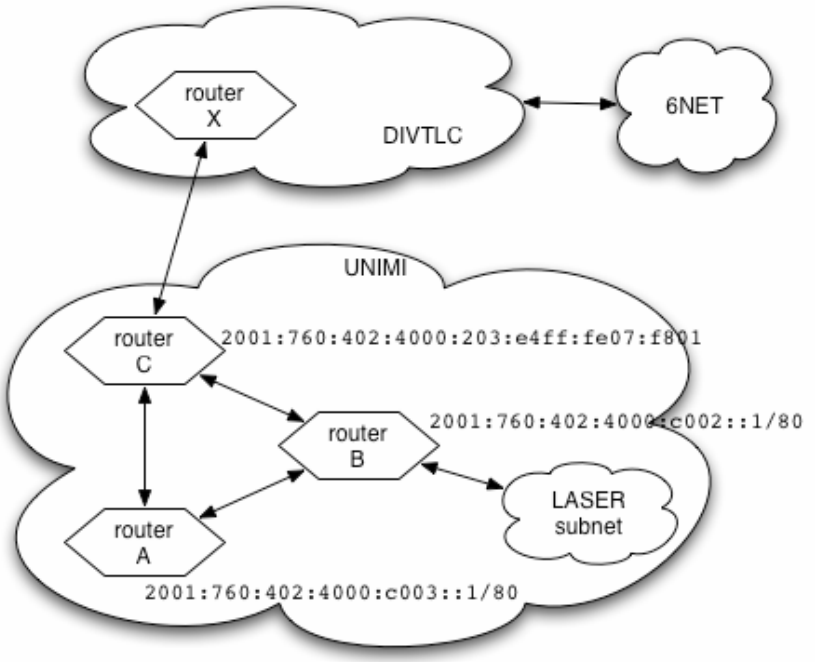

Figure 6: Network configuration for the target experimental system at UNIMI.

The general attack scheme is the following. Router A in Figure 2 is the malicious machine that injects fake router advertisements and other possible support packets into the link, part of the target WAN. Besides routers A, B, and C, another router is present on out target link but since it is not necessary for our tests, we will ignore it. Router A will try to block the communications between routers B and C, deny communications between them and affect communications in its own interest. For the attacks to succeed, it is critical that A be able to send packets to the target link. Whether A has been assigned a static address or used the stateless auto-configuration mechanism is irrelevant.

For the sake of convenience, we report here the configurations of the three routers, which will be the ones at the beginning of each of the attacks we try. The three routers are all statically configured with static IPv6 addresses.

Router A's configuration is given in Figure 3.

```
[root@routerA]# ip -6 addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,NOTRAILERS,UP> mtu 1500 qdisc pfifo_fast qlen 100
    inet6 2001:760:402:4000:c003::1/80 scope global
    inet6 fe80::201:2ff:fe07:8d50/10 scope link
```

Figure 7: Router A's configuration parameters.

IPv6 forwarding is enabled and router advertisements are accepted, although A is responsible for sending the various fake router advertisements during the attacks.

Router B, usually the target of the attacks, is configured as shown in Figure 4.

```
[root@routerB]# ip -6 addr show dev eth1
3: eth1: <BROADCAST,MULTICAST,PROMISC,UP> mtu 1500 qdisc pfifo_fast qlen 100
    inet6 2001:760:402:4000:c002::1/80 scope global
    inet6 fe80::204:76ff:fee2:c73a/64 scope link
```

Figure 8: Router B's configuration parameters.

Its IPv6 forwarding is enabled and it accepts router advertisements. In the case of the forwarding functionality, we used */proc/sys/net/IPv6/conf/all* because in the Linux OS IPv6 packet forwarding cannot be enabled per interface but as a whole.

According to the Neighbour Discovery protocol, router B periodically sends router advertisements to

```
[root@routerB]#radvdump
Router advertisement from fe80::204:76ff:fee2:c73a (hoplimit 255)
Received by interface eth1
        # Note: {Min,Max}RtrAdvInterval cannot be obtained with radvdump
        AdvCurHopLimit: 64
        AdvManagedFlag: off
        AdvOtherConfigFlag: off
        AdvHomeAgentFlag: off
        AdvReachableTime: 0
        AdvRetransTimer: 0
        Prefix 2001:760:402:4000:c002::/80
                AdvValidLifetime: 2592000
                AdvPreferredLifetime: 604800
                AdvOnLink: on
                AdvAutonomous: on
                AdvRouterAddr: on
        AdvSourceLLAddress: 00 04 76 E2 C7 3A
```

Router C's global IpV6 address is `2001:760:402:4000:203:e4ff:fe07:f801`, its link IpV6 address is `fe80::203:e4ff:fe07:f801`. IPv6 forwarding is enabled and router advertisements are accepted. It sends router advertisements as shown below:

```
[root@routerB]#radvdump
Router advertisement from fe80::203:e4ff:fe07:f801 (hoplimit 255)
Received by interface eth1
        # Note: {Min,Max}RtrAdvInterval cannot be obtained with radvdump
        AdvCurHopLimit: 64
        AdvManagedFlag: off
        AdvOtherConfigFlag: off
        AdvHomeAgentFlag: off
        AdvReachableTime: 0
        AdvRetransTimer: 0
        AdvSourceLLAddress: 00 03 E4 07 F8 01
        AdvLinkMTU: 1500
        Prefix 2001:760:402:4000::/64
                AdvValidLifetime: 2592000
                AdvPreferredLifetime: 604800
                AdvOnLink: on
                AdvAutonomous: on
                AdvRouterAddr: off
```

When off-link communication is performed, we refer to one of DIVTLC's routers, which we call router X and whose global IpV6 address is `2001:760:402:3800:2d0:58ff:fe61:18e0`

In order to complete the configuration information of the routers and facilitate their identification throughout the remaining of the text, Table 1 reports their MAC addresses.

ROUTER	MAC ADDRESS
A	00:01:02:07:8d:50
B	00:04:76:e2:c7:3a
C	00:03:e4:07:f8:01

Table 1: MAC addresses of the three routers used in the tests.

Each one of the systems on the target link, i.e., A, B. or C, may be used for sniffing packets during test sessions. Packets captured with `tcpdump`[2] during sessions will be used as evidence of the occurrence of certain events or actions performed by participating actors. The system is reset to its original state after each attack, in order to avoid affecting the outcome of a test with the results of a previous one.

4.2. Attacks on the target system

[2] `tcpdump` is a popular tool used to dump the content of a network in ASCII format (http:www.tcpdump.org). In the rest of this section, we use it with option `-e` in order to show the data link layer information and option `-vvv` so that the maximum amount of information is reported.

The attacks we planned against the target system focus on the following parameters:

a) Prefixes
b) MTU
c) Current hop limit
d) Link layer address
e) Router lifetime

We describe them in the remaining part of this section.

Prefixes

By communicating spurious prefixes, with the on-link flag set when needed, an attacking node may achieve the following goals:

a) Force other nodes to misconfigure their addresses.
b) Lead nodes to think that some frequently needed destinations are directly connected to the link they are attached to, thus they should be contacted as on-link destinations rather than off-link.
c) Confuse nodes in order to affect their choice of interface they should use for sending or forwarding some given packets.

It is fundamental that nodes insert the false prefixes into their prefix lists for this attack to succeed.

Case a)

We first consider the attack where a node is misled into reconfiguring its own address. In our experiment, attacking router A announces a non-existent prefix, namely `2001:760:402:3700::/64`, on behalf of default router C, i.e., A uses C's IPv6 address in the source field of the advertisement. Router B accepts the newly announced prefix and inserts it in its prefix list.

Before the attack starts, B's configuration is the one reported in Figure 4. It uses its original address 2001:760:402:4000:c002::1 to communicate, as the information captured by `tcpdump` shows in case of an `echo` request from B to router X, the off-link router at DIVTLC which is two hops away from B:

```
[root@routerB]# tcpdump -i eth1 -e -vvv
17:59:45.348483 0:4:76:e2:c7:3a 0:3:e4:7:f8:1 ip6 118: 2001:760:402:4000:c002::1
> 2001:760:402:3800:2d0:58ff:fe61:18e0: icmp6: echo request (len 64, hlim 64)
17:59:45.354469 0:3:e4:7:f8:1 0:4:76:e2:c7:3a ip6 118:
2001:760:402:3800:2d0:58ff:fe61:18e0 > 2001:760:402:4000:c002::1: icmp6: echo
reply (len 64, hlim 63)
```

When A starts its attack, it sends the following router advertisement on behalf of C, announcing the fake prefix `2001:760:402:3700::/64`:

```
[root@routerB]#radvdump
Router advertisement from fe80::203:e4ff:fe07:f801 (hoplimit 255)
Received by interface eth1
        # Note: {Min,Max}RtrAdvInterval cannot be obtained with radvdump
        AdvCurHopLimit: 64
        AdvManagedFlag: off
```

```
        AdvOtherConfigFlag: off
        AdvHomeAgentFlag: off
        AdvReachableTime: 0
        AdvRetransTimer: 0
        AdvSourceLLAddress: 00 03 E4 07 F8 01
        AdvLinkMTU: 1288
        Prefix 2001:760:402:3700::/64
                AdvValidLifetime: 2592000
                AdvPreferredLifetime: 604800
                AdvOnLink: on
                AdvAutonomous: on
                AdvRouterAddr: off
```

As the screen shot given below shows, after receiving the router advertisement with the fake prefix, in spite of the fact that its original address had been statically configured, router B assumes the additional address `2001:760:402:3700:204:76ff:fee2:c73a` derived from the received prefix. The new address is added and used instead of the original static address for all the indicated lifetime.

```
[root@routerB]# ip -6 addr show dev eth1
3: eth1: <BROADCAST,MULTICAST,PROMISC,UP> mtu 1500 qdisc pfifo_fast qlen 100
    inet6 2001:760:402:4000:c002::1/80 scope global
    inet6 2001:760:402:3700:204:76ff:fee2:c73a/64 scope global dynamic
       valid_lft 2591981sec preferred_lft 604781sec
    inet6 fe80::204:76ff:fee2:c73a/64 scope link
```

Since the other on-link routers do not recognize such an address, router B is isolated and cannot send packets to any other node using its new (fake) address. Tests show that this denial of service is somehow unidirectional. When transmitting, router B will use the new address, thus being unable to communicate. However, if other machines contact B using its original honest address, B can reply. As an example, when B `ping`s router X, which it could contact before the attack through C, none of the packets is ever delivered, as the screen shot below shows.

```
[root@routerB]# ping6 2001:760:402:3800:2d0:58ff:fe61:18e0
PING 2001:760:402:3800:2d0:58ff:fe61:18e0(2001:760:402:3800:2d0:58ff:fe61:18e0)
56 data bytes
From ::1 icmp_seq=1 Destination unreachable: Address unreachable
From ::1 icmp_seq=2 Destination unreachable: Address unreachable
From ::1 icmp_seq=3 Destination unreachable: Address unreachable
From ::1 icmp_seq=4 Destination unreachable: Address unreachable
From ::1 icmp_seq=5 Destination unreachable: Address unreachable
From ::1 icmp_seq=6 Destination unreachable: Address unreachable
From ::1 icmp_seq=7 Destination unreachable: Address unreachable
From ::1 icmp_seq=8 Destination unreachable: Address unreachable
From ::1 icmp_seq=9 Destination unreachable: Address unreachable

--- 2001:760:402:3800:2d0:58ff:fe61:18e0 ping statistics ---
9 packets transmitted, 0 received, +9 errors, 100% packet loss, time 16053ms
```

The packet content of the `ping` command is illustrated in the output of `tcpdump`:

```
[root@routerB]# tcpdump -i eth1 -e -vvv
tcpdump: WARNING: eth1: no IPv4 address assigned
tcpdump: listening on eth1
18:55:04.187124 0:4:76:e2:c7:3a 0:3:e4:7:f8:1 ip6 118:
2001:760:402:3700:204:76ff:fee2:c73a > 2001:760:402:3800:2d0:58ff:fe61:18e0:
icmp6: echo request (len 64, hlim 64)
18:55:05.186940 0:4:76:e2:c7:3a 0:3:e4:7:f8:1 ip6 86: fe80::204:76ff:fee2:c73a >
fe80::203:e4ff:fe07:f801: icmp6: neighbor sol: who has
fe80::203:e4ff:fe07:f801(src lladdr: 00:04:76:e2:c7:3a) (len 32, hlim 255)
18:55:05.187012 0:4:76:e2:c7:3a 0:3:e4:7:f8:1 ip6 118:
2001:760:402:3700:204:76ff:fee2:c73a > 2001:760:402:3800:2d0:58ff:fe61:18e0:
icmp6: echo request (len 64, hlim 64)
18:55:05.187551 0:3:e4:7:f8:1 0:4:76:e2:c7:3a ip6 78: fe80::203:e4ff:fe07:f801 >
fe80::204:76ff:fee2:c73a: icmp6: neighbor adv: tgt is
fe80::203:e4ff:fe07:f801(RS) [class 0xe0] (len 24, hlim 255)
18:55:06.186821 0:4:76:e2:c7:3a 0:3:e4:7:f8:1 ip6 118:
2001:760:402:3700:204:76ff:fee2:c73a > 2001:760:402:3800:2d0:58ff:fe61:18e0:
icmp6: echo request (len 64, hlim 64)
```

As we said, B's packets are lost because B uses its newly auto-configured address, the bogus one it accepted from A's fake router advertisement. As a consequence, C does not send its own data link layer address to B, thus indicating it refuses to forward B's packets. Note that since B is not receiving any reply, it checks whether the default router is reachable (neighbour solicitation and neighbour advertisement in the `tcpdump` output shown above), but this does not help. After router B receives the default router C advertisement, it keeps sending echo requests to the destination, which will be ignored since their source address contains a wrong prefix.

The attack succeeds because B is configured to accept router advertisements, i.e., file `/proc/sys/net/ipv6/conf/all/accept_ra` contains the value 1, which means that all the information they carry is considered valid and saved appropriately.

A simple remedy to protect the routers from the attack described above is to have them not accept router advertisement messages. This, however, would make it vain to run the Neighbour Discovery protocol.

Case b)

The way nodes communicate with other nodes is different depending upon whether the destination is on-link or off-link. In the former case neighbour/router solicitations and neighbour/router advertisements are used to discover the destination's data link layer address. In the latter, packets destined to remote nodes are sent to the default router, which in turn forwards them toward the destination. By announcing fake information about off-link nodes so as to make them look like on-link nodes, a node that accepts router advertisements will believe such nodes are on-link and will try to contact them accordingly. That is, it will send neighbour or router advertisements and expect neighbour or router advertisements, respectively, which will never be sent back because the selected destinations are not on-link.

In order to better understand the effects of the proposed attack, we first illustrate a normal communication exchange between off-link nodes, when the router information is sound. All the participating routers are configured as described in Figures 3-4. The screen shot below shows how an `echo` request sent by router B to router X is dealt with, as seen with `tcpdump`. Router B knows that X is not on-link because the prefix of its address is not advertised by any of the on-link routers, so it sends its `echo` request to X to the data link layer address of default router C for further forwarding. C, in turn, sends a neighbour solicitation to the link in order to discover B's data link

layer address, to which B replies with a neighbour advertisement. C then forwards B's request to X, not visible in the screen shot since it uses a link out of the `tcpdump` scope, and X's `echo` reply to B (last two lines in the `tcpdump` output).

```
[root@machineB]# tcpdump -i eth1 -e -vvv
tcpdump: WARNING: eth1: no IPv4 address assigned
tcpdump: listening on eth1
13:57:14.693759 0:4:76:e2:c7:3a 0:3:e4:7:f8:1 ip6 118: 2001:760:402:4000:c002::1
> 2001:760:402:3800:2d0:58ff:fe61:18e0: icmp6: echo request (len 64, hlim 64)
13:57:14.700907 0:3:e4:7:f8:1 33:33:ff:0:0:1 ip6 86: fe80::203:e4ff:fe07:f801 >
ff02::1:ff00:1: icmp6: neighbor sol: who has
2001:760:402:4000:c002::1(src lladdr: 00:03:e4:07:f8:01) [class 0xe0] (len 32,
hlim 255)
13:57:14.700981 0:4:76:e2:c7:3a 0:3:e4:7:f8:1 ip6 86: 2001:760:402:4000:c002::1
> fe80::203:e4ff:fe07:f801: icmp6: neighbor adv: tgt is
2001:760:402:4000:c002::1(SO)(tgt lladdr: 00:04:76:e2:c7:3a) (len 32, hlim 255)
13:57:14.702117 0:3:e4:7:f8:1 0:4:76:e2:c7:3a ip6 118:
2001:760:402:3800:2d0:58ff:fe61:18e0 > 2001:760:402:4000:c002::1: icmp6: echo
reply (len 64, hlim 63)
```

During the attack, router A sends a fake router advertisement on behalf of default router C announcing prefix `2001:760:402:3800::/64`, i.e., router X's prefix, as its own. The content of such a fake router advertisement, as received by B, is the following:

```
[root@machineB]# radvdump
Router advertisement from fe80::203:e4ff:fe07:f801 (hoplimit 255)
Received by interface eth1
        # Note: {Min,Max}RtrAdvInterval cannot be obtained with radvdump
        AdvCurHopLimit: 64
        AdvManagedFlag: off
        AdvOtherConfigFlag: off
        AdvHomeAgentFlag: off
        AdvReachableTime: 0
        AdvRetransTimer: 0
        AdvSourceLLAddress: 00 03 E4 07 F8 01
        AdvLinkMTU: 1288
        Prefix 2001:760:402:3800::/64
                AdvValidLifetime: 2592000
                AdvPreferredLifetime: 604800
                AdvOnLink: on
                AdvAutonomous: on
                AdvRouterAddr: off
```

After processing the fake router advertisement, router B believes that router X is, or has become, on-link. Therefore, when B tries to contact it again, it does so the way on-link routers are contacted, thus failing as the output of the `ping` command reported below shows. The details of the packet information used in the `ping` command can be seen in the `tcpdump` output immediately following.

```
[root@machineB]# ping6 2001:760:402:3800:2d0:58ff:fe61:18e0
PING 2001:760:402:3800:2d0:58ff:fe61:18e0(2001:760:402:3800:2d0:58ff:fe61:18e0)
56 data bytes
From ::1 icmp_seq=1 Destination unreachable: Address unreachable
From ::1 icmp_seq=2 Destination unreachable: Address unreachable
From ::1 icmp_seq=3 Destination unreachable: Address unreachable

--- 2001:760:402:3800:2d0:58ff:fe61:18e0 ping statistics ---
```

```
4 packets transmitted, 0 received, +3 errors, 100% packet loss, time 3042ms


[root@machineB]# tcpdump -i eth1 -e -vvv
tcpdump: WARNING: eth1: no IPv4 address assigned
tcpdump: listening on eth1
13:59:43.020320 0:4:76:e2:c7:3a 33:33:ff:61:18:e0 ip6 86:
2001:760:402:4000:c002::1 > ff02::1:ff61:18e0: icmp6: neighbor sol: who has
2001:760:402:3800:2d0:58ff:fe61:18e0(src lladdr: 00:04:76:e2:c7:3a) (len 32,
hlim 255)
13:59:44.062775 0:4:76:e2:c7:3a 33:33:ff:61:18:e0 ip6 86:
2001:760:402:4000:c002::1 > ff02::1:ff61:18e0: icmp6: neighbor sol: who has
2001:760:402:3800:2d0:58ff:fe61:18e0(src lladdr: 00:04:76:e2:c7:3a) (len 32,
hlim 255)
13:59:45.060319 0:4:76:e2:c7:3a 33:33:ff:61:18:e0 ip6 86:
2001:760:402:4000:c002::1 > ff02::1:ff61:18e0: icmp6: neighbor sol: who has
2001:760:402:3800:2d0:58ff:fe61:18e0(src lladdr: 00:04:76:e2:c7:3a) (len 32,
hlim 255)
```

Case c)

The last consequence of fake router advertisements we analyze is that of confusing nodes in order to affect their choice of interface to use for sending packets to a given host. In our usual setting, let us assume that router B reaches router X using its network interface `eth1`. If a fake router advertisement like the one used in the attack of Case b) is received by B through its network interface `eth0`, then B will assume it can reach router X using `eth0`. The actions and results of this attack are the same as those described in Case b).

We tried a different attack, aiming at exhausting the memory space for advertised prefixes, by advertising a progressively larger number of prefixes. However, Linux proved to be resilient to such an attack, as only up to 15 prefixes are stored by a listening node, regardless of the distribution used.

Maximum Transfer Unit

By advertising a very small link MTU size, routers will use small size packets, thus possibly congesting the network by increasing the packet headers overhead. Furthermore, if the advertised MTU is very small, a node may not be able to communicate at all with the router that appears to have sent the advertisement, as we show below. IPv6 address length of 128 bits is critical in this scenario, as they increase the overhead in each packet. The overhead of an IPv6 packet is the sum of the number of bits in the various headers, namely the data link layer header, the IPv6 header, and the extension headers:

$$\text{Overhead} = \text{Header\_size}/\text{Total\_packet\_length}$$

Where the Header_size = (#bits in data link layer header)+ (#bits in IPv6 header) + (#bits in eventual extension headers). Furthermore, the extension headers, which are accommodated in the IPv6 packet payload, can also become too large to be contained in a limited size packet. We show here that this causes a complete break down of communications.

As Figure 4 shows, router B's MTU is initially set to 1500. This means that other routers accepting B's advertisements will respect this MTU when sending packets to B. Attacking router A injects a fake router advertisement forging the source address so as to make it look like coming from router C. This fake router advertisement announces an MTU of 17 bytes. As a result, router B believes that router C is advertising an MTU of 17 bytes, which means that router C wants other machines to adopt the advertised MTU when sending packets to it. B is not able to communicate with C, since even the smallest packet exceeds the current MTU. The consequence of the attack is to block the communication towards the node that has advertised a very small MTU value, or flood such nodes with a large quantity of small packets.

As the output of the `ping` command shows, before the attack B can reach router C:

```
[root@routerB]# ping6 2001:760:402:4000:203:e4ff:fe07:f801
PING 2001:760:402:4000:203:e4ff:fe07:f801(2001:760:402:4000:203:e4ff:fe07:f801)
56 data bytes
64 bytes from 2001:760:402:4000:203:e4ff:fe07:f801: icmp_seq=1 ttl=64 time=2.48
ms
64 bytes from 2001:760:402:4000:203:e4ff:fe07:f801: icmp_seq=2 ttl=64 time=0.634
ms
64 bytes from 2001:760:402:4000:203:e4ff:fe07:f801: icmp_seq=3 ttl=64 time=0.725
ms

--- 2001:760:402:4000:203:e4ff:fe07:f801 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2015ms
rtt min/avg/max/mdev = 0.634/1.282/2.488/0.853 ms
```

B's knowledge of other on-link routers before the attack is reported below:

```
[root@routerB]# ip -6 route show dev eth1
2001:760:402:4000:c002::/80  proto kernel  metric 256  mtu 1500 advmss 1440
2001:760:402:4000::/64  proto kernel  metric 256  expires 2591921sec mtu 1500
advmss 1440
fe80::/64  proto kernel  metric 256  mtu 1500 advmss 1440
ff00::/8  proto kernel  metric 256  mtu 1500 advmss 1440
default via fe80::203:e4ff:fe07:f801  proto kernel  metric 1024  expires 1721sec
mtu 1500 advmss 1440
default via fe80::204:76ff:fee2:c73a  proto kernel  metric 1024  expires 29sec
mtu 1500 advmss 1440
```

In particular, default router fe80::203:e4ff:fe07:f801, i.e., C, advertises MTU = 1500. As the fake router advertisement sent by router A that appears to come from C given below shows, the newly advertised MTU is 17:

```
[root@routerB]#radvdump
Router advertisement from fe80::203:e4ff:fe07:f801 (hoplimit 255)
Received by interface eth1
        # Note: {Min,Max}RtrAdvInterval cannot be obtained with radvdump
        AdvCurHopLimit: 64
        AdvManagedFlag: off
        AdvOtherConfigFlag: off
        AdvHomeAgentFlag: off
        AdvReachableTime: 0
        AdvRetransTimer: 0
        AdvSourceLLAddress: 00 03 E4 07 F8 01
```

```
        AdvLinkMTU: 17
        Prefix 2001:760:402:4000::/64
                AdvValidLifetime: 2592000
                AdvPreferredLifetime: 604800
                AdvOnLink: on
                AdvAutonomous: on
                AdvRouterAddr: off
```

Since B is configured to accept router advertisements, it updates its MTU to the advertised value (17). As a consequence, it cannot reach router C anymore because even the simple `ping` message is "too long" and all packets are discarded:

```
[root@routerB]# ping6 2001:760:402:4000:203:e4ff:fe07:f801
PING 2001:760:402:4000:203:e4ff:fe07:f801(2001:760:402:4000:203:e4ff:fe07:f801)
56 data bytes
ping: local error: Message too long, mtu=17
ping: local error: Message too long, mtu=17
ping: local error: Message too long, mtu=17
ping: local error: Message too long, mtu=17

--- 2001:760:402:4000:203:e4ff:fe07:f801 ping statistics ---
4 packets transmitted, 0 received, +4 errors, 100% packet loss, time 2999ms
```

Similarly, when another on-link router `pings` B, B cannot reply:

```
[root@anotherRouter]# ping6 2001:760:402:4000:c002::1
PING 2001:760:402:4000:c002::1(2001:760:402:4000:c002::1) from
2001:760:402:4000::dead:beef : 56 data bytes

--- 2001:760:402:4000:c002::1 ping statistics ---
13 packets transmitted, 0 received, 100% loss, time 12018ms
```

After the attack, B's knowledge about routers on-link is as follows:

```
[root@routerB]# ip -6 route show dev eth1
2001:760:402:4000:c002::/80  proto kernel  metric 256  mtu 17 advmss 65535
2001:760:402:4000::/64  proto kernel  metric 256  expires 2591905sec mtu 17
advmss 65535
fe80::/64  proto kernel  metric 256  mtu 17 advmss 65535
ff00::/8  proto kernel  metric 256  mtu 17 advmss 65535
default via fe80::203:e4ff:fe07:f801  proto kernel  metric 1024  expires 1705sec
mtu 17 advmss 65535
```

In this case we observed that not all the Linux distributions behave the same. We ran the same experiment with B running RedHat, Mandrake, and Debian distributions, respectively. The RedHat distribution maintains the MTU value received from the router, thus blocking the machine until the network administrator manually resets it. The Mandrake distribution seems to apply a static solution, that is, it changes

`/proc/sys/net/ipv6/conf/interface-name/mtu` but does not respect it. The Debian distribution ignores very little MTU values. In the case of this attack, the `/proc/sys/net/ipv6/conf/interface-name/mtu` file did not change. In case of bigger values, however, the MTU change is reflected in B's knowledge about on-link routers.

Since it is not possible to selectively accept part of the information in a router advertisement message, in this case as in the prefix case, in the absence of cryptographically secured information, there is no other defence but to reject router advertisements, which implies avoiding the Neighbour Discovery protocol at all. While this could be a viable solution in small static networks with few nodes, it is not practical in large highly dynamic networks where nodes frequently join and leave the network. Furthermore, one of the founding reasons behind the introduction of IPv6, namely the ability to add network devices with no manual intervention, is defeated.

Current hop limit

By sending fake router advertisements that advertise very small current hop limit values, routers that accept such advertisements will send packets that may not be able to reach their destinations.

According to the original configuration, see Figure 4, router B's hop limit is set to 64. Router B can reach router X in 2 hops, one from B to C and one from C to X, as the result of the `traceroute` command to X reported below shows.

```
[root@routerB]#traceroute6 2001:760:402:3800:2d0:58ff:fe61:18e0
traceroute to 2001:760:402:3800:2d0:58ff:fe61:18e0
(2001:760:402:3800:2d0:58ff:fe61:18e0) from 2001:760:402:4000:c002::1, 30
hops max, 16 byte packets
 1  2001:760:402:4000:203:e4ff:fe07:f801 (2001:760:402:4000:203:e4ff:fe07:f801)
27.272 ms  0.69 ms  28.689 ms
 2  2001:760:402:3800:2d0:58ff:fe61:18e0 (2001:760:402:3800:2d0:58ff:fe61:18e0)
87.144 ms  61.629 ms  61.686 ms
```

In the attack, machine A sends the following fake router advertisement forging its origin as from C:

```
[root@routerB]# radvdump
Router advertisement from fe80::203:e4ff:fe07:f801 (hoplimit 255)
Received by interface eth1
        # Note: {Min,Max}RtrAdvInterval cannot be obtained with radvdump
        AdvCurHopLimit: 1
        AdvManagedFlag: off
        AdvOtherConfigFlag: off
        AdvHomeAgentFlag: off
        AdvReachableTime: 0
        AdvRetransTimer: 0
        AdvSourceLLAddress: 00 03 E4 07 F8 01
        AdvLinkMTU: 1288
        Prefix 2001:760:402:4000::/64
                AdvValidLifetime: 2592000
                AdvPreferredLifetime: 604800
                AdvOnLink: on
                AdvAutonomous: on
                AdvRouterAddr: off
```

When B receives the fake router advertisement, it updates its current hop limit to the advertised value, i.e., 1, as file `/proc/sys/net/IPv6/conf/eth1/hop_limit` shows:

```
[root@routerB]# cat /proc/sys/net/IPv6/conf/eth1/hop_limit
1
```

Experiments show that also the Mandrake and Debian distributions update the configuration file with the value announced by the fake router advertisement. Whether the information stored in the configuration file is ever used depends upon how closely they respect RFC 2461 specifications.

Data link layer address manipulation

Routers in a network store each other's data link layer and IPV6 addresses, and other information needed to forward packets to each other. In this experiment we send router advertisements with spoofed data link layer address and forged source address. This way we can reroute a message from the default router to the node that sent the fake router advertisement. Our goal is to have the spoofing router receive the packets that the target router sends to the spoofed router, i.e., hijack the traffic that is supposed to be sent towards the router whose link layer address is being spoofed. Thus, we send fake router advertisements to a target router forging the source as one of its default routers. These fake router advertisements contain another machine's data link layer address as both sender's data link layer address and the value of the data link layer option in the ICMPv6 extension header.

Unlike what happens with the other bogus information announced in the fake router advertisements, the system resists the attack but the result is blocking communication towards the spoofed machine. In the presence of inconsistent router identifying information, the target router removes the entry relative to the spoofed router from its memory. This could be particularly serious if the removed router was the only default router for the target machine, as this breaks the communication with the rest of the world for some time.

The information machine B has about on-link routers before the attack is reported below:

```
[root@routerB]# ip -6 route show dev eth1
2001:760:402:4000:c002::/80  proto kernel  metric 256  mtu 1288 advmss 1440
2001:760:402:4000::/64  proto kernel  metric 256  expires 2591953sec mtu 1288
advmss 1440
fe80::/64  proto kernel  metric 256  mtu 1288 advmss 1440
ff00::/8  proto kernel  metric 256  mtu 1288 advmss 1440

default via fe80::203:e4ff:fe07:f801  proto kernel  metric 1024  expires 1753sec
mtu 1500 advmss 1440
```

We can see that the last entry is relative to router C, which is a default router for B. Router B forwards packets to C for further forwarding towards destinations, as shown below in the `traceroute` output to router X:

```
[root@routerB]# traceroute6 2001:760:402:3800:2d0:58ff:fe61:18e0
traceroute to 2001:760:402:3800:2d0:58ff:fe61:18e0
(2001:760:402:3800:2d0:58ff:fe61:18e0) from 2001:760:402:4000:c002::1, 30 hops
max, 16 byte packets
 1  2001:760:402:4000:203:e4ff:fe07:f801 (2001:760:402:4000:203:e4ff:fe07:f801)
3.131 ms  0.698 ms  0.574 ms

 2  2001:760:402:3800:2d0:58ff:fe61:18e0  (2001:760:402:3800:2d0:58ff:fe61:18e0)
4.97 ms  5.004 ms  5.013 ms
```

As it is shown above, the first router reached by the `traceroute` packets is the default router C. The same operation, as captured with `tcpdump`, shows that packets are being sent to C's data link layer address, that is, `0:3:e4:7:f8:1`

```
[root@routerB]# tcpdump -i eth1 -e -vvv
14:57:49.950508 0:4:76:e2:c7:3a 0:3:e4:7:f8:1 ip6 78:
2001:760:402:4000:c002::1.32796 >
2001:760:402:3800:2d0:58ff:fe61:18e0.traceroute: [udp sum ok] udp 16 [hlim 1]
(len 24)
```

This is a router advertisement sent by router C at regular intervals.

```
[root@routerB]# tcpdump -i eth1 -e -vvv
11:49:57.010497 0:3:e4:7:f8:1 33:33:0:0:0:1 ip6 118: fe80::203:e4ff:fe07:f801 >
ff02::1: icmp6: router advertisement(chlim=64, pref=medium, router_ltime=1800,
reachable_time=0, retrans_time=0)(src lladdr: 00:03:e4:07:f8:01)(mtu:
mtu=1500)[ndp opt] [class 0xe0] (len 64, hlim 255)
```

During the attack, router A sends a router advertisement to router B forging the source as C. Note that both the sender's data link layer address and the value of the data link layer option will be equal to the data link layer of A, which is `0:1:2:7:8d:50`, although the IPv6 source address is C's (`fe80::203:e4ff:fe07:f801`). The fake router advertisement, as captured with `tcpdump`, is as follows:

```
[root@routerB]# tcpdump -i eth1 -e -vvv
11:50:58.955502 0:1:2:7:8d:50 33:33:0:0:0:1 ip6 118: fe80::203:e4ff:fe07:f801 >
ff02::1: icmp6: router advertisement(chlim=64, pref=medium, router_ltime=0,
reachable_time=0, retrans_time=0)(src lladdr: 00:01:02:07:8d:50)(mtu:
mtu=1288)[ndp opt] [class 0xf0] (len 64, hlim 255)
```

Target router B receives inconsistent router advertisement information and consequently deletes the entry of default router C, as shown below:

```
[root@routerB]#ip -6 route show dev eth1
2001:760:402:4000:c002::/80  proto kernel  metric 256  mtu 1288 advmss 1484
2001:760:402:4000::/64  proto kernel  metric 256  expires 2591997sec mtu 1288
advmss 1484
fe80::/64  proto kernel  metric 256  mtu 1288 advmss 1484
ff00::/8  proto kernel  metric 256  mtu 1288 advmss 1484
```

As a result, no default routers are left. After the attack, B's request to reach X fails because B has no default routers that can forward its packets, as shown in the `traceroute` output:

```
[root@routerB]# traceroute6 2001:760:402:3800:2d0:58ff:fe61:18e0
connect: Network is unreachable
```

This type of attack works against all the Linux distributions we tested.

Router lifetime

Another way to have a router remove a default router from its router list is to send a fake router advertisement that advertises a router lifetime value of 0. This way it is possible to make targets delete all of the routers present in their respective routers lists. Router entries in the router list can also be made to time out, thus making the information about the specific router useless. This form of attack relies on the fact that routers update their information regarding a router with the one received by the latest router advertisement from that router.

In our experiment, B's router list includes C as the only default router (link address `fe80::203:e4ff:fe07:f801`). Attacking machine A sends the following fake router advertisement carrying a router lifetime of 0 to target router B, using C as forged source address[3]

```
[root@routerB]# tcpdump -i eth1 -e -vvv
tcpdump: WARNING: eth1: no IPv4 address assigned
tcpdump: listening on eth1
16:18:27.753376 0:3:e4:7:f8:1 33:33:0:0:0:1 ip6 118: fe80::203:e4ff:fe07:f801 >
ff02::1: icmp6: router advertisement(chlim=64, pref=medium, router_ltime=0,
reachable_time=0, retrans_time=0)(src lladdr: 00:03:e4:07:f8:01)(mtu:
mtu=1288)[ndp opt] [class 0xf0] (len 64, hlim 255)
```

B's routers list after receiving and processing the fake router advertisement does not include C anymore, as shown below:

```
[root@localhost jlr670790]# ip -6 route show
2001:760:402:4000:c002::/80 dev eth1  proto kernel  metric 256  mtu 1288 advmss
1228
2001:760:402:4000::/64 dev eth1  proto kernel  metric 256  expires 2591986sec
mtu 1288 advmss 1228
fe80::/64 dev eth0  proto kernel  metric 256  mtu 1500 advmss 1228
fe80::/64 dev eth1  proto kernel  metric 256  mtu 1288 advmss 1228
ff00::/8 dev eth0  proto kernel  metric 256  mtu 1500 advmss 1228
ff00::/8 dev eth1  proto kernel  metric 256  mtu 1288 advmss 1228
unreachable default dev lo  metric -1  error -101 advmss 1228
```

The lack of a default router in B's router list prevents B from communicating, as the result of the `ping` command shows:

```
[root@routerB]# ping6 2001:760:402:3800:2d0:58ff:fe61:18e0
connect: Network is unreachable
```

By sending a fake router advertisement with a small but not null router lifetime with forged source address of C, attacking machine A makes target router B change expiration time for C's entry in its routers list from the current value to the new small one. As a consequence, B knows about C for that time interval only. After that time interval, C cannot be used anymore as a default router. As an

[3] Note that the retransmission time and reachable time set to 0, as in the router advertisement sent in this case, do not affect our test, since such a value is meant to let the receiving nodes use their default values for those parameters.

example, a fake router advertisement sent by attacking machine A with a router lifetime of 5 seconds follows:

```
[root@routerB]# tcpdump -i eth1 -e -vvv
tcpdump: WARNING: eth1: no IPv4 address assigned
tcpdump: listening on eth1
17:18:58.624862 0:3:e4:7:f8:1 33:33:0:0:0:1 ip6 118: fe80::203:e4ff:fe07:f801 >
ff02::1: icmp6: router advertisement(chlim=64, pref=medium, router_ltime=5,
reachable_time=153, retrans_time=0)(src lladdr: 00:03:e4:07:f8:01)(mtu:
mtu=1288)[ndp opt] [class 0xf0] (len 64, hlim 255)
```

As time passes, C's entry eventually times out:

```
[root@routerB]# ip -6 route show dev eth1
2001:760:402:4000:c002::/80  proto kernel  metric 256  mtu 1288 advmss 1228
2001:760:402:4000::/64  proto kernel  metric 256  expires 2591997sec mtu 1288
advmss 1228
fe80::/64  proto kernel  metric 256  mtu 1288 advmss 1228
ff00::/8  proto kernel  metric 256  mtu 1288 advmss 1228
default via fe80::203:e4ff:fe07:f801  proto kernel  metric 1024  expires 2sec
mtu 1288 advmss 1228

[root@routerB]# ip -6 route show dev eth1
2001:760:402:4000:c002::/80  proto kernel  metric 256  mtu 1288 advmss 1228
2001:760:402:4000::/64  proto kernel  metric 256  expires 2591996sec mtu 1288
advmss 1228
fe80::/64  proto kernel  metric 256  mtu 1288 advmss 1228
ff00::/8  proto kernel  metric 256  mtu 1288 advmss 1228
default via fe80::203:e4ff:fe07:f801  proto kernel  metric 1024  expires 1sec
mtu 1288 advmss 1228

[root@routerB]# ip -6 route show dev eth1
2001:760:402:4000:c002::/80  proto kernel  metric 256  mtu 1288 advmss 1228
2001:760:402:4000::/64  proto kernel  metric 256  expires 2591995sec mtu 1288
advmss 1228
fe80::/64  proto kernel  metric 256  mtu 1288 advmss 1228
ff00::/8  proto kernel  metric 256  mtu 1288 advmss 1228
default via fe80::203:e4ff:fe07:f801  proto kernel  metric 1024  expires 0sec
mtu 1288 advmss 1228
```

At this point, when the counter reaches "expires 0sec", the entry is removed and B is isolated. All the Linux distributions we tested behaved the same in this test.

Router lifetime and Data link layer manipulation

Traffic hijacking usually requires that traffic be redirected to a machine while obscuring the default router from the network traffic. In the IPv6 context with Neighbour Discovery protocol running traffic hijacking can be accomplished by sending a sequence of fake router advertisements. The first one advertises a router lifetime equal to 0 and data link layer equal to the one of the default router. This will force listening routers to classify the interface of the spoofed router as a non-advertising one. Immediately after that, the attacker sends another router advertisements, with source IP address of the spoofed router and data link layer address of the attacker, i.e., the spoofing router. This will

reinstate the spoofed router in the router list with a different data link layer address, the attacker's one. The overall result is the same as if the spoofed router had replaced its interface with another one.

In our experiment, A's goal is to hijack all traffic directed to C originating at B. Like in the case described before, A sends a fake router advertisement to router B with C's address as source address. This fake router advertisement carries a router lifetime of 0, which tells router B that router C is not its default router any longer. Therefore B deletes C's entry from its routers list. We recall that the data link layer address of router C is `0:3:e4:7:f8:1` and the data link layer address of router A is `0:1:2:7:8d:50`.

The fake router advertisement, as captured at B with `tcpdump`, is reported below:

```
[root@routerB]# tcpdump -i eth1 -e -vvv
14:55:12.555186 0:3:e4:7:f8:1 33:33:0:0:0:1 ip6 118: fe80::203:e4ff:fe07:f801 >
ff02::1: icmp6: router advertisement(chlim=64, pref=medium, router_ltime=0,
reachable_time=0, retrans_time=0)(src lladdr: 00:03:e4:07:f8:01)(mtu:
mtu=1288)[ndp opt] [class 0xf0] (len 64, hlim 255)
```

Attacking machine A then sends a second fake router advertisement with C's IPV6 link address as source address. This message advertises A's data link layer address in the sender's data link layer address (first line in the `tcpdump` output below) and in the data link layer option in the ICMPv6 extension header (`src lladdr: 00:01:02:07:8d:50`) in the output below).

```
[root@routerB]# tcpdump -i eth1 -e -vvv
14:55:26.606437 0:1:2:7:8d:50 33:33:0:0:0:1 ip6 118: fe80::203:e4ff:fe07:f801 >
ff02::1: icmp6: router advertisement(chlim=64, pref=medium, router_ltime=1800,
reachable_time=0, retrans_time=0)(src lladdr: 00:01:02:07:8d:50)(mtu:
mtu=1288)[ndp opt] [class 0xf0] (len 64, hlim 255)
```

At this point router B stores router C's information in its routers list. Router C will be a default router in B's router list but A's data link layer appears as C's. As an example of the consequences of the attack just described, we report the `tcpdump` output of router B trying to `traceroute` router X. In order to do this, target router B looks up its routers list, where it finds C as its default router. Thus it sends packets to C using the information stored in C's entry, including the data link layer address, which is in fact A's. As the output of `tcpdump` shows, B's packets are actually directed to the hijacking machine A (see first line):

```
[root@routerB]# tcpdump -i eth1 -e -vvv
14:33:32.191823 0:4:76:e2:c7:3a 0:1:2:7:8d:50 ip6 78:
2001:760:402:4000:c002::1.32795 >
2001:760:402:3800:2d0:58ff:fe61:18e0.traceroute: [udp sum ok] udp 16 [hlim 1]
(len 24)
```

All the Linux distributions we tested, namely RedHat, Mandrake and Debian, both kernel versions 2.4 and 2.6, behaved the same and the attack succeeded in all three cases.