Security for home, small and medium sized enterprises IPv6 networks

Security using simple network equipment

FREDRIK FOLKE

KTH Information and Communication Technology

Degree project in Communication Systems
First level, 15.0 HEC
Stockholm, Sweden

KTH Royal Institute of Technology
KUNGLIGA TEKNISKA HÖGSKOLAN

Security for home, small & medium sized enterprises IPv6 networks

Security using simple network equipment

Fredrik Folke
ffolke@kth.se

2012-06-21

Bachelor thesis

Examiner & Supervisor:

Professor Gerald Q. Maguire Jr.

This page intentionally left blank

Abstract

This theses project investigates and presents different threats that a network can be exposed to and the common protection techniques that can be applied, with a focus on the network perimeter – specifically the router/firewall between the local area network and the Internet. All Internet connected devices and networks are exposed to and affected by security threats to some degree, hence security is important in almost every type of network. With the constant growth of the Internet the 32-bit addressing scheme ipv4 is proving to be inadequate, and therefore the transition to the 128-bit addressing scheme ipv6 is becoming critical. With ipv6 comes new security threats (while still old threats remain) that requires an understanding of perimeter security. In this thesis we secure a home router and describe these steps to enable home and small business owners to secure their IPv6 network at a relatively low cost.

Sammanfattning

Detta projekt kommer att undersöka och presentera olika hot som ett IPv6 nätverk kan utsättas för samt de vanligaste skydds mekanismer som används idag, med fokus på nätverkets skallskydd mellan det interna lokala nätet och det yttre publika Internet. I stort sätt all Internet ansluten utrustning och nätverk är exponerad och påverkad i någon grad av säkerhets brister, säkerhet är en viktig del i stort sätt alla nätverk oavsett syfte eller verksamhet. Genom ett ständigt växande Internet börjar de 32-bitar adresser tillhörande IPv4 nätet ta slut, vilket gör behovet av att immigrera till 128-bitar adresser på IPv6 nätet allt mer kritiskt. Med IPv6 kommer nya säkerhetshot, samt att även vissa äldre hot kvarstår, som kräver en förståelse av perimeter skydd. I denna rapport säkrar vi en hemma router och beskriver för varje steg tillvägagångssättet för att hem och små företagare ska få möjlighet att skydda sina IPv6 nätverk till en relativt låg kostnad.

Table of Contents

List of Figures

List of Listings

This page intentionally left blank

List of Codings

This page intentionally left blank

List of Acronyms

ACL	Access Control List
CIFS	Common Internet File System
CPU	Central Processing Unit
CSRF	Cross-Site Request Forgery
DNS	Domain Name System
DoS	Denial of Service
DPI	Deep Packet Inspection
ETH	Ethernet
FTP	File Transfer Protocol
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IANA	the Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
ICMPv6	Internet Control Message Protocol Version 6
IDS	Intrusion Detection System
IP	Internet Protocol
IPsec	Internet Protocol Security
IPv6	Internet Protocol Version 6
IPv4	Internet Protocol Version 4
JFFS	Journaling Flash File System
LAN	Local Area Network
MAC	Media Access Control
MTU	Maximum Transmission Unit
NAS	Network Attached Storage
NVRAM	Non-Volatile Random-Access Memory

QoS	Quality of Service
RAM	Random-Access Memory
RFC	Request for Comments
RH0	Routing Header Type 0
RIR	Regional Internet Registry
ROM	Read Only Memory
SCP	Secure Copy Protocol
SIP	Session Initiation Protocol
SME	Small and Medium Enterprise
SNMP	Simple Network Management Protocol
SSH	Secure Shell
TCP	Transmission Control Protocol
TLS	Transport Layer Security
USB	Universal Serial Bus
VLAN	Virtual Local Area Network
VPN	Virtual Private Network
WAN	Wide Area Network
XSS	cross Site Scripting

Acknowledgments

I would like to express my sincere gratitude to my examiner and supervisor Professor Gerald Q. Maguire Jr. for giving me the opportunity to work with this interesting topic and his valuable support and feedback. I want to thank you for your patience and your very helpful guidance that inspired me to take this project further.

I would also like to thank my family, especially my dear mother for the effort to not disturb me when I was needed.

This page intentionally left blank

Chapter 1

1 Introduction

This chapter provides a general introduction of why there is a need to implement new IPv6 networks in homes and small and medium sized enterprises (SMEs). Following this overview from a security perspective we will examine in home and SME networks, along with a description of the importance of this bachelor's thesis and how it will improve network security in these IPv6 networks. The chapter ends with a description of who are the intended reader and a summary of the limitations of this thesis project.

1.1 The need for IPv6

Since the numbers of available Internet Protocol Version 4 (IPv4) addresses have been decreasing rapidly toward zero, there has been a growing concern for several years in different areas, including the fundamental infrastructure for economic and social activity around the world [1]. The Internet Assigned Numbers Authority [20] (IANA), the world´s source for IPv4 addresses, allocated the last blocks of IPv4 addresses to the regional registrars in January 2011 [2], which means the IANA´s pool of IPv4 addresses is already exhausted. We can still get IPv4 addresses allocated from those that each region was allocated in Africa, Asia, America, and Europe. However, when the current assignment to each Regional Internet Registry (RIR) has been depleted they will no longer be able to assign IPv4 addresses. This is estimated to occur around February 2013 according to Internet monitoring [23]. With the exhaustion of IPv4 addresses the growth of the IPv4 Internet will have reached its limit.

When an RIR has no more public IPv4 addresses to assign, then when new homes, businesses, or network devices need a public address they will have to face a reality where they cannot have a public IPv4 address. Some previously assigned IPv4 addresses may become available as contracts end, but this only delays the inevitable exhaustion of addresses. The actual address space is limited due to the choice of a 32 bit address for the IP source and destination address fields of the IP protocol packets and due to how the address space was divided when IPv4 was introduced.

A solution to this limitation of the Internet's expansion is to deploy Internet Protocol Version 6 (IPv6). Due to its 128 bit address space it offers greater scalability then IPv4 [2]. Since IPv6 is a new protocol for many users and network operators, there is a great need for reconfiguration and consideration of added security mechanisms to ensure that IPv6 packets do not bypass the existing security mechanisms which were originally intended to process only IPv4 traffic.

1.2 The scale of the problem

The security demands are increasing as IPv6 deployment expands over the globe. The fundamental problem is that every single network will be more or less forced to implement IPv6 at some point if they want to maintain their connectivity. Furthermore, all these new IPv6 networks need to consider their security against both new and old threats. Today ordinary IPv4 routers and firewalls simply cannot protect against IPv6 specific traffic.

Popular client operating systems (such as Microsoft's Windows, MAC OS, and Linux) all ship with IPv6 enabled by default which contributes to the increased preparedness of these systems to adopt IPv6, but also increases the impact of IPv6 related security problems. Because in most cases IPv6 security is not a high priority or even a concern, thus most users are not aware of using IPv6 or that they even have it enabled. Unfortunately in reality this means that these hosts could be wide open for all sorts of attacks and abuse by anyone connected anywhere on the Internet via IPv6.

Malware, malicious intended code such as virus, worms, and Trojans, often infiltrates and in some cases remotely control network connected devices without the user's knowledge. Such malware has become a common global security problem because this malware takes advantage of the limited or non-existent security in many networks. One of the most famous examples of malware today is the Zeus Trojan specialized in stealing banking information. This malware now claims to have IPv6 support [4]. Such malware mainly uses IPv6 to tunnel traffic between the compromised host and the attacker (i.e., to provide a secure tunnel for the botnet's command and control traffic) [7].

One common problem that motivates this thesis project for home users is that the use of virtual private network (VPN) services has grown in popularity due to concerns about Internet integrity. Many home VPN users are unaware that they have an IPv6 address as part of their IPv4 VPN tunnel service. Thus the VPN can deliver IPv6 traffic that they are unaware of and unprotected from. This particular problem cannot be solved at the router/firewall since the VPN traffic is encrypted before it reaches the router! Knowledge of the existence of IPv6 threats is a requirement in today's Internet.

1.3 Overview of the planned bachelor's thesis project

With increasing IPv6 deployment, this is an important and good opportunity to consider the new and old threats that IPv6 networks will be exposed to. Today there are small IPv6 routers and firewalls available from all of the common brands, including: Cisco, Zyxel, D-link, and Netgear. However, in recent product reviews [3], few of these home/SME firewalls actually deliver good security or they need very specific configuration in order to deliver good security.

This bachelor's thesis project will investigate how to implement security for home and SME IPv6 networks, without the need for expensive pre-built hardware firewalls, rather this security can be implemented with simple home networking equipment, thus decreasing the home owner's or business owner's cost and giving them control of their own network's security.

The investigation begins with what type of threats target IPv6 networks. Specifically we will examine unfiltered traffic, router break-ins, inside-out attacks, service vulnerabilities, and IPv6 protocol specific vulnerabilities. Secondly we will examine what can be done with an ordinary home router in order to provide perimeter security and what can be done to hardening such a router.

This bachelor's thesis project will examine how to implement several different known solutions, such as configure filtering for IPv6 traffic and how to lockdown services in order to harden common routers. The focus is on the perimeter router between the Internet and the Local Area Network (LAN).

1.4 Problem Definition

The need for protection is obvious, but the cost of security can be a concern, along with how to implement the security in an existing network due to a lack of knowledge about relevant threats. Since there are both old and new threats that the IPv6 network will be exposed to, a threat analysis needs to be done in order to gain a better understanding of the threat picture and to ensure that we protect ourselves against the relevant threats to a home or SME IPv6 network.

While common pre-built routers and firewalls have received much criticism [3], they also tend to be an expensive **and** imperfect solution. However, as will be presented in the thesis the security needs for a home or SME network can be solved with simple home networking equipment, thus decreasing cost *and* providing higher security.

A variety of open source router software offer an opportunity to re-use routers and existing computer's by giving them increased functionality, potentially providing a better and more cost effective solution to achieve perimeter security.

Achieving network security means configuring and implementing known solutions to known threats. Identifying these threats and implementing solutions is non-trivial for the ordinary home or SME network user, so there is a need for a structured method to implement the appropriate security mechanisms. This thesis intends to offer and document such a structured method.

1.5 The intended audience

This thesis is intended for an average to more advanced network user who wants to know more about IPv6 security, specifically how they can harden their network perimeter security. Small scale users along with networking enthusiasts will also get an overview of how to use their existing hardware for this new purpose.

1.6 The limitation of this bachelor thesis project

The thesis will **not** cover basic network knowledge, readers are referred to one of the standard internetworking textbooks. However, reading this thesis will not require special expertise. References will be used to point to more specific knowledge as needed. Furthermore, LAN security threats that are not related to perimeter security or in-depth client security on the IPv6 network will not be covered since there already is good documentation of these in other sources. Details of any wireless interface of a router that are not specifically related to IPv6 or perimeter security will not be covered in this thesis.

This page intentionally left blank

Chapter 2

2 Background

This chapter describes what others have done in the form of threat analysis and what protection mechanisms are currently available. The chapter is divided into an in-depth overview of IPv6 along with the network outside and then an inside security perspective.

2.1 Internet Protocol Version 6

IPv6 addresses are 128 bits long instead of the IPv4 32 bits, which provide more address space than available at the moment and are the main reason for users to migrate. The addresses are also hierarchically constructed to provide different classes of addresses. The generic way of dividing the 128 bits for global unicasting is in three sections as Figure 2.1 showing.

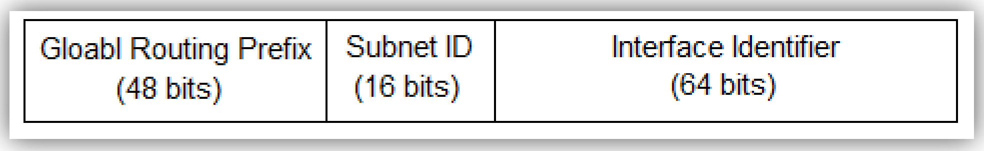


Figure 2.1: 128 bits IPv6 Global Unicast Address

The first section, Global Routing Prefix, is the prefix of the address used for routing, along with the first three bits that indicates that this is a unicast address. The subnet ID is used to identify an internal subnet at the destination network. The interface identifier is used to identify the network interface on a particular host. The IPv6 header has fewer fields than the IPv4 header because some fields were removed. The next header field is used together with extension headers. See Figure 2.2.

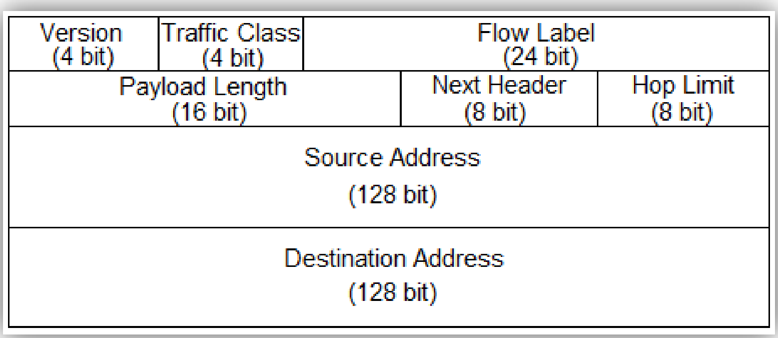


Figure 2.2: IPv6 Header format.

Extension headers, allow multiple extension headers to appear between the main IP header and the IP payload. This is called header chaining/linking. The next header field identifies the next extension header, as shown in Figure 2.3.

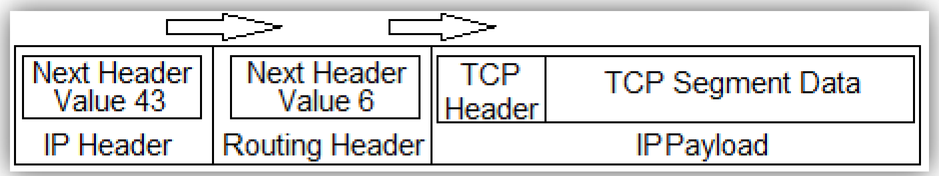


Figure 2.3: Illustrating the use of extension headers, called header chaining/linking.

When using extension headers together with next header values to identify different extension headers, the next header value 43 refers to a header called the “Routing Header”, see Figure 2.4. This can be used for a function similar to IPv4’s “lose source routing”, i.e., listing one or more intermediate nodes to be visited along the path to the destination [24].

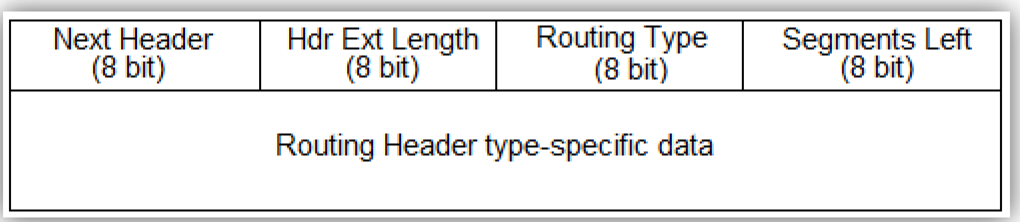


Figure 2.4: IPv6 Routing Header format.

However, there is a Type 0 Routing Header referred to as “RH0”. This routing header carries a list of transit IP addresses that the packet **must** visit. The RH0 mechanism uses the “segments left” field to point to the next IP address within the “routing header type-specific data” field that replaces the destination field in the main header. Figure 2.5 shows the RH0 format. This Type 0 Routing Header causes security concerns and will be discussed in section 2.3.3.

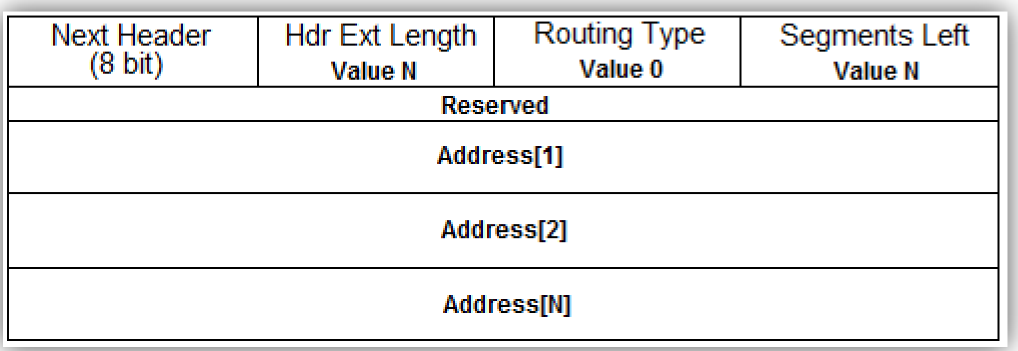


Figure 2.5: Routing Header Type 0 format.

Interesting features of IPv6 and the security support in terms of IPsec (i.e., authentication and encryption of extension headers,) and other features. This type of network layer security is between end hosts or routers and secures the communication by encryption. However, it needs to be configured between the two end nodes, thus it is not used for ordinary traffic via a home router.

ICMPv6 is one of the most essential parts of IPv6 since it handles the connectivity. ICMP messages are divided into two main categories, error messages with Type values 0 to 127 and information messages within values 128 to 255, only a part of the values are defined yet. The Code field defines the sort of message of a particular Type. An important function of ICMP is Neighbor Discovery mechanism to gain LAN connectivity.

Looking at the minimum requirements for connectivity according to the RFC for Filtering Recommendations [19]. This RFC clearly states what needs to be processed in order to establishing and maintain IPv6 connectivity, specifically the error messages shown in Listing 2.1.

Listing 2.1: Minimum ICMPv6 messages required for connectivity at WAN side of an IPv6 host or router.

- Type 1 Destination Unreachable: No connection is possible
- Type 2 Packet to Big: Needed for the Path MTU discovery
- Type 3 Time Exceeded – code 0: Which means the packet Time To Live exceeded
- Type 4 Parameter Problem: Header error of a packet

Error messages such as “Destination Unreachable” and “Packet Too Big”, needed for MTU discovery, are necessary for IPv6 connectivity [14]. Furthermore, echo messages are essential in order to maintain communication for tunnels. Figure 2.6 shows the ICMPv6 message format. Additionally, ICMP handles multicast connectivity. Some security aspects of the ICMPv6 protocol discussed in section 2.3.2 at page 10.

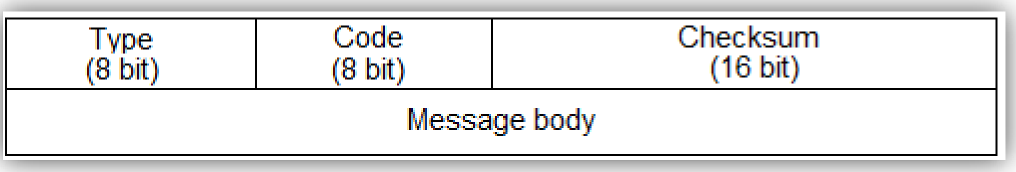


Figure 2.6: ICMPv6 packet format.

2.2 Threats overview

In this section we states what we are seeking to guard against. Attacks and threats come in different ways and via different TCP/IP stack layers, both originating from outside of the border router and from the inside LAN. IPv6, dual-stack, and IPv4 routers all face the same threats at the application layer. Here we will present basic attack terminology and what the reader should keep in mind when looking at different defense solutions.

2.2.1 Outside attacks

Threats exposing the router’s Internet port from the border router’s perspective include many well-known attacks that target the router’s application layer regardless of the Internet protocol used [8]. However, there are some attacks and security flaws specific to the implementation of the firewall, along with IPv6 specific threats which are clearly in the scope of this thesis. When connecting a network device to the Internet the device will be exposed to different threats, such as botnets with malware and malicious hackers making targeted attacks.

2.2.1.1 *Reconnaissance and information gathering*

Scanning either for available routers or services on a specific router is the first step of an attack. This is often the first sign of malicious behavior. These scanning attempts are designed to gather information that can be used to identify potential targets or weaknesses of a particular target.

Vulnerability scanners such as Nessus[1] can be used to target a specific router as a first penetration step. Given sufficient information about this router, the attacker can make use of vulnerability databases that list potential security weaknesses of specific services. For this reason we need to consider which open ports and services are available and which *should* be available on the router in order to avoid attackers exploiting well known vulnerabilities of the router [13].

2.2.1.2 *Denial of Service attacks*

Denial of service (DoS) attacks occur when the router’s resources, such as memory and processing or forwarding capacity, are consumed by the attacker, decreasing service to legitimate traffic. It is extremely hard to prevent and protect against a large distributed denial-of-service attack, since we cannot decide upon or affect the amount of traffic that is directed toward us.

2.2.1.3 *Outside router break-ins*

Unauthorized access to the router itself from the Internet side can take place as a brute force attack, in order to gain access to a service simply by making a series of login attempt with all possible passwords until gaining access. A management service that is enabled for outside remote password authentication is a perfect target for brute force attacks.

2.2.2 Inside attacks

There are threats that target the inside LAN by means other than forcing their way through the border router. These attacks include social engineering attacks and Trojans. All of these types of

[1] Nessus vulnerability scanner, http://www.tenable.com/products/nessus

attacks seek to exploit security weaknesses of the network structure in different ways, along with flaws in application's implementations. Attacks originated outside but using the inside LAN to target the router from an inside perspective occur mainly at the application layer.

2.2.2.1 *Inside-out attacks*

Trojans that utilize a tunnel to open channels and ports from the LAN side to the Internet are a popular inside-out attack. This technique not only is used with tunnels, but a reverse connection can be initiated from the LAN side to open path through a stateful firewall to allow malicious traffic direct access to the LAN client. Another version is cross-site request forgery (CSRF) that exploits the router's web based configuration from the user's web browser.

Users and software developers are another aspect of inside threats, intentionally or not, bypassing firewalls by using different tools and techniques that can compromise security, in order to make the software more user-friendly.

2.2.2.2 *Tunnels and encapsulated traffic*

Tunnels, opened from a LAN client to an external computer, can allow malicious traffic to pass the firewall undetected because the traffic is encapsulated inside the tunnel and not inspected [5]. This is a popular dual stack (using both IPv4 and IPv6) or IPv4-only issue since clients comes with default IPv6 support and if the LAN uses only IPv4, then the encapsulated IPv6 traffic will not be expected, hence there will be no security mechanisms implemented to deal with it. Of course an attacker can encapsulate IPv6 traffic in IPv6 over the LAN. Not to be forgotten is that in an IPv6 world there could be IPv4 traffic that is encapsulated and sent over an IPv6 network [18], in order to bypass IPv4 security.

2.3 IPv6 specific threats

In this section we will focus on attacks and security flaws specific to IPv6 along with a firewall implementation. This is the main focus of this thesis.

2.3.1 IPv6 address space

Since the IPv6 address space is quite large the risk of a global host scan is not a major concern, because it would take years to do even a ping scan, i.e., to identify a live host within a given IP range, across the entire global Internet. However, DNS servers can be used by an attacker to collect addresses, these specific addresses could easily be scanned for vulnerabilities.

2.3.2 ICMPv6

ICMPv6 can be used to gather information about a specific router in order to learn more about the network's structure [12]. The Echo Request/Reply is one of the common ICMP messages used for network probing. Furthermore, blocking ICMP is not an option since IPv6 connectivity relies on it, but blocking specific messages that is not used and not required is a better solution. However, the Echo messages is often necessary for tunnels, which means if the router we will configure as an perimeter defense needs to use a tunnel broker for IPv6 connectivity, then echo messages are needed and cannot be blocked. However, using iptable mechanisms for packet filtering to drop outside probe packets can make the router invisible for the majority of scans, offering a simple and effective method. Make sure to not only block the packets, because the blocking method may send a response packet back to the scanner, rather you want to simply drop the packet to avoid providing any information to the attacker.

ICMPv6 can be used in DoS attacks by sending a stream of error message to the targeted machine, since the receiving host needs to process these error messages this creates an increased load on the machine's resources. Furthermore, from a multicast perspective a multicast packet can be sent with the unknown destination option marked as mandatory with a spoofed source address of a multicast source host, thus triggering other nodes to send an ICMP parameter problem message to the source address which results in a lot of traffic [14].

From a border router's perspective securing the LAN from the Internet by blocking all multicasts would be sufficient, but this will cause problems with legitimate multicast services. In order to prevent DoS attacks, we can use rate-limiting to determine how much ICMP traffic we want to allow or to limit how much we generate. This limiting function is a part of ip6tables which is easy to implement via ICMPv6 rules, that will be discussed in detail on page 32.

TCP specific denial of service attacks exists where an ICMPv6 error message is used to trigger a host to tear down an active TCP connection [14, 25]. This is not a common problem, but suggests the need for error message validation.

Consider limiting the common ICMPv6 message types, by determining which messages we need to permit to pass through the Ingress filter. As mentioned in section 2.1, ICMPv6 massage types 1,2,3, and 4 are needed for connectivity at the outside along with echo messages if tunnels is needed.

2.3.3 Type 0 Routing Header

There is a method that may allow scanning by using the Type 0 Routing Header. In this method the outside attacker already knows one internal host and specifies more than one destination IP address which causes the receiving host to forward the packet to the next LAN client in the RH0 IP address list. When the final destination node is reached it can directly respond back to the attacker's source address, and another LAN node is discovered.

For example, a host that would have been out of reached for direct communication, due to firewall access control lists, might be accessed via a transit node, such as a public web server connected to the LAN. When the final destination node is reached it can simply respond directly back to the attacker's source address. See Figure 2.7.

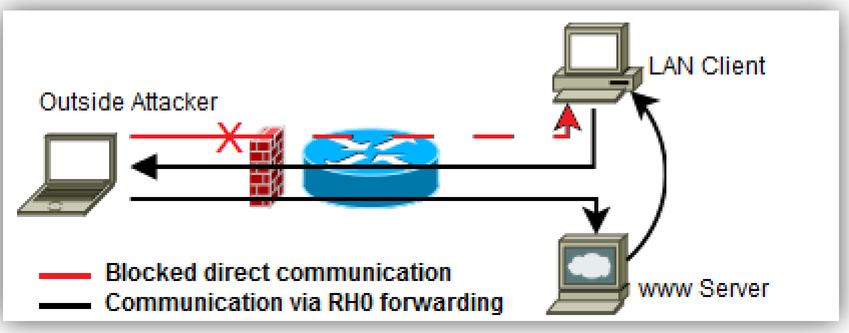

Outside Attacker
LAN Client
www Server
Blocked direct communication
Communication via RH0 forwarding

Figure 2.7: Bypass firewall using Routing Header Type 0.

Therefore, this type of traffic should be blocked by the perimeter router; either by blocking RH0 from the outside or blocking the LAN client from responding back to the attacker [17]. Another exploitation via RH0 is as part of a DoS attack, where a single RH0 may have the same intermediate address in its list multiple times making the packet travel back and forth between two nodes within the list. This will cause congestion on the path between the nodes, potentially negatively affect the LAN since the congestion is on the entire link [6, 17]. See Figure 2.8.

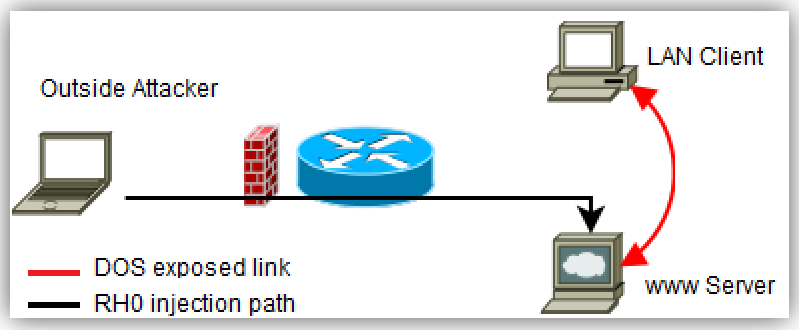


Figure 2.8: Denial of Service attack using Type 0 Routing Header.

Blocking all packets containing Routing Headers is not a desirable solution as this could have a negative impact on future development of IPv6. The solution is for all IPv6 nodes to disable support for RH0, which means if the web server does not support RH0 it will not be forwarded to the LAN client but there is yet no guarantee that all nodes will block RH0. Therefore ingress filtering in firewalls should block packets containing Type 0, but not disable other types of routing headers. Furthermore, routing headers of Type 0 are no longer required for IPv6 implementations in any way [6].

2.4 Application layer threats

In this section we focus on which threats are isolated to the application layer and determent if we can counter them at the border router.

2.4.1 Management services / unauthorized access

The effectiveness of a brute force attack lies in the choice of the complexity of the password and the time it takes for each login attempt. Another password cracking attack similar to a brute force attack is a dictionary attack, where a specific dictionary is used as a list of potential passwords in order to speed up the break-in. The effectiveness of a dictionary attack lies in the password list itself since the correct password must be in the list for the attack to find the matching password [10]. The protection that is generally proposed against such attacks is to limit the number of attempts and to ensure that every user uses a complex password[2] that is unlikely to be found in a dictionary, hence making the attack very time consuming since the attacker must resort to a brute force attack. It is worth noting that some of the resources of the router can be consumed by processing all of these login attempts and logging the failed attempts. Also note that blocking logins for some time after N failed attempts can be used as a DoS attack preventing legitimate users from being able to remotely login to this router.

Cross-Site Request Forgery is a typical application layer attack that exploits the router's web based configuration if it has a predictable structure[3]. This attack needs to be combined with social engineering, which means the user is tricked to perform an action enabling the attack to change the configuration of the router, such as lower its security or open security holes [10].

Since the countermeasures to defend against application layer attacks, such as cross-site request forgery, are within the application layer itself regardless of the version of the Internet protocol used, the recommended action is to revise the browser security at the clients; for example, avoiding saving password history and using add-ons such as "NoScript", that blocking execution of scripts locally, for better cross site scripting (XSS) control [11, 37].

Furthermore, there are a variety of protocols used for remote management protocols such as telnet, secure shell (SSH), simple network management protocol (SNMP), hypertext transfer protocol secure (HTTPS), etc. These can all run on top of IPv6. It is important that, if there is a need for them, these application protocols should be restricted based on authentication or simply disabled if they are not going to be used. Usually there is a tradeoff between security and service, hence disabling remote management is a recommended security countermeasure to hardening your network device but this makes remote management more difficult. In general you should try to minimize the potential for attacks by disabling services that you are not going to use. However, remote management provides faster incident response when you do not have local access at the time of an incident. For this reason, some sort of remote management may need to be enabled, but our recommendation is to (1) use the most secure mechanism possible, such as permitting SSH only from an internal machine – thus for remote access you have to authenticate yourself to this machine and then SSH to the router/firewall;

[2] Complex passwords are long and include upper and lower case letters, numbers, and special characters.

[3] All common routers have predictable structure since they have a well documented configuration interface.

(2) to permit SSH only from specific external machines; or (3) permit SSH from any machine, but utilize a strong password and/or two factor identification.

For home and SME networks, remote management services should be disabled because the router configuration probably will not change very often due to the simplicity of the network. Hence one can avoid many forms of attacks. There will still be the issue of what restrictions to put on the remote administration from the LAN side of the router/firewall.

Running services on different ports than the default port will confuse port scans, since the result is a false-positive, scanners often check which ports are open and then compare the responding port number with a database of default port services and then present a possible running service on a system, providing information to continue with vulnerability searches for that particular service [13]. Running a necessary service on a non-default port will slow the information gathering, but this works best for non-public services that do not need to be running on a well known port number. However, these ports are relatively few, making this security countermeasure less effective.

2.4.2 Encapsulated and reverse traffic

Encapsulated traffic can be used for getting covert malicious traffic through a firewall. Trojans that utilize a tunnel to gain control over the LAN computer are common and hard to filter since there is lots of legitimate tunnel traffic. See Figure 2.9 for an example. To counter encapsulated traffic, the best approach would be to deny any tunnel traffic within the LAN, but this will cause problems with legitimate traffic that uses such tunnels, for example HTTP over transport layer security (TLS), session initiation protocol (SIP) over TLS, etc.

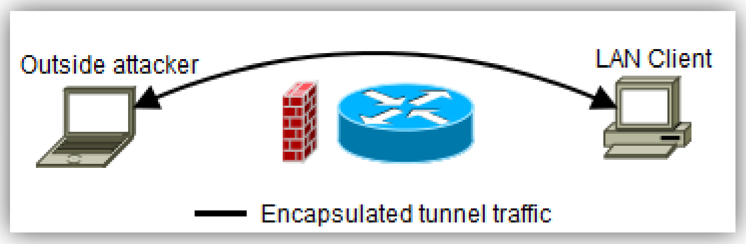


Figure 2.9: Tunnel traffic bypassing router firewall.

Reverse connections is another common way to bypass firewalls by initiate the connection from inside the LAN making the firewall associate the malicious traffic with a legitimate LAN user, causing the firewall to accept the connection because it originates from the inside. See Figure 2.10.

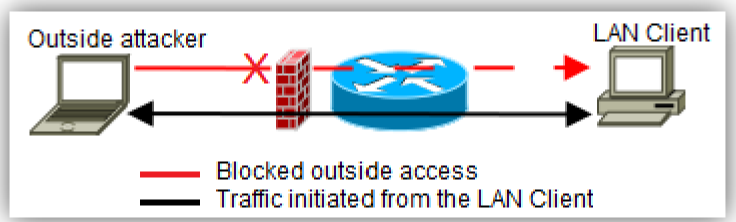


Figure 2.10: A malicious inside-out connection.

Countering this type of inside-out attacks requires egress filtering, which is monitoring and restricting outbound traffic, from the LAN hosts at the application layer [15]. However, application layer filtering is uncommon for ordinary home routers.

2.5 Defending the router

This section describes the solutions and what is needed to prepare your router. Remember when configuring or setting up a device for security purposes it is important that you not expose the device to any traffic or insecure media before the configuration and security mechanisms are correctly installed.

2.5.1 Tradeoff between service and security

The rule of thumb is to deny everything and explicit allow the minimum set of that you need. In the perfect security world there is no freedom or flexibility of services or usage. Implementing services potentially lowers the security. Users want more flexibility and greater freedom, even as software developers use protocols that are more firewall friendly, which means protocols that are usually allowed through the firewall policies therefore causing fewer conflicts and minimizing configuration of the firewall. HTTP is one of these ordinary protocols that is used for all kind of communication purposes simply because it is user friendly and almost always is allowed through the firewall. Considering both the inside and the outside of the firewall is important because there are outside **and** inside-out threats as described in the previous sections.

2.5.2 Hardware and Firmware

Routers have different types of hardware capabilities, for the purpose of implementing IPv6 support and packet filtering we need sufficient memory to hold this extra functionality. Firmware that is the hardware integrated software running on the router. This firmware is usually stored in a flash-memory or read only memory (ROM).

There is open firmware that provides the desired functionality for common router platforms. This thesis uses the dd-wrt firmware, which is an open source Linux implementation [30]. Today Linksys has several models of routers that come standard with dd-wrt firmware. Recently Buffalo has decided to ship models with the dd-wrt firmware [26]. This means that home and SME networks can chose dd-wrt as a standard, and hopefully more vendors will follow in the near future. However, you may be able to reflash (i.e., reinstall), your router with the dd-wrt firmware. It is important to make sure your specific router is supported by the firmware you going to use. dd-wrt supports the most common home routers and has a database of the supported hardware that also gives tips on how to install the firmware on each specific router. The exact procedure depends on the hardware.

Memory space is another concern because if the router does not have sufficient space the functions will not fit in the router. However, some routers have USB ports that can be used to extend the memory and dd-wrt has Samba/CIFS support enables the router to access files over the network significantly extending memory space [21]. Keep in mind that this extended memory space usually just stores extra functionality and scripts, as the firmware itself needs to be stored onboard on the router in order to function properly.

When it comes to configure the router, you need to know the general architecture of how the firmware works with the hardware in order to use the interfaces correctly, you should keep in mind there is a difference in firmware versions and particular hardware, but the general concept is the same. There are three different parts of a router: (1) central processing unit (CPU) that handles the virtual local area networks (VLAN) tagging and bridges between the other two parts; (2) the switch that is the outside visible physical ports on the router that is divided into two VLAN as including both the wide area network (WAN) port that usually is the higher VLAN number and LAN ports that usually is the lower VLAN number; and (3) the wireless access point, usually the higher value Ethernet "eth" interface. See Figure 2.11 for an illustration of the general architecture.

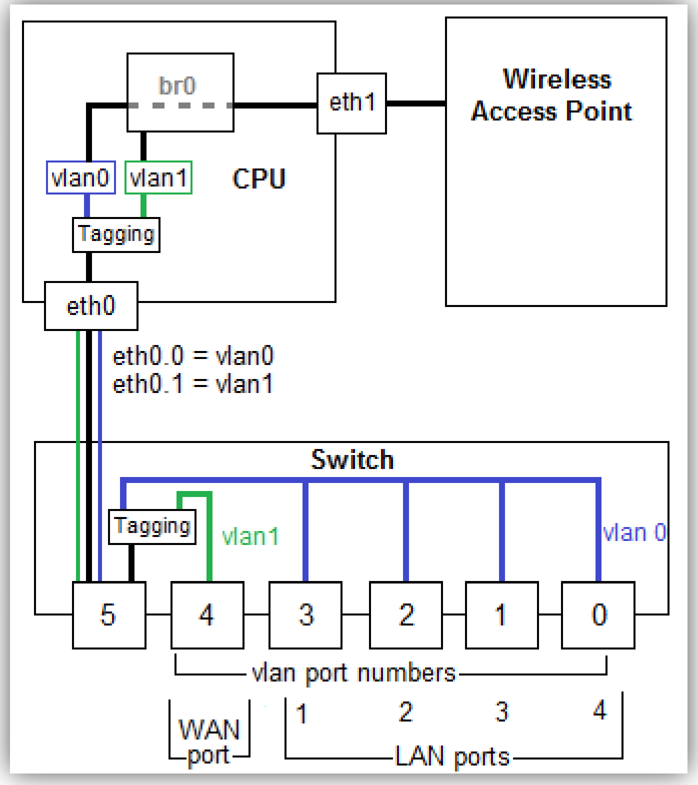


Figure 2.11: Hardware architecture.

2.5.3 Startup scripts

Startup scripts are used to automatically run some predefined commands that usually are not persistent between device reboots. Using startup scripts, specific user configurations can be stored and loaded in the router without any reconfigurations, making all of the administration easier between reboots and power failure.

Permanent settings are stored in the non-volatile random-access memory (NVRAM). This is basically a RAM memory that retains data even if the router is powered off. This is where the settings of the web Graphical User Interface (GUI) are stored along with the NVRAM startup script [34].

Furthermore, the NVRAM is limited to capacity and in this case creating more than one script, the preferred method is to use shell scripts stored in a Journaling Flash File System (JFFS/JFFS2) that we can use to read and write files on the router that is persistent between reboots [36].

The startup script must have the file extension of ".startup" and the kernel is configured to search for startup scripts at specific locations in the router, therefore we need to store our scripts in the "/jffs/etc/config/" folder in order for the kernel to find them. The scripts must be marked as executable. This is done by using the "chmod" command to change the permissions on the file [35]. Furthermore, you do not need to reboot the device for the startup script to take effect, you can simply run the script with "./*.startup" immediately as it is an ordinary script.

Notice, if you chose to use an SCP program to transfer the startup script that is written on a different machine. Different text editor software may attach unknown characters to your script that need to be edited once the script is transferred to your linux router. Using the "VI" text editor on the router is a way to edit or create startup scripts. Coding 2.1 shows an example of a startup script.

Coding 2.1: Example of how to make a startup script.

```
root@DD-WRT:~# mkdir -p /jffs/etc/config
root@DD-WRT:~# cd /jffs/etc/config
root@DD-WRT:/jffs/etc/config# vi start.startup
#!/bin/sh
# set static IPv4 WAN address
ifconfig vlan2 10.10.1.2 netmask 255.255.255.0

root@DD-WRT:/jffs/etc/config# chmod 700 start.startup
root@DD-WRT:/jffs/etc/config# ./start.startup
```

2.5.4 Introduction to packet filtering

Packet filtering, also known as access control lists (ACL), is the most common and basic firewall approach. The purpose is to protect the users connected to a specific LAN network interface from the outside Internet WAN threats. Most routers today have packet filtering built-in, but the filters are hard to configure. When it comes to realizing network security, a packet filtering firewall is a preferred way to securing the network against known types and patterns of traffic. This functionality is implemented by many of the commercial and non-commercial firewalls sold today [16].

The filtering process examines traffic entering or leaving the network at the border router by passing the traffic through a packet filtering mechanism. The goal of this filtering is to accept or reject each packet. Packet filtering is often called packet inspection where the inspection refers to looking only at the packet header, rather than the entire packet which includes the payload. Deep packet inspection (DPI) refers to filtering that looks at the contents of the IP packet. DPI can be limited to transport and other headers or can also look at the actual application payload.

Within the IPv6 packet header, shown in Figure 2.2, there is a source and destination address along with next header, these three values are often the basis for the filtering decision based upon predefined rules for how to handle these three fields.

There are two types of packet filtering: stateless and stateful packet inspection. Stateless refers to making a decision based only on static rules and *without* regard to any *previous* packets. The flaw with stateless firewalls is that they can easily be fooled when filtering on a packet by packet basis. For stateful packet inspection there needs to be a memory function so that the filter maintains state for each connection from when a session is established to when it is closed. This state information includes the associated IP addresses and port numbers.

Today the trend is to use smarter firewalls that are stateful. In addition, many of these routers can be used to implement policies such as to only accept inbound connections that are associated with an already established outbound connection, for instance a TCP connection initiated by a web browser running on a host attached to the LAN [16].

There are different packet filtering alternatives in different systems, some of the proven IPv4 mechanisms are “Netfilter”, “IPfilter”, and the Linux “iptables”. However, there is a Linux IPv6 packet filter, “ip6tables”, based on the proven IPv4 version of iptables. This type of packet filer can be configured to perform both stateless and stateful inspection [33, 38, 39, 40].

2.5.4.1 *Filtering mechanism*

The filtering mechanism is one of the important concepts necessary to understand this thesis because filtering is the foundation of providing protection. Therefore we will take a closer look at how the filtering mechanism works, in this thesis we use Linux iptables in our dd-wrt firmware to implement good basic protection.

We start by describing how packets traverse the router via the packet filer. The iptables are divided in different chains containing rules. The main chains that hold the filtering rules are the INPUT, OUTPUT, and the FORWARD chain. These chains exists in tables, and for the IPv6 version of iptables there are three tables which are (1) the raw table is used for marking packets for connection tracking for stateful filtering; (2) mangle is used for specialized packet alteration; and (3) the filter table is mainly the default table and holds those filtering rules that we will focus on. Each IP packet traverses the IP filter by visiting each of these tables following each chain and matching against the rules in the filter table [26]. Furthermore, as we see in Figure 2.12, a routing decision is made just after the PREROUTING chain into two different categories: packets that are sent to the local host’s IP address or packets having another host as the destination address. Important to note is that a packet either passes through the FORWARD or the INPUT chain, then depending on the packet it may very well continue from the local process to the OUTPUT chain and thus pass the router. This means filtering rules need to be configured for **both** of these paths!

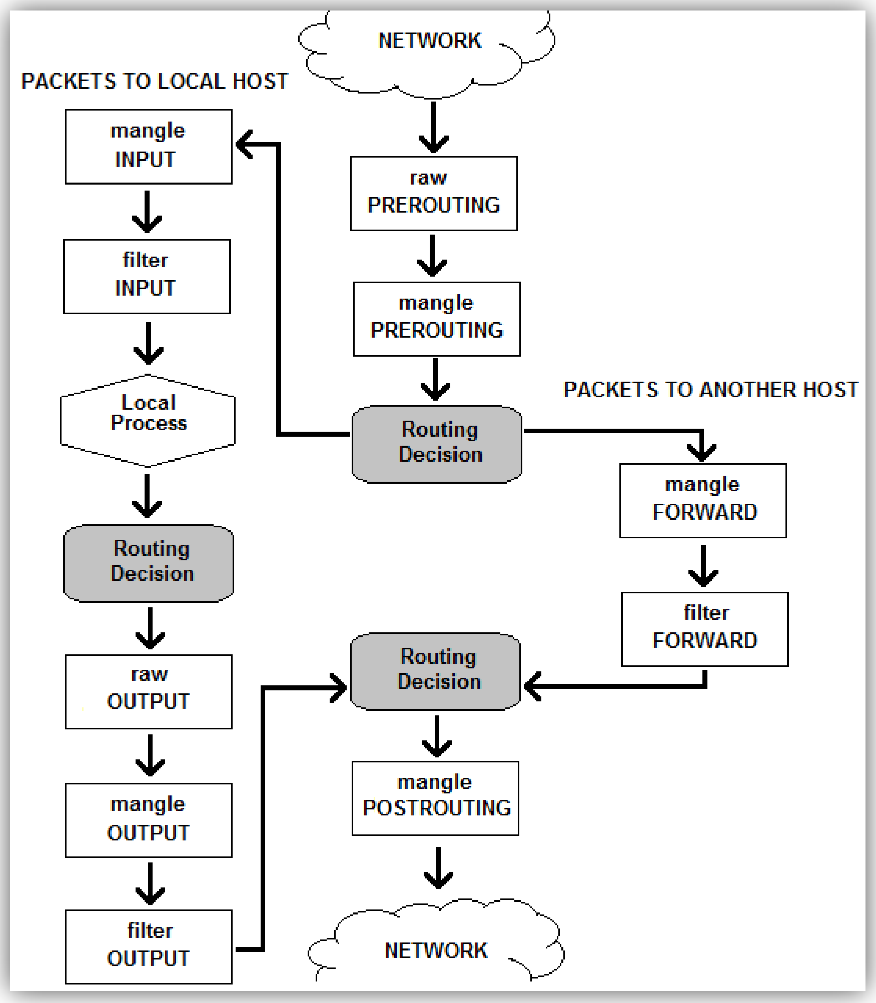


Figure 2.12: The packet flow when traversing through the packet filtering.

New and additional chains can be specified and added to a default chain simply by adding an jump rule to an existing chain making the rule matching process temporarily follow the new chain in-between the rules in the default chain, see Figure 2.13.

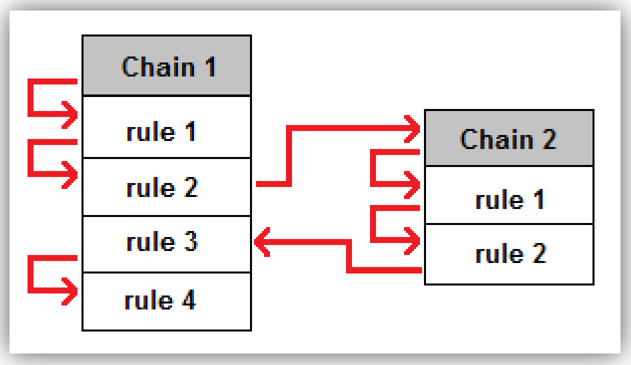


Figure 2.13: Adding a new chain.

The states for stateful packet filtering are another important concept that needs to be understood in order to configure the stateful feature which is recommended. The connection tracking is done during PREROUTING, except for locally generated packets that are marked by the OUTPUT chain. The different states that the packets are marked with are one of these four states: NEW, ESTABLISHED, RELATED, or INVALID. The NEW state is used when we see a packet incoming for the first time regardless of whether it originated from the WAN or LAN side of the firewall. The packet has no connection to other flows or streams of packets. The ESTABLISHED state is when traffic is going in both directions, which means packets are marked as belonging to a flow or stream of packets. See Figure 2.14 illustrates the NEW and ESTABLISHED states.

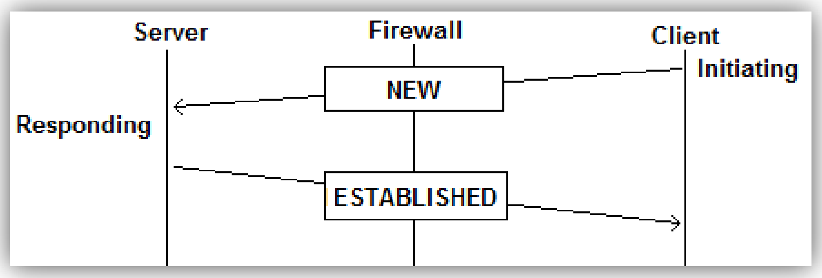


Figure 2.14: The relationship between the firewall states NEW and ESTABLISHED.

The RELATED state is intended for traffic belonging to an already ESTABLISHED connection, such as ICMP messages related to a specific connection or complex protocols that need more than one connection flow to work, such as FTP. See Figure 2.15 for an example of ICMP in the RELATED state.

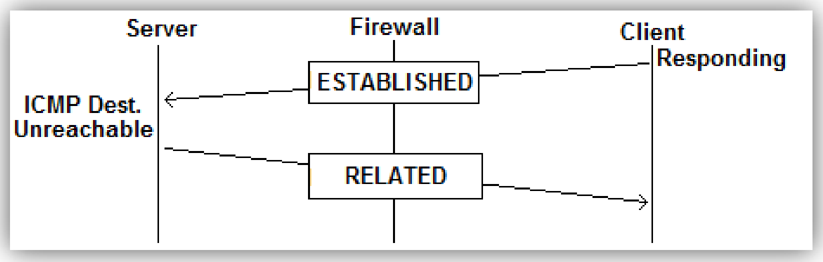


Figure 2.15: The relationship between two connections using the RELATED state.

2.6 Testing tools

The tools used for testing are the Nmap network scanner version 5.51 and the Nessus vulnerability scanner version 5.0.0 Home Feed [41]. These tools have IPv6 support and are popular for both network administrators and malicious hackers. The two tools works in a similar way, but use different databases and the Nessus scanner includes various tools, such as brute forcing, backdoor detection, DoS vulnerabilities, etc., including network scanning [31]. Both of these tools are used with a default configuration over a directly connected link, the only specific configuration is that Nessus is used with the “External Network Scan” set at the default scan policy [32].

This page intentionally left blank

Chapter 3

3 Method

This chapter will cover the general concept of how to turn an ordinary home router into a more secure device with a firewall for IPv6. The chapter offers a step by step description. We also describe the implementation and configuration of the firmware and features necessary to give an existing router more functionality. The steps in this chapter can be done in varies ways depending on what hardware and firmware version you have. However, the selected steps used in this thesis are the most general sequence of steps compatible with most hardware and firmware versions. The implementation will be done with different startup scripts to make the changes in a simple and persistent manner.

3.1 Equipment list

The router used in this project is an Asus RT-N16. Here after it will simply be referred to as Asus. Hardware, firmware, and additional software used with this router are listed in Listing 3.1.

Listing 3.1: Router specification.

```
Hardware: Asus RT-N16
Firmware: DD-WRT v24-sp2 std 2010 Release: 08/07/10 (SVN revision 14896)
Kernel: Linux DD-WRT 2.6.24.111 #1982
pcap library:       version 0.9.4-1 mipsel.ipk
tcpdump:            version 3.9.4-1 mipsel.ipk
ip6tables:          version 1.4.0-1 mipsel.ipk, kmod 2.6.25.20 brcm47xx-1 mipsel.ipk
```

The PC client used both for LAN and WAN activity is referred to as “the PC”. Listing 3.2 gives the details of this PC.

Listing 3.2: PC specifications.

```
Operation system: BlackBuntu: Ubuntu 10.10 – the Maverick Meerkat
Nmap: version 5.51
Nessus: version 5.0.0 - Home Feed
Wireshark: version 1.6.7 (SVN Rev 41973)
```

3.2 Implementation network map

When it comes to securing and implementing security via the router, it is important to not expose the device to any threat before the security mechanism is in place. This first network map is also the first setup for flashing the firmware. We start the implementation by connecting a PC to the LAN port number 1 on the router. The simple network setup is shown in Figure 3.1.

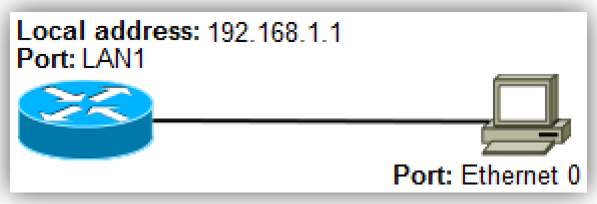


Figure 3.1: Router security implementation network map.

3.3 Firmware upgrade

First we need to install the dd-wrt firmware in the router. This is the most critical part of the whole implementation, since doing this step wrong could lead to total disabling your router, also known as “bricking” the device. Therefore the recommendation is to visit the dd-wrt database and installation documentation section [29]. The web page has flashing instructions for each and every type of supported hardware. It is also important to choose a firmware version sufficient to support IPv6 features and of course compatible with the router hardware you have.

In this thesis project, the firmware used to flash the Asus is the mega firmware version for Asus[4]. The steps are in Listing 3.3.

Listing 3.3: Flashing the ASUS RT-N16 router.

1. Disconnect all cables, attachments, and connect directly with a PC using an Ethernet cable to the router’s LAN1 port, see Figure 3.1. Then browse the router using the default address to the web GUI at http://192.168.1.1.
2. Factory reset from the web GUI.
3. Flash the router ROM using the downloadable Asus Firmware Restoration Utility[5], then wait for the router to reboot. There are often restoration utilities from the manufacturer that is the easiest way to flash regardless of if there is a flashing function in the web GUI menu. In my case flashing did not work with the dd-wrt firmware.
4. After the reboot, reconnect to the web GUI and set temporarily login settings such as user: root and password: root. This is to confirm that the flash was successful at an early stage.
5. In order to clear memory, we clear the NVRAM of the router by “telneting” into the router using an telnet client like Putty[6] to the default IP address: 192.168.1.1 and the temporarily login settings from step 4 to login. Then clear the NVRAM using the command: **erase nvram** followed by: **reboot**. After rebooting the change password screen will appear. The

[4] Asus firmware with IPv6 support: “dd-wrt.v24-14896_NEWD-2_K2.6_std_usb_ftp.bin”.

[5] Asus utility download: http://support.asus.com/download.aspx?SLanguage=en&m=RT-N16&p=11&s=2&os=30&hashedid=WAa6AQFncrceRBEo

[6]Putty telnet and SSH client: http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

previous login settings are erased. Furthermore, SSH is not enabled in the dd-wrt firmware yet, and that is why Telent is used.

6. This last step seems to be a repetition of step 5, but according to the dd-wrt documentation this step is important and should be done before every firmware upgrade. In order to erase NVRAM and restore the dd-wrt default state do a “Hard Reset” called a “30-30-30 reset” is a critical type of reset where we refer to the dd-wrt documentation [28] to be sure how to do a hard reset on your particular hardware. In this thesis using the Asus the hard reset is done by:
    - **a.** Press and hold the WPS button on the back for 30 seconds.
    - **b.** Do not release the WPS button, unplugging the router and hold reset for another 30 seconds.
    - **c.** Plug the router back in without releasing the WPS button a final 30 seconds.

At this point you should have a fully functional dd-wrt router, supporting IPv6 and ready to be configured. If the router has become bricked Then a recovery processes that depends on hardware version is needed. My recommendation is to consult the dd-wrt documentation for your particular router [30]. Furthermore, after the firmware is successfully installed in your router, you need to disable the wireless module or encrypt it since it broadcasts as a non-protected access point by default.

3.4 Experimental Environment

To ease access and management of the server during the experimental setup **and** in order to fully understand and demonstrate the purpose of the firewall we will add the popular SSH service on the LAN side. This service is more secure since the traffic in encrypted between the router and the PC, while telnet is not. The SSH service is popular and encrypted services are normally a better security practice. However, the recommendation of using SSH was discuss in section 2.4.1 about how to properly secure SSH. Enabling SSH is done via the web GUI of the router by simply logging in, clicking “Enable” at the “Secure Shell” section under the “Services” tab. Do not forget to disable telnet via the same tab. Finally click “Apply Settings” at the bottom of the page. At this point, we have actually enabled an SSH server on the router, and since the default firewall block the SSH service on the WAN port, SSH is now only a LAN service, at least in terms of IPv4. Additionally, we will need to enable JFFS/JFFS2 to store files, this can be found at the “Administration” tab. The preferred method is to login to the router using SSH and issue the commands shown in Coding 3.1. The reboot will take longer than usual, but that is normal when enabling JFFS.

Coding 3.1: Asus router configuration for JFFS.

```
root@DD-WRT:~# nvram set jffs_mounted=1
root@DD-WRT:~# nvram set enable_jffs2=1
root@DD-WRT:~# nvram set sys_enable_jffs2=1
root@DD-WRT:~# nvram set clean_jffs2=1
root@DD-WRT:~# nvram set sys_clean_jffs2=1
root@DD-WRT:~# nvram commit
root@DD-WRT:~# reboot
```

3.4.1 Packet capturing

In order to debug and analyze what type of packets arrive on the router’s WAN port, we need to install a packet capture program. Since we use have limited space and using dd-wrt firmware the preferred and supported choice is “tcpdump” [42]. The installation process can be done in various ways, but we prefer to manually download the ipk libpcap[7] and tcpdump[8] package since we do not want the router to be exposed to the Internet yet. Use an SCP transfer program over SSH to upload the packages to the router’s “/jffs” folder. To install the packages you may need to do a forced installation to make the dependency check less likely to throw errors when you do the install. Coding 3.2 shows the commands used on the Asus router.

Coding 3.2: Asus router tcpdump installation.

```
root@DD-WRT:/jffs# ipkg -force-depends install libpcap_0.9.4-1_mipsel.ipk
...
Unpacking libpcap...Done.
Configuring libpcap...Done.
root@DD-WRT:/jffs# ipkg -force-depends install tcpdump_3.9.4-1_mipsel.ipk
...
Unpacking tcpdump...Done.
Configuring tcpdump...Done.
```

To perform analysis using tcpdump, we use tcpdump to capture WAN interface traffic and save the captured packets in a “.pcap” file in the “/tmp” directory of the router. The file can be transferred later using an secure copy (SCP) transfer program over to my PC where the file can be analyzed using “Wireshark” [43].

3.4.2 Isolated network

In order to test our firewall security without exposing the router, we created an isolated network where we simulate WAN traffic to the router performing different types of scans to investigate the filtering rules and the firewall’s behavior in these different scenarios. Static IP addresses were assigned for both IPv4 and IPv6 to the routers WAN port and to an outside PC. We keep the default configuration of the router as much as possible because we will investigate the difference between traffic over both IPv4 and IPv6 when using the default IPv4 firewall. At this point there is no IPv6 firewall.

With the PC still connected to the LAN side of the router (from the previous step), we assign a static IPv4 and IPv6 WAN address to the router’s WAN interface, if you are unsure of your hardware and which VLAN that is your WAN you can check the NVRAM since the WAN is associated with only two switch ports, as was shown in Figure 2.11. For the IPv6 support, in most cases you need to enable the IPv6 kernel module, then allow IPv6 traffic to be forwarded. We will now create the first startup script to handle the experimental environment. See Coding 3.3 for the lab environment script.

[7] File: libpcap package, http://downloads.openwrt.org/whiterussian/packages/libpcap_0.9.4-1_mipsel.ipk
[8] File: tcpdump package, http://downloads.openwrt.org/whiterussian/packages/tcpdump_3.9.4-1_mipsel.ipk

Coding 3.3: The Asus lab environment startup script.

```
root@DD-WRT:/jffs/etc/config# nvram show | grep port.*vlans
port5vlans=1 2 16
port3vlans=1
port1vlans=1
port4vlans=1
port2vlans=1
port0vlans=2
...
root@DD-WRT:/jffs/etc/config# vi lab.startup
#!/bin/sh
# set static IPv4 WAN address
ifconfig vlan2 10.10.1.2 netmask 255.255.255.0

# set static IPv6 WAN address with forwarding
insmod ipv6
ip -6 addr add 2001:470:27:c1c::2/64 dev vlan2
echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
root@DD-WRT:/jffs/etc/config# chmod 700 lab.startup
root@DD-WRT:/jffs/etc/config# ./lab.startup
```

Simulating WAN traffic will be done by connecting the LAN PC to the WAN port of the router and assigning both an IPv4 and IPv6 static address to the PC’s Ethernet interface that will communicate with the WAN port on the router. Commands used on the PC are shown in Coding 3.4. When the PC has connectivity to the router’s WAN port, then the network should look like Figure 3.2.

Coding 3.4: PC configuration for simulating WAN traffic.

```
root@PC/# ifconfig eth0 10.10.1.1 netmask 255.255.255.0
root@PC/# ip -6 addr add 2001:470:27:c1c::1/64 dev eth0
```

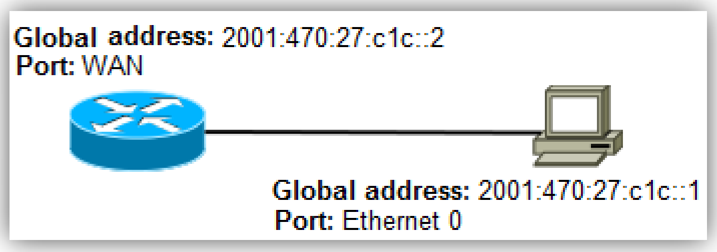

Global address: 2001:470:27:c1c::2
Port: WAN
Global address: 2001:470:27:c1c::1
Port: Ethernet 0

Figure 3.2: Router and WAN connected PC for simulated Internet traffic.

3.5 Testing the router with default configurations

The purpose of this section is to show that the default configuration along with the IPv4 firewall is not enough to protect against IPv6 traffic. By scanning the WAN port on the router we can investigate on a simple level how the IPv4 firewall and the router behave when exposed to IPv4 and IPv6 traffic.

3.5.1 Brief look at an IPv4 firewall

When looking at the iptable handling the IPv4 traffic, the first thing we see is that both the INPUT and the FORWARD chain have the policy ACCEPT, which is user friendly, but **not** good security practice. However, there is a last rule in the chains that matching all traffic to DROP, which is capturing all traffic that is not RELATED or ESTABLISHED from the outside by rule number 1. See Coding 3.5 for an sample of the iptables in the Asus.

Coding 3.5: Output of iptables, IPv4 firewall at the Asus.

```
root@DD-WRT:~# iptables –vL
Chain INPUT (Policy ACCEPT)
target  prot  opt  source     destination
ACCEPT  0     --   anywhere   anywhere      state RELATED,ESTABLISHED
...
DROP    icmp  --   anywhere   anywhere
...
DROP    0     --   anywhere   anywhere

Chain FORWARD (Policy ACCEPT)
...
```

The conclusion of looking at the IPv4 firewall is that it should block all traffic on the WAN port initiated from the outside, such as ping and SSH. However, this is not the case when it comes to IPv6 traffic.

3.5.2 Security testing of the IPv4 firewall

When scanning the IPv4 address with Nmap the default ports are filtered and do not respond, which is a good thing. The Nmap output is show in Coding 3.6.

Coding 3.6: Nmap scan output using IPv4 address with IPv4 firewall.

```
root@PC/# nmap 10.10.1.2
root@PC/# ...
root@PC/# Nmap scan report for 10.10.1.2
root@PC/# Host is up (0.00029s latency).
root@PC/# All 1000 scanned ports on 10.10.1.2 are filtered (1000)
root@PC/# MAC Address: BC:AC:C5:C4:CA:8F (Unknown)

Nmap done: 1 IP adress (1 host up) scanned in 21.30 secounds
```

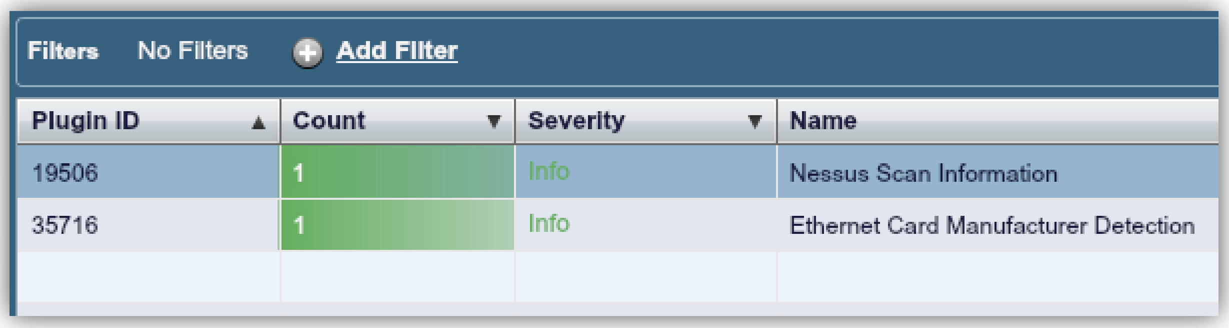
However, when scanning the IPv6 address using Nmap, there were two open ports, 53 domain and the 22 SSH. Port 53 is used mainly for DNS, but is also known to be used by trojans and worms. Furthermore, the SSH management service seems to be widely exposed over IPv6. See the Nmap output shown in Coding 3.7 below.

Coding 3.7: Nmap scan output using IPv6 address with IPv4 firewall.

```
root@PC/# nmap –6 2001:470:27:c1c::2
...
Nmap scan report for 2001:470:27:c1c::2
Host is up (0.00088s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
53/tcp open  domain

Nmap done: 1 IP address (1 host up) scanned in 1.14 seconds
```

Scanning using Nessus found similar results as Nmap. Using IPv4 the only information found was the network card manufacturer which actually was because of the MAC address. This means that the IPv4 firewall still is doing a good job when it handles IPv4 traffic, see Figure 3.3 for the IPv4 Nessus reprt.

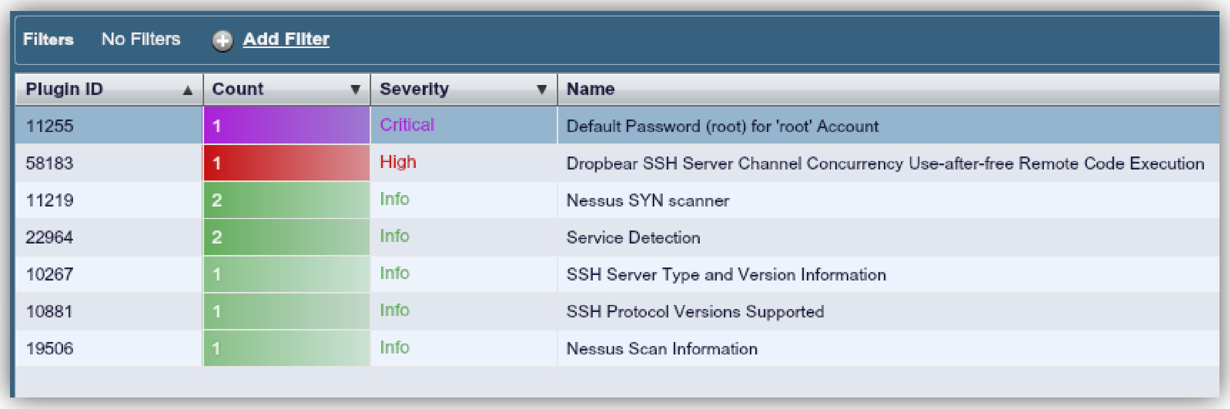


Filters No Filters Add Filter

Plugin ID	Count	Severity	Name
19506	1	Info	Nessus Scan Information
35716	1	Info	Ethernet Card Manufacturer Detection

Figure 3.3: Nessus scan screenshot, using IPv4 address with the IPv4 firewall.

However, using Nessus over the IPv6 address was much more dangerous since we now see the vulnerability of the unprotected SSH server, that has a well known vulnerability for remote code execution. Nessus even did a brute force attack and found the simple SSH password “root”. See Figure 3.4 for the IPv6 Nessus report.



Filters No Filters Add Filter

Plugin ID	Count	Severity	Name
11255	1	Critical	Default Password (root) for 'root' Account
58183	1	High	Dropbear SSH Server Channel Concurrency Use-after-free Remote Code Execution
11219	2	Info	Nessus SYN scanner
22964	2	Info	Service Detection
10267	1	Info	SSH Server Type and Version Information
10881	1	Info	SSH Protocol Versions Supported
19506	1	Info	Nessus Scan Information

Figure 3.4: Nessus scan screenshot, using the IPv6 address with the IPv4 firewall.

Since the default iptables focus on IPv4 and cannot even recognize IPv6 traffic, the IPv4 packet filtering firewall is clearly not sufficient to protect your device from IPv6 traffic. Even services that first seem to be secure over IPv4 are actually wide open for attacks over IPv6. The lesson learned from this experiment was, do not rely on your IPv4 firewall to handle everything, instead you will need an IPv6 firewall when IPv6 traffic is enabled.

3.6 Installing ip6tables

Installing ip6tables is similar to the previous installation of tcpdump at section 3.4.1. First you manually download the ipk ip6table[9] and the kmod[10] packages. It is again important that you download the correct packages for your kernel. You can check the kernel version by using the "uname –a" command. The ip6table file is basically the ip6tables program itself. The kmod file is simply a set of precompiled kernel modules that you need in order to use the ip6tables program.

At this point you either enable remote SSH over IPv4, this is done via the web GUI under the "Administration" tab, or simply reconnect the PC to the LAN port again because we will block IPv6 traffic.

Transfer the files using a SCP program to the router's "/jffs" folder. Using a forced installation makes the dependency check easier. The kmod packages contain the modules for stateful packet filtering, such as "nf_conntrack_ipv6.ko" which is a connection tracking module for state determination, "ip6table_filter.ko" that enables the filter table which will hold our filtering rules, and "ip6_tables.ko" which enables the use of the ip6table_filter, and finally "ip6t_rt.ko" which is needed to block packets containing RH0. All modules are unpacked to the "/jffs/lib/modules/2.6.25.20" folder and needs to be inserted using the "insmod" command. See Coding 3.8 for the commands used during the installation of ip6tables. Furthermore, if your downloaded modules are not compatible, insmod will not throw errors instead the use of ip6tables will inform that certain modules are missing. By using the commands "lsmod" you can see which modules are already enabled. Checking modules verifies that ip6tables is correctly installed and therefore is a part of the installation before we create the startup scripts for configuration and persistence. See the module error messages in Coding 3.9.

Coding 3.8: Asus router, ip6tables installation.

```
root@DD-WRT:/jffs# ipkg -force-depends install ip6tables_1.4.0-1_mipsel.ipk
...
Unpacking ip6tables...Done.
Configuring ip6tables...Done.
root@DD-WRT:/jffs# ipkg -force-depends install kmod-ip6tables_2.6.25.20-
brcm47xx-1_mipsel.ipk
...
Unpacking kmod-ip6tables...Done.
Configuring kmod-ip6tables...Done.
root@DD-WRT:/jffs# cd lib/modules/2.6.25.20
root@DD-WRT:/jffs/lib/modules/2.6.25.20# insmod ip6_tables.ko
root@DD-WRT:/jffs/lib/modules/2.6.25.20# insmod ip6table_filter.ko
root@DD-WRT:/jffs/lib/modules/2.6.25.20# insmod nf_conntrack_ipv6.ko
root@DD-WRT:/jffs/lib/modules/2.6.25.20# insmod ip6t_rt.ko
```

[9] File: ip6tables package, http://downloads.openwrt.org/kamikaze/8.09.2/brcm47xx/packages/ip6tables_1.4.0-1_mipsel.ipk

[10] File: kmod package, http://downloads.openwrt.org/kamikaze/8.09.2/brcm47xx/packages/kmod-ip6tables_2.6.25.20-brcm47xx-1_mipsel.ipk

Coding 3.9: Asus router, ip6tables missing module error message.

```
root@DD-WRT:/# ip6tables –L
ip6tables v1.4.0: can’t initialize ip6tables table ‘filter’: Table dose not
exist (do you need to insmod?)
Perhaps ip6tables or your kernel needs to be upgraded.
```

For some firmware versions, ip6tables finds its shared object files together with the original iptables, this is because the userland for ip6tables is organized in a different way than the iptables structure. In order to make ip6tables find the binary objects we need to edit the “IP6TABLES_LIB_DIR” environment variable along with the path variable “PATH”, using the commands shown in Coding 3.10.

Coding 3.10: Asus router, editing the ip6tables environment variables.

```
root@DD-WRT:/# export IP6TABLES_LIB_DIR=/jffs/usr/lib/iptables
root@DD-WRT:/# PATH=”$PATH”:/jffs/usr/sbin
```

In order to make these changes permanent, you need to add both the insmod and the export command to a startup script in your /jffs/etc/config directory, either as a new startup file or by adding the lines to your previous startup script. At this point, you should have ip6talbes installed and ready to configure the tables. Now we have the default chains, INPUT, OUTPUT, and FORWARD along with the filter table enabled and we connection tracking with the “conntrack” module. Remember that there are no rules in the chains yet along with an ACCEPT policy (i.e., there is no protection at this point).

3.7 Configuring an IPv6 firewall

Adding rules to the tables is the most important step. For information regarding specific commands beyond those addressed in this section we refer to the ip6tables manual [33]. Notice, good security practice and the rule of thumb is to deny everything and accept only the minimum we need, which means setting chain policies to DROP on arriving WAN traffic. See the example in Coding 3.11.

By creating the second startup script, for configuring the IPv6 firewall we need to start by enabling connectivity. ICMPv6 is critical for communication therefore we need some specific rules to allow ICMP to work properly. ICMPv6 types 1,2,3, and 4 are critical. Furthermore, in our test environment we need neighbor discovery enabled at the WAN temporarily and only during our test session for the directly connected attacking PC to have connectivity to the WAN port, i.e., message types 133-136 needs to be accepted as well. The easiest way to allow ICMP on the different default chains is to create a new chain with all the ICMPv6 rules and adding it to the default chains. This saves space and will speed up the rule matching process in those cases where a packet that is not a ICMPv6 packet just need to be matched one single time against the jump rule and not against all ICMPv6 rules if these rules was implemented in the default chain without a new chain jump rule. The firewall startup script used is shown in Coding 3.11.

Coding 3.11: Asus, IPv6wall.startup script part 1, securing default chains and allowing ICMPv6.

```
#!/bin/sh

# set environment variables
export IP6TABLES_LIB_DIR=/jffs/usr/lib/iptables
PATH="$PATH":/jffs/usr/sbin

# enable the kernel modules
insmod /jffs/lib/modules/2.6.25.20/ip6_tables.ko
insmod /jffs/lib/modules/2.6.25.20/ip6table_filter.ko
insmod /jffs/lib/modules/2.6.25.20/nf_conntrack_ipv6.ko
insmod /jffs/lib/modules/2.6.25.20/ip6t_rt.ko

# delete any old default chain rules and remove ICMPfilter
ip6tables -F INPUT
ip6tables -F FORWARD
ip6tables -F OUTPUT
ip6tables -F ICMPfilter
ip6tables -X ICMPfilter

# set chain policy
ip6tables -P INPUT DROP
ip6tables -P FORWARD DROP
ip6tables -P OUTPUT ACCEPT

# create ICMPv6 chain
ip6tables -N ICMPfilter

# allow specific ICMPv6 types in ICMPfilter chain
# allow basic connectivity
ip6tables -A ICMPfilter -p icmpv6 --icmpv6-type 1 -j ACCEPT
ip6tables -A ICMPfilter -p icmpv6 --icmpv6-type 2 -j ACCEPT
ip6tables -A ICMPfilter -p icmpv6 --icmpv6-type 3 -j ACCEPT
ip6tables -A ICMPfilter -p icmpv6 --icmpv6-type 4 -j ACCEPT

# allow neighbor discovery for lab purposes
ip6tables -A ICMPfilter -p icmpv6 --icmpv6-type 133 -j ACCEPT
ip6tables -A ICMPfilter -p icmpv6 --icmpv6-type 134 -j ACCEPT
ip6tables -A ICMPfilter -p icmpv6 --icmpv6-type 135 -j ACCEPT
ip6tables -A ICMPfilter -p icmpv6 --icmpv6-type 136 -j ACCEPT

# allow echo if router is a tunnel node
# using rate limiting for suppressing DoS attacks
ip6tables -A ICMPfilter -p icmpv6 --icmpv6-type 128 -m limit --limit 5/sec
--limit-burst 10 -j ACCEPT
ip6tables -A ICMPfilter -p icmpv6 --icmpv6-type 129 -j ACCEPT

# adding the ICMPfilter chain to default chains for WAN interface
ip6tables -A INPUT -i vlan2 -p icmpv6 -j ICMPfilter
ip6tables -A FORWARD -i vlan2 -p icmpv6 -j ICMPfilter
```

At this point the startup script provides ICMPv6 protection on the WAN port, but it is far from complete since we need to add more protection rules and allow more traffic to have a fully working LAN network. Next, packets containing routing header type 0 need to be dropped and this is done at each chain using the ip6t_rt module. See Coding 3.12 for the continuing part of the startup script.

Coding 3.12: Asus, IPv6wall.startup script part 2, preventing RH0.

```
# preventing Routing Header type 0
ip6tables -I INPUT 1 -m rt --rt-type 0 -j DROP
ip6tables -I FORWARD 1 -m rt --rt-type 0 -j DROP
ip6tables -I OUTPUT 1 -m rt --rt-type 0 -j DROP
```

To add more use of the LAN network we need to allow traffic for the LAN side and open common services and protocols that may want to communicate between LAN and WAN along with using connection tracking to make the firewall stateful. See Coding 3.13 for next part of the startup script.

Coding 3.13: Asus, IPv6wall.startup script part 3, allow basic LAN traffic.

```
# allow all traffic to loopback
ip6tables –A INPUT –i lo –j ACCEPT

# allow all traffic from LAN to local host
ip6tables –A INPUT –i br0 –j ACCEPT
# allow specific traffic from WAN to local host
ip6tables –A INPUT –I vlan2 –m state --state ESTABLISHED,RELATED –j ACCEPT

# allow all traffic from LAN to LAN/WAN
ip6tables –A FORWARD –i br0 –j ACCEPT
# allow traffic from WAN related or est. to LAN connections
ip6tables –A FORWARD –i vlan2 –m state --state ESTABLISHED,RELATED –j ACCEPT
```

```
root@DD-WRT:/jffs/etc/config# chmod 700 IPv6wall.startup
root@DD-WRT:/jffs/etc/config# ./IPv6wall.startup
```

At this point, we now accept all traffic originated from the LAN or local host to the WAN. Additionally, we accept only related or established traffic from the WAN to the LAN and local host. Furthermore from the previous parts; we accept specific ICMPv6 messages that are needed for the minimum connectivity at the WAN port; along with offering RH0 protection.

3.8 Testing Router Security

With the IPv6 packet filtering firewall in place we scan using the same procedure as in section 3.5.2. We now see that the nmap scanner did not even find a target. The Nmap scan output is shown in Coding 3.14 and Coding 3.15.

Coding 3.14: Nmap scan output try nr.1 using IPv6 address with IPv6 firewall.

```
root@PC/# nmap –6 2001:470:27:c1c::2
...
Note: Host seems down. If it is really up, but blocking our ping probes, try –Pn
Nmap done: 1 IP address (0 hosts up) scanned in 3.05 seconds
```

By using the recommended option “-Pn" (Treat all host as online), we found the host was up and discovered that all the common 1000 ports (i.e. ports specified in nmap-services config file) where filtered.

Coding 3.15: Nmap scan output try nr.2 using IPv6 address with IPv6 firewall.

```
root@PC/# nmap –6 -Pn 2001:470:27:c1c::2
...
Nmap scan report for 2001:470:27:c1c::2
Host is up.
All 1000 scanned ports on 2001:470:27:c1c::2 are filtered

Nmap done: 1 IP address (1 host up) scanned in 201.35 seconds
```

Next step is to test the Nessus vulnerability scanner again, exactly as in section 3.5.2. We previous found critical security flaws, such as the SSH admin password. This time using the IPv6 firewall Nessus does not even find the host online, quite similar to the Nmap scan. As a result no security vulnerabilities ware found by Nessus. As a consequence Nessus cannot brute force the SSH password and the router is a bit safer. See Figure 3.5 for the Nessus output.

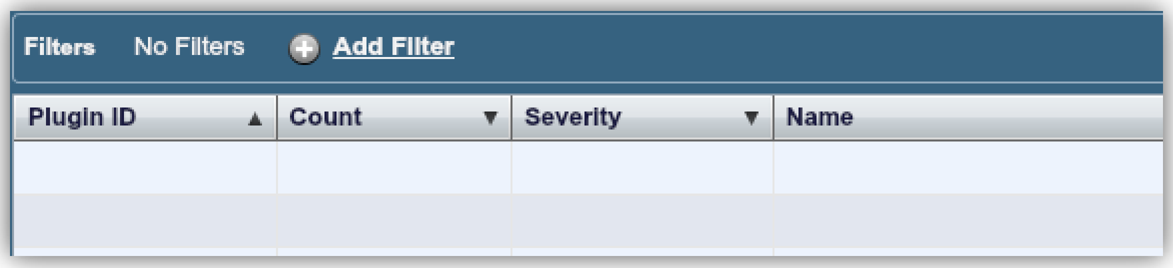
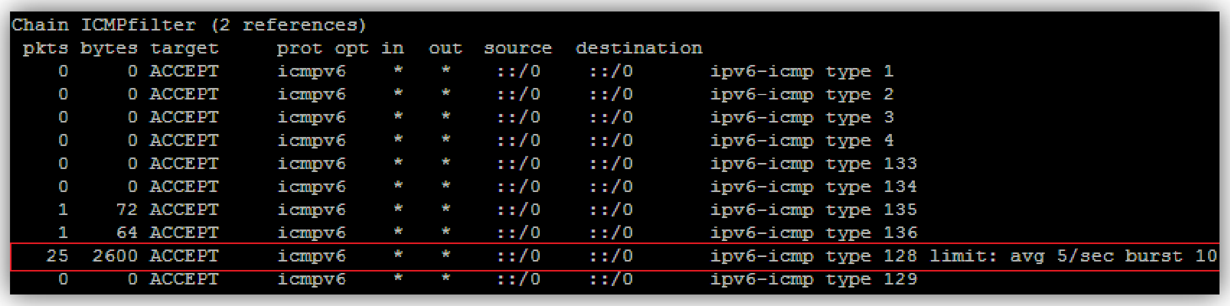


Figure 3.5: Nessus scan screenshot, empty result when using the IPv6 address with the IPv6 firewall configured.

Scanning seems not to reveal any security concerns any longer. Furthermore, we could test the rate-limiting functions manually by using Tcpdump and Wireshark while ping flooding the router to trigger the rate-limiting functions on the icmpv6 traffic. The pinging is done on the WAN as in the previous scans by using the option “-i” for interval set to 0 (i.e. less than 1 second) The rate-limiting rule states that only 5 packets are allowed each second and with a maximum burst of 10 packets, Figure 3.6 showns the output of ip6tables. From the ping statistics we can see that packet loss occurs for 88% of the transmitted packets. See Coding 3.16 for the exact ping command along with ping output.



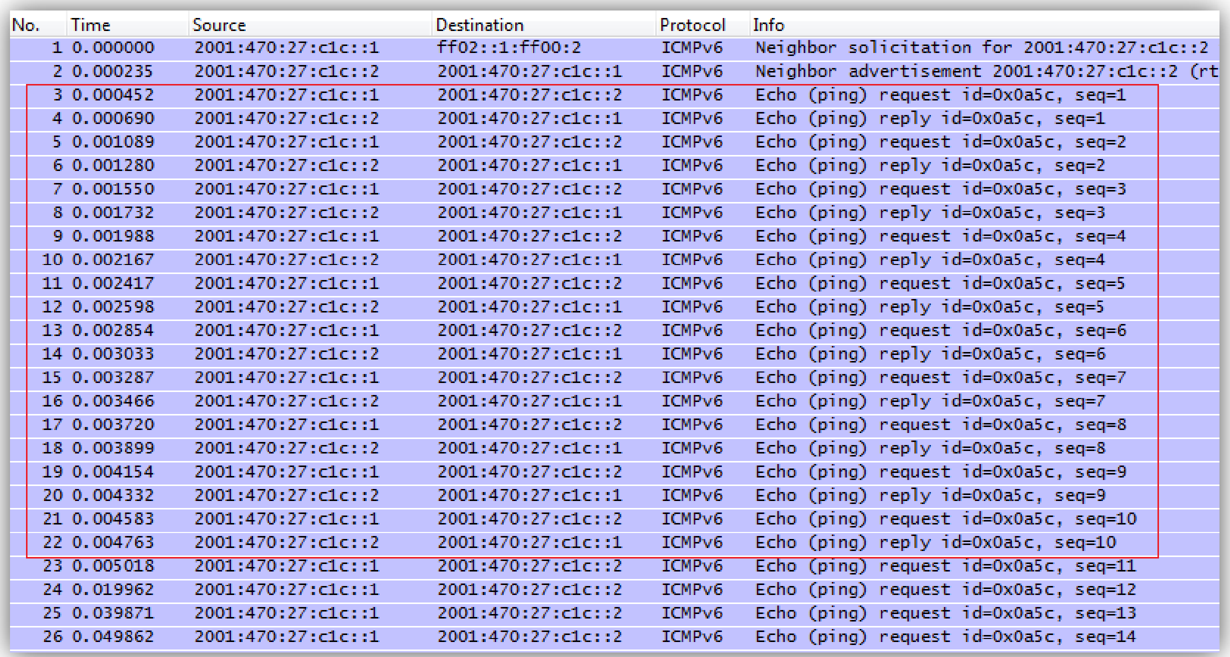
```
Chain ICMPfilter (2 references)
 pkts bytes target     prot opt in   out  source   destination
    0     0 ACCEPT     icmpv6    *    *    ::/0     ::/0       ipv6-icmp type 1
    0     0 ACCEPT     icmpv6    *    *    ::/0     ::/0       ipv6-icmp type 2
    0     0 ACCEPT     icmpv6    *    *    ::/0     ::/0       ipv6-icmp type 3
    0     0 ACCEPT     icmpv6    *    *    ::/0     ::/0       ipv6-icmp type 4
    0     0 ACCEPT     icmpv6    *    *    ::/0     ::/0       ipv6-icmp type 133
    0     0 ACCEPT     icmpv6    *    *    ::/0     ::/0       ipv6-icmp type 134
    1    72 ACCEPT     icmpv6    *    *    ::/0     ::/0       ipv6-icmp type 135
    1    64 ACCEPT     icmpv6    *    *    ::/0     ::/0       ipv6-icmp type 136
   25  2600 ACCEPT     icmpv6    *    *    ::/0     ::/0       ipv6-icmp type 128 limit: avg 5/sec burst 10
    0     0 ACCEPT     icmpv6    *    *    ::/0     ::/0       ipv6-icmp type 129
```

Figure 3.6: ip6tables -nvL output screenshot, focusing on icmpv6 rate-limiting echo requests.

Coding 3.16: Ping output, flood experiment.

```
root@PC/# ping6 -i 0 2001:470:27:c1c::2
PING 2001:470:27:c1c::2(2001:470:27:c1c::2) 56 data bytes
64 byte from 2001:470:27:c1c::2: icmp_seq=1 ttl=64 time=9.80 ms
64 byte from 2001:470:27:c1c::2: icmp_seq=2 ttl=64 time=0.336 ms
...
64 byte from 2001:470:27:c1c::2: icmp_seq=214 ttl=64 time=0.342 ms
--- 2001:470:27:c1c::2 ping statistics ---
221 packets transmitted, 25 received, 88% packet loss, time 3119ms
...
```

If we looking at the routers Tcpdump file, we see in Wireshark that the 10 first packets, sequence number 1-10, get allowed but the rest outside the 10 packet burst are dropped, i.e. the rate-limiting function works as intended to limit flooding and DoS attacks against the ICMPv6 echo message that is needed for tunnel connectivity. See Figure 3.7 for a Wireshark screenshot.



```
No.  Time      Source             Destination        Protocol  Info
  1 0.000000   2001:470:27:c1c::1  ff02::1:ff00:2      ICMPv6    Neighbor solicitation for 2001:470:27:c1c::2
  2 0.000235   2001:470:27:c1c::2  2001:470:27:c1c::1  ICMPv6    Neighbor advertisement 2001:470:27:c1c::2 (rt
  3 0.000452   2001:470:27:c1c::1  2001:470:27:c1c::2  ICMPv6    Echo (ping) request id=0x0a5c, seq=1
  4 0.000690   2001:470:27:c1c::2  2001:470:27:c1c::1  ICMPv6    Echo (ping) reply id=0x0a5c, seq=1
  5 0.001089   2001:470:27:c1c::1  2001:470:27:c1c::2  ICMPv6    Echo (ping) request id=0x0a5c, seq=2
  6 0.001280   2001:470:27:c1c::2  2001:470:27:c1c::1  ICMPv6    Echo (ping) reply id=0x0a5c, seq=2
  7 0.001550   2001:470:27:c1c::1  2001:470:27:c1c::2  ICMPv6    Echo (ping) request id=0x0a5c, seq=3
  8 0.001732   2001:470:27:c1c::2  2001:470:27:c1c::1  ICMPv6    Echo (ping) reply id=0x0a5c, seq=3
  9 0.001988   2001:470:27:c1c::1  2001:470:27:c1c::2  ICMPv6    Echo (ping) request id=0x0a5c, seq=4
 10 0.002167   2001:470:27:c1c::2  2001:470:27:c1c::1  ICMPv6    Echo (ping) reply id=0x0a5c, seq=4
 11 0.002417   2001:470:27:c1c::1  2001:470:27:c1c::2  ICMPv6    Echo (ping) request id=0x0a5c, seq=5
 12 0.002598   2001:470:27:c1c::2  2001:470:27:c1c::1  ICMPv6    Echo (ping) reply id=0x0a5c, seq=5
 13 0.002854   2001:470:27:c1c::1  2001:470:27:c1c::2  ICMPv6    Echo (ping) request id=0x0a5c, seq=6
 14 0.003033   2001:470:27:c1c::2  2001:470:27:c1c::1  ICMPv6    Echo (ping) reply id=0x0a5c, seq=6
 15 0.003287   2001:470:27:c1c::1  2001:470:27:c1c::2  ICMPv6    Echo (ping) request id=0x0a5c, seq=7
 16 0.003466   2001:470:27:c1c::2  2001:470:27:c1c::1  ICMPv6    Echo (ping) reply id=0x0a5c, seq=7
 17 0.003720   2001:470:27:c1c::1  2001:470:27:c1c::2  ICMPv6    Echo (ping) request id=0x0a5c, seq=8
 18 0.003899   2001:470:27:c1c::2  2001:470:27:c1c::1  ICMPv6    Echo (ping) reply id=0x0a5c, seq=8
 19 0.004154   2001:470:27:c1c::1  2001:470:27:c1c::2  ICMPv6    Echo (ping) request id=0x0a5c, seq=9
 20 0.004332   2001:470:27:c1c::2  2001:470:27:c1c::1  ICMPv6    Echo (ping) reply id=0x0a5c, seq=9
 21 0.004583   2001:470:27:c1c::1  2001:470:27:c1c::2  ICMPv6    Echo (ping) request id=0x0a5c, seq=10
 22 0.004763   2001:470:27:c1c::2  2001:470:27:c1c::1  ICMPv6    Echo (ping) reply id=0x0a5c, seq=10
 23 0.005018   2001:470:27:c1c::1  2001:470:27:c1c::2  ICMPv6    Echo (ping) request id=0x0a5c, seq=11
 24 0.019962   2001:470:27:c1c::1  2001:470:27:c1c::2  ICMPv6    Echo (ping) request id=0x0a5c, seq=12
 25 0.039871   2001:470:27:c1c::1  2001:470:27:c1c::2  ICMPv6    Echo (ping) request id=0x0a5c, seq=13
 26 0.049862   2001:470:27:c1c::1  2001:470:27:c1c::2  ICMPv6    Echo (ping) request id=0x0a5c, seq=14
```

Figure 3.7: Wireshark output screenshot from router Tcpdump file, highlighted rate-limiting function of the 10 packet burst.

3.9 Connecting to the Internet

This section is important before connecting to the Internet, you need to make the router deployment ready by cleaning up the rules. Depending on your IPv6 connection you need to configure the firewall accordingly. If you using native IPv6 the basic set of rules should work as a standard traffic filter out of the box after cleaning. However, if you using an IPv6-tunnel (which is the most common connectivity solution for most home and SME networks today) you will need to add the tunnel interface to the rules.

3.9.1 Clean up from the test session

Now it is time to **remove** the unnecessary icmpv6 rules in the ip6tables that was used for the test session only, which is the neighbor discovery messages shown in Coding 3.17. For the complete set of the rules that is recommended to use, see A.3 in the Appendix at page 46.

Coding 3.17: Neighbor discovery rules on WAN port.

```
# allow neighbor discovery for lab purposes
ip6tables -A ICMPfilter -p icmpv6 --icmpv6-type 133 -j ACCEPT
ip6tables -A ICMPfilter -p icmpv6 --icmpv6-type 134 -j ACCEPT
ip6tables -A ICMPfilter -p icmpv6 --icmpv6-type 135 -j ACCEPT
ip6tables -A ICMPfilter -p icmpv6 --icmpv6-type 136 -j ACCEPT
```

3.9.2 Tunnel or native IPv6 connection

If you have access to native IPv6 connectivity directly to the router, then you will not need any additional configuration from here on, and the ip6tables discussed in the clean up section 3.9.1 are ready to be deployed.

However, if you use a tunnel broker both the INPUT and the FORWARD chain must be modified to support the IPv6 tunnel. Coding 3.18 shows the tunnel rules. Note that the tunnel interface is not the WAN port interface "vlan2", but rather the newly created tunnel interface ("IPv6tun") between the tunnel broker and your device. Another even better solution would be to replace the "vlan2" rules to instead "IPv6tun" rules. In that case, the firewall is more precise to allow tunnel traffic only, instead of allowing both pure WAN port traffic that is not tunneled along with tunnel traffic.

Coding 3.18: Asus, IPv6wall, allow tunnel traffic.

```
# allow specific inbound traffic from interface "IPv6tun"
ip6tables -A INPUT -i IPv6tun -m state --state ESTABLISHED,RELATED -j ACCEPT
ip6tables -A FORWARD -i IPv6tun -m state --state ESTABLISHED,RELATED -j ACCEPT
ip6tables -A FORWARD -i IPv6tun -p icmpv6 -m state --state ESTABLISHED,RELATED -j
ACCEPT
```

With the firewall configured to filter the tunnel traffic, we are ready to connect the device to the live Internet. The configuration used for my tunnel connection is given in A.1 in the Appendix at page 45.

3.10 Firewall alternatives

In this section we are looking at alternative ways to implement the iptable firewall other than from the previous perimeter router approach. The entire firewall or parts of it could be moved or distributed among the LAN hosts, either for performance or to avoid having to configure the router. The complete set of rules recommended on the router is found in Appendix A.3 at page 46. Rules regarding the network, such as to suppress DoS attacks on the router to keep the network up are examples of a router based approach to implementing security, unlike the host protection parts such as RH0 dropping that easily could be implemented at the hosts instead. When dividing the firewall into perimeter-router and LAN host categories, we should always focus on what needs to be secure on the different platforms and keep the threat overview given in section 2.2 in mind.

3.10.1 The router approach

The router itself holds the key to the network, therefore it performs rate limiting and drops packets used for scanning to perform information gathering that could facilitate attacks at the router provides a good base for protecting the network. Having centralized security at one perimeter point makes administration easier, since the router can implement either global LAN rules that apply to all hosts, or use access control lists to limit access to specific hosts. Furthermore, as shown in the security testing of the IPv4 firewall in section 3.5.2, without an IPv6 firewall the router could be wide open for break-ins, which means that if you decide to **not** implement an IPv6 firewall, then you **must** revise the running services on the router. For instance to ensure that there is no SSH server running that could be scanned and brute forced over IPv6, simply by using the method described earlier this thesis.

3.10.2 The host approach

In the case of a dd-wrt non-compatible router or by putting the firewall completely in each host we must consider implementing iptables on workstations such as PC's. However, implementing iptables in other network devices, such as television sets and network attached storage (NAS) could be harder to implement, but necessary to implement if the router that provides connectivity to these devices gives attackers direct access to the network interface of these devices. However, if the firewall is implemented directly on each of the hosts there is no longer a single point of failure since each host can defend itself.

The portion of the iptables that easily can be implemented on any LAN host is the default INPUT chain, since from a host's perspective packets are never forwarded – thus packets are either destined for the local host through the INPUT chain or the packets get dropped. An example of such an iptable used on a LAN host is given in Appendix A.4 at page 47.

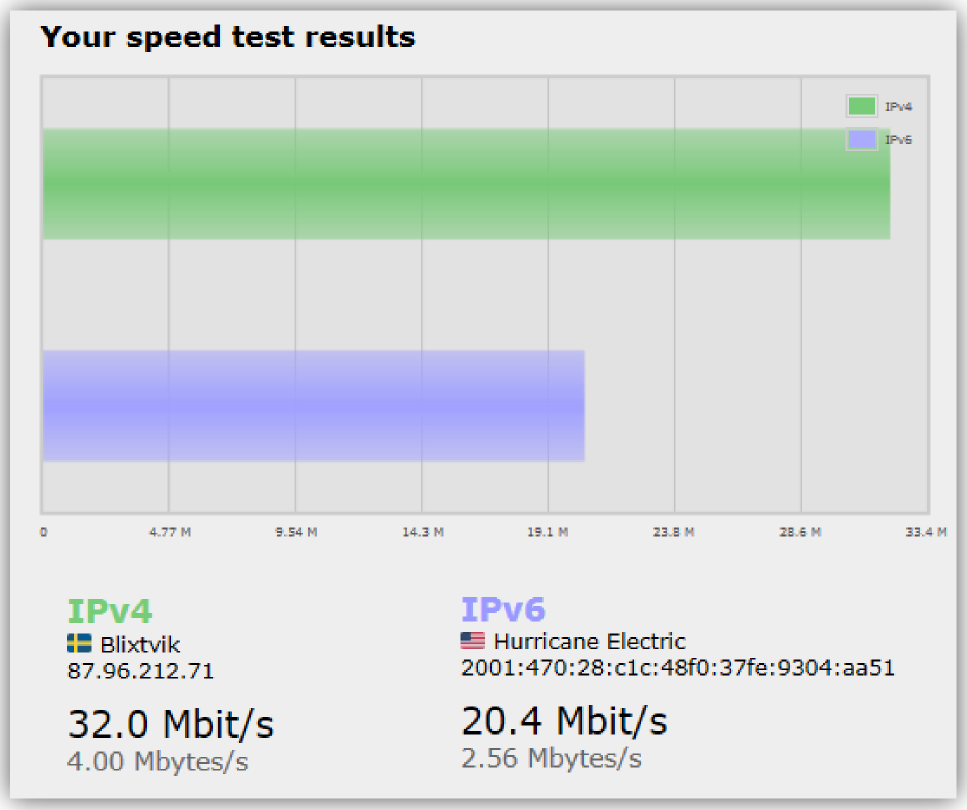
3.11 Performance analysis

In this section we take a look at how ip6tables work, how they are deployed, and we will compare the size of files and the required hardware resources. Additionally, we will look at speed optimizations for the router and firewall implementation.

3.11.1 LAN host evaluation of the ipv6tables

Checking connectivity and evaluating the router from a LAN host perspective should be done to confirm the configuration. Using your preferred way of getting online connectivity you next connect to a LAN host. You can evaluate the connection by visiting different websites providing pinging-based speed measurements. In this thesis we use a tunnel broker for connectivity along with "ipv6-test.com" for connectivity testing to provide an evaluation of the ipv6tables [44]. Figure 3.8 shows the speed test without the ipv6tables, and Figure 3.9 shows the same test with the ipv6 firewall.

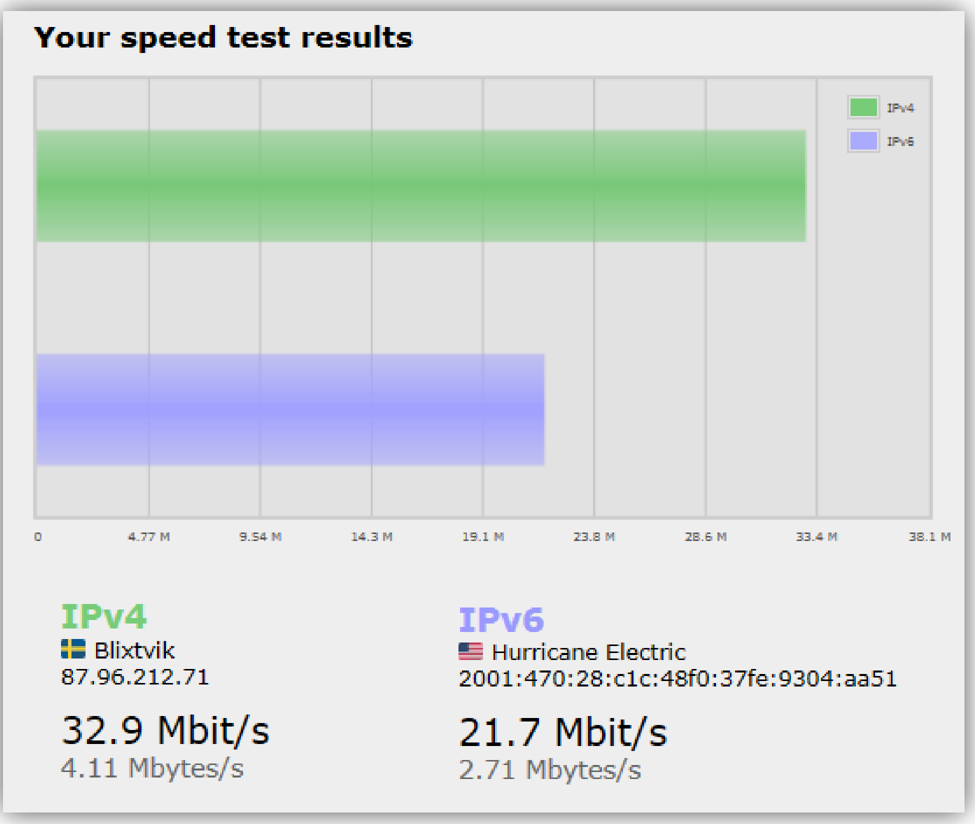
We clearly see that there is not a big difference in performance on a SME type network when deploying ipv6tables, which means this security method has a relatively low performance penalty. From a service perspective common protocols that use communication established from within the LAN will not require special configuration of the firewall. But more complicated protocols such as the FTP protocol might need additional configuration. This thesis does not cover configurations of the ipv6tables for specific protocols, hence this is left for future work (see section 4.1 at page 39).



IPv4
Blixtvik
87.96.212.71
32.0 Mbit/s
4.00 Mbytes/s

IPv6
Hurricane Electric
2001:470:28:c1c:48f0:37fe:9304:aa51
20.4 Mbit/s
2.56 Mbytes/s

Figure 3.8: Speed results from ipv6-test.com on LAN host without using router ipv6tables.



IPv4
Blixtvik
87.96.212.71
32.9 Mbit/s
4.11 Mbytes/s

IPv6
Hurricane Electric
2001:470:28:c1c:48f0:37fe:9304:aa51
21.7 Mbit/s
2.71 Mbytes/s

Figure 3.9: Speed results from ipv6-test.com on LAN host using router ipv6tables.

3.11.2 Storage space and file size comparison

The lack of storage space is sometimes a concern when using routers due to the often small flash memory (as discussed earlier in section 0 at page 14). The Asus router used in this project has a fairly large memory with 32 MB flash and 128 MB RAM. Choosing this router was a deliberate choice because this large amount of memory facilitated the activities in this thesis project. The files used during this project are shown in Listing 3.1 at page 21. Although router hardware might differ allot in terms of available storage capacity, the needed files will have a similar size. The dd-wrt firmware with IPv6 support always tend to be a bigger firmware build (about 6 MB compared to the smaller builds that lack IPv6 support that are only about 3 MB in size). The firmware used in this project has a total size of 6.82 MB which easily fits in the Asus 32 MB flash memory. Notice, that this is the only file that needs to be stored in the flash memory, all other files can be stored on an extern file system as was discussed in sections 0 and 2.5.3.

The secondly critical file is the ip6tables. In this project this file has a size of 47.9 KB, which is far smaller than the firmware. The ip6tables are stored in the RAM memory as described in section 2.5.3 at page 15. Since the ip6tables are fairly small in comparison to the Asus, 128 MB RAM memory, we do not need to store it outside the router.

Scripts is the final critical files that needs to be stored either onboard in the RAM or externally in order for the configurations to be loaded both for connectivity and firewall setup. The complete ip6tables firewall startup script shown in appendix A.3 has a size of 2.1 KB, which is very small compared to the onboard RAM storage of 128 MB.

If we looking at another common SME router, such as the D-link DIR-600 that is also supported by dd-wrt, we see that the builds available are only small builds that do not support IPv6. The DIR-600 router has a flash memory of 4 MB, therefore it lacks the space needed for including IPv6, hence IPv6 support is not included in any of the builds.

3.11.3 Speed optimization

Simple speed configurations are supported by most dd-wrt firmware versions, such as the “Overclocking Frequency” menu under the “Administration” and “Management” tab. Thus menu basically allows a user to increase the clock rate of the CPU and memory. Increasing these clock speeds places added pressure on the hardware making it perform faster but the device could consume more power and generate more heat. Secondly, it is possible to increase “Maximum Ports” under the same tab, which simply allows the router to have more available ports open at the same time, hence it can handle more sessions simultaneously. This is useful if there is a large number of network devices attached to the router. Furthermore, it is possible to use the quality of service (QoS) function under the “Application & Gaming” and “QoS” tab, to increase the priority of the most frequently used protocols.

From a firewall perspective, speed optimization should be accompanied by traffic analysis. The firewall rule list should be ordered to match the most frequent types of traffic. Since the firewall rule matching operation is sequential when a packet “traverses” the rule list looking for a matching rule, this packet matching time could be greatly decreased by simply putting the rules that apply to very common traffic at the very beginning of the rule list. Furthermore, using jump rules can optimize performance, as mentioned in section 3.7 at page 29, simply by defining specific rules that only need to be traversed if the jump rule is matched. Distributing parts of the firewall is another way of load balancing the firewall matching as discussed in section 3.10.

This page intentionally left blank

Chapter 4

4 Conclusion

Providing suitable IPv6 security is a global issue. The need for this security is growing with the expanding IPv6 deployment. Unfortunately, it is easy to attack IPv6 these days due to the lack of security knowledge by most users regarding this protocol, but fortunately it is harder to discover targets via global scanning. There are some new threats to the IPv6 protocol itself, such as new DoS possibilities and the increasing difficulty of ingress filtering (as ICMPv6 plays a bigger part in IPv6 than ICMP plays in IPv4).

This project has shown that common routers originally focused on IPv4 are not safe to use for IPv6 traffic without reconfiguration. The usual packet filtering IPv4 firewall does not protect against attacks carried out over IPv6, such as brute force attacks against application layer services. Secondly, this project has shown that an ordinary home router bought to support IPv4, **can** be used to support IPv6 in a more secure manner if the reconfiguration is done correctly.

We have seen that on the perimeter, there is a great need for finer granularly filters for ICMPv6 messages (along with rate-limiting), and that we must filter ports to secure services over IPv6 in order to provide security for IPv6 that is comparable with the existing IPv4 security. Furthermore, basic performance analysis has shown that the IPv6 security implementation used in this project does not affect the overall throughput on the IPv6 protected router.

Analysis of the hardware used with the firewall has shown that it is an important to choose hardware with sufficient storage space to support firmware that is IPv6 compatible and has a little extra room for the firewall itself.

Alternative security solutions such as implementing a host version of the ip6tables directly on the host is another approach that also gives IPv6 protection if the perimeter router does not have any running services that needs to be protected, such as an SSH service.

4.1 Future Work

A suggestion for further research would be to investigate how to optimize the firewall for protocol specific needs such as FTP helper modules [45]. Besides optimal firewall configurations for specific needs, an investigation of implementing user-awareness in the firewall so that firewall rules could be user specific and less IP address or host specific. It may be feasible to implement a light weight user authentication that the router could take advantage of when a larger number of users need to be authenticated. Investigate implementation of other security mechanisms in the increasingly equipped routers, such as DPI or intrusion detection systems (IDS). Furthermore, it would be interesting to investigate a comparison between physical and virtual firewalls in both performance and size minimization for devices, such as a larger home router.

4.2 Required Reflections

This section describes the author's reflections about how this bachelor thesis could affect the user and the society in different ways such as ethical, economical, and environmental aspects.

4.2.1 Social and ethical aspects

The author has preformed all work during this project in a responsible way since, due to the nature of the project, the work and experiments conducted during this bachelor thesis includes elements of security attacks. Therefore, all experiments were preformed in an isolated and controlled environment owned by the author. In this way, there was no chance of damage any other user or social function. All the results and knowledge of this thesis is of benefit for the good of the society, because the author proves and clarifies existing security problems and proposed solutions to these problems.

4.2.2 Economics and environmental aspects

All work regarding this thesis project was performed without any negative effect of the environment in general. The result of this thesis contributes to the reuse of network equipment and thus to the environment and the economics. The author does not recommend buying new expensive hardware, but proves instead the concept of a cost-effective solution by reusing existing hardware, that is both an economic and environmental benefit.

References

[1] Karine Perset, ‘Internet Address Space: Economic Considerations in the Management of IPv4 and in the Deployment of IPv6’, presented at the OECD Ministeral Meeting on The Future of the Internet Economy, Seoul, Korea, 2008, Available at http://www.oecd.org/dataoecd/7/1/40605942.pdf, [Accessed 20-Mars-2012].

[2] ‘IP Address Pools’. American Registry for Internet Numbers, Available at https://www.arin.net/knowledge/ip_address_pools.pdf, [Accessed 20-Mars-2012].

[3] Jan Hedström, ‘6 brandväggar för ipv6’, *TechWorld*, p. 6, 13-February-2012, Available at http://techworld.idg.se/2.15821/1.431895/6-brandvaggar-for-ipv6, [Accessed 20-Mars-2012].

[4] ‘Overview: Inside the Zeus Trojan’s source code’, *Ghana-India Kofi Annan Institute of Excellence in ICT*. [Online]. Available: http://ipv6.aiti-kace.com.gh/?q=content/overview-inside-zeus-trojan%E2%80%99s-source-code, [Accessed: 21-Mars-2012].

[5] Carolyn Duffy Marsan, ‘Five of the biggest IPv6-based threats facing CIOs’, *Network World*, p. 2, 13-July-2009, Available at http://www.networkworld.com/news/2009/071309-ipv6-network-threat.html, [Accessed: 21-Mars-2012].

[6] J. Abley, P. Savola, and G. Neville-Neil, ‘Deprecation of Type 0 Routing Headers in IPv6’, *Internet Request for Comments*, vol. RFC 5095 (Proposed Standard), December 2007, Available at http://www.rfc-editor.org/rfc/rfc5095.txt, [Accessed: 21-Mars-2012].

[7] ‘Malware Tunneling in IPv6’. US-CERT, 26-May-2005, Available at http://www.us-cert.gov/reading_room/IPv6Malware-Tunneling.pdf, [Accessed: 22-Mars-2012].

[8] Michael H. Warfield, ‘Security Implicitations of IPv6’, presented at the BlackHat Federal 2003, Virginia, United States, 2003, p. 15, Available at http://www.first.org/conference/2004/papers/c06.pdf, [Accessed: 22-Mars-2012].

[9] ‘IPv6 Security Brief’. Cisco System Inc., October-2011, Available at http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6553/white_paper_c11-678658.pdf, [Accessed: 22-Mars-2012].

[10] E. Karamanos, ‘Investigation of home router security’, Masters Thesis, KTH Royal Institute of Technology, Stockholm, Sweden, Available at http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-91107, 2010.

[11] ‘Cross-Site Request Forgery (CSRF) Prevention Cheat Sheet - OWASP’, 21-November-2011. [Online]. Available: https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet#No_Cross-Site_Scripting_.28XSS.29_Vulnerabilities, [Accessed: 23-Mars-2012].

[12] Neal Ziring, ‘Router Security Guidance Activity of the Systems and Network Attack Center (SNAC)’. National Security Agency, 23-May-2006, Available at http://www.nsa.gov/ia/_files/routers/I33-002R-06.pdf, [Accessed: 23-Mars-2012].

[13] 'Chapter 7. Service and Application Version Detection'. [Online]. Available: http://nmap.org/book/vscan.html, [Accessed: 23-Mars-2012].

[14] A. Conta, S. Deering, and M. Gupta, 'Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification', *Internet Request for Comments*, vol. RFC 4443 (Draft Standard), March 2006, Available at http://www.rfc-editor.org/rfc/rfc4443.txt, [Accessed: 23-Mars-2012].

[15] Fredrik Folke and Gilbert Lidholm, 'The exploitation of reverse TCP, Explaning the inside out vulnerability', 24-November-2011. [Online]. Available: http://dl.dropbox.com/u/15311781/The%20exploitation%20of%20reverse%20TCP.pdf, [Accessed: 23-Mars-2012].

[16] Ganesh Dutt Sharma, 'Packet Filtering Firewall: An Introduction', *Security World*, 27-June-2010, Available at http://securityworld.worldiswelcome.com/packet-filtering-firewall-an-introduction, [Accessed: 24-Mars-2012].

[17] Philippe Biondi and Arnaud Ebalard, 'IPv6 Routing Header Security.', IT Sec lab, Suresnes, France, 2007, Available at http://www.secdev.org.nyud.net:8080/conf/IPv6_RH_security-csw07.pdf, [Accessed: 26-Mars-2012].

[18] A. Conta and S. Deering, 'Generic Packet Tunneling in IPv6 Specification', *Internet Request for Comments*, vol. RFC 2473 (Proposed Standard), December 1998, Available at http://www.rfc-editor.org/rfc/rfc2473.txt, [Accessed: 26-Mars-2012].

[19] E. Davies and J. Mohacsi, 'Recommendations for Filtering ICMPv6 Messages in Firewalls', *Internet Request for Comments*, vol. RFC 4890 (Informational), May 2007, Available at http://www.rfc-editor.org/rfc/rfc4890.txt, [Accessed: 26-Mars-2012].

[20] Hogg Scott and Eric Vyncke, *IPv6 Security*, vol. no. 5133. Indianapolis, USA: Cisco Press, 2009, ISBN: 978-1-58705-594-2.

[21] 'Samba filesystem', *DD-WRT Wiki*, 15-February-2011. [Online]. Available: http://www.dd-wrt.com/wiki/index.php/Samba, [Accessed: 01-April-2012].

[22] 'IANA — Internet Assigned Numbers Authority'. [Online]. Available: http://www.iana.org/, [Accessed: 02-April-2012].

[23] Stephan Lagerholm, 'The IPv4 Depletion site'. [Online]. Available: http://www.ipv4depletion.com/?page_id=326, [Accessed: 08-April-2012].

[24] S. Deering and R. Hinden, 'Internet Protocol, Version 6 (IPv6) Specification', *Internet Request for Comments*, vol. RFC 2460 (Draft Standard), December 1998, Available at http://www.rfc-editor.org/rfc/rfc2460.txt, [Accessed: 08-April-2012].

[25] F. Gont, 'ICMP Attacks against TCP', *Internet Request for Comments*, vol. RFC 5927 (Informational), July 2010, Available at http://www.rfc-editor.org/rfc/rfc5927.txt, [Accessed: 08-April-2012].

[26] Anders Magnusson, 'Småföretagsroutrar får dd-wrt', *idg*, 22-March-2010. [Online]. Available: http://www.idg.se/2.1085/1.304328/smaforetagsroutrar-far-dd-wrt, [Accessed: 08-April-2012].

[27] Oskar Andreasson, ‘Iptables Tutorial 1.2.2’, *frozentux*, 2006. [Online]. Available: http://www.frozentux.net/iptables-tutorial/iptables-tutorial.html, [Accessed: 09-April-2012].

[28] ‘Hard reset or 30/30/30’, *DD-WRT Wiki*, 15-August-2010. [Online]. Available: http://www.dd-wrt.com/wiki/index.php/Hard_reset_or_30/30/30, [Accessed: 11-April-2012].

[29] ‘Installation’, *DD-WRT Wiki*, 09-March-2012. [Online]. Available: http://www.dd-wrt.com/wiki/index.php/Installation, [Accessed: 11-April-2012].

[30] ‘Main Page’, *DD-WRT Wiki*, 04-June-2007. [Online]. Available: http://www.dd-wrt.com/wiki/index.php/Main_Page, [Accessed: 11-April-2012].

[31] ‘Tenable Products Plugin Families’, *Tenable*, 13-April-2012. [Online]. Available: http://static.tenable.com/documentation/Tenable_Products_Plugin_Families.pdf, [Accessed: 15-April-2012].

[32] ‘Nessus Documentation’, *Tenable*, 2012. [Online]. Available: http://www.tenable.com/products/nessus/documentation, [Accessed: 15-April-2012].

[33] ‘ip6tables(8): IPv6 packet filter administration’, *Linux man page*. [Online]. Available: http://linux.die.net/man/8/ip6tables, [Accessed: 18-April-2012].

[34] ‘Hardware’, *DD-WRT Wiki*, 03-January-2011. [Online]. Available: http://www.dd-wrt.com/wiki/index.php/Hardware, [Accessed: 18-April-2012].

[35] ‘Script Execution’, *DD-WRT Wiki*, 22-March-2012. [Online]. Available: http://www.dd-wrt.com/wiki/index.php/Script_Execution, [Accessed: 18-April-2012].

[36] ‘Startup Scripts’, *DD-WRT Wiki*, 22-November-2010. [Online]. Available: http://www.dd-wrt.com/wiki/index.php/Startup_Scripts, [Accessed: 18-April-2012].

[37] ‘NoScript - InformAction’, *NoScript*. [Online]. Available: http://noscript.net/, [Accessed: 11-May-2012].

[38] ‘netfilter/iptables project’, *The netfilter.org project*. [Online]. Available: http://www.netfilter.org/, [Accessed: 12-May-2012].

[39] Darren Reed, ‘TCP/IP Firewall/NAT Software’, *IP Filter*. [Online]. Available: http://coombs.anu.edu.au/~avalon/, [Accessed: 12-May-2012].

[40] ‘iptables(8)’, *Linux man page*. [Online]. Available: http://linux.die.net/man/8/iptables, [Accessed: 12-May-2012].

[41] ‘Free Security Scanner For Network Exploration & Security Audits’, *Nmap*. [Online]. Available: http://nmap.org/, [Accessed: 12-May-2012].

[42] ‘Tcpdump/Libpcap public repository’, *Tcpdump & Libpcap*. [Online]. Available: http://www.tcpdump.org/, [Accessed: 12-May-2012].

[43] ‘Wireshark · Go deep.’, *Wireshark*. [Online]. Available: http://www.wireshark.org/, [Accessed: 12-May-2012].

[44] ’IPv6 vs. IPv4 broadband’, *IPv6 test* [Online]. Available: http://ipv6-test.com/speedtest/, [Accessed: 23-May-2012]

[45] Eric Leblond, Pablo Neira Ayuso, Patrick McHardy, Jan Engelhardt, and Dash Four, ‘Secure use of iptables and connection tracking helpers’, *To Linux and beyond*, 2012. [Online]. Available: https://home.regit.org/netfilter-en/secure-use-of-helpers/, [Accessed: 23-May-2012].

Appendix

This appendix lists the basic background configurations used during installations.

A.1: Startup script for configuring IPv6 tunnel between IPv6 broker and router.

```
# Load kernel module
insmod ipv6

# Enable IPv6 forwarding
echo 1 > /proc/sys/net/ipv6/conf/all/forwarding

# Create the tunnel interface with server/client
ip tunnel add IPv6tun mode sit remote 216.66.80.90 local “router wan IPv4” ttl
255

# Activate the new tunnel interface
ip link set IPv6tun up

# Add the router’s IPv6 address to the tunnel device
ip -6 addr add 2001:470:27:c1c::2/64 dev IPv6tun

# Add the IPv6 address of the router’s bridge interface for LAN ports
ip -6 addr add 2001:470:28:c1c::1/64 dev br0

# Add the IPv6 internet traffic pointing to the tunnel
ip -6 route add 2000::/3 dev IPv6tun

# Enforce the protocol family identifier
ip –f inet6 addr
```

A.2: radvd.conf, configurations for LAN prefix announcement.

```
interface br0
{
            AdvSendAdvert on;
            prefix 2001:470:28:c1c::/64
            {
                        AdvOnLink on;
                        AdvAutonomus on;
                        AdvRouterAddr on;
            };
};
```

A.3: Complete ip6tables rules set recommended for perimeter router.

```
#!/bin/sh

# set environment variables
export IP6TABLES_LIB_DIR=/jffs/usr/lib/iptables
PATH="$PATH":/jffs/usr/sbin

# enable the kernel modules
insmod /jffs/lib/modules/2.6.25.20/ip6_tables.ko
insmod /jffs/lib/modules/2.6.25.20/ip6table_filter.ko
insmod /jffs/lib/modules/2.6.25.20/nf_conntrack_ipv6.ko
insmod /jffs/lib/modules/2.6.25.20/ip6t_rt.ko

# delete any old default chain rules and remove ICMPfilter
ip6tables -F INPUT
ip6tables -F FORWARD
ip6tables -F OUTPUT
ip6tables -F ICMPfilter
ip6tables -X ICMPfilter

# set chain policy
ip6tables -P INPUT DROP
ip6tables -P FORWARD DROP
ip6tables -P OUTPUT ACCEPT

# allow all traffic from LAN to LAN/WAN
ip6tables -A FORWARD -i br0 -j ACCEPT
# allow traffic from WAN related or est. to LAN connections
ip6tables -A FORWARD -i vlan2 -m state --state ESTABLISHED,RELATED -j
ACCEPT

# allow all traffic to loopback
ip6tables -A INPUT -i lo -j ACCEPT

# allow all traffic from LAN to local host
ip6tables -A INPUT -i br0 -j ACCEPT
# allow specific traffic from WAN to local host
ip6tables -A INPUT -i vlan2 -m state --state ESTABLISHED,RELATED -j ACCEPT

# create ICMPv6 chain
ip6tables -N ICMPfilter

# allow specific ICMPv6 types in ICMPfilter chain
# allow basic connectivity
ip6tables -A ICMPfilter -p icmpv6 --icmpv6-type 1 -j ACCEPT
ip6tables -A ICMPfilter -p icmpv6 --icmpv6-type 2 -j ACCEPT
ip6tables -A ICMPfilter -p icmpv6 --icmpv6-type 3 -j ACCEPT
ip6tables -A ICMPfilter -p icmpv6 --icmpv6-type 4 -j ACCEPT

# allow echo if router is a tunnel node
# using rate limiting for suppressing DoS attacks
ip6tables -A ICMPfilter -p icmpv6 --icmpv6-type 128 -m limit --limit 5/sec
--limit-burst 10 -j ACCEPT
ip6tables -A ICMPfilter -p icmpv6 --icmpv6-type 129 -j ACCEPT

# adding the ICMPfilter chain to default chains for WAN interface
ip6tables -A INPUT -i vlan2 -p icmpv6 -j ICMPfilter
ip6tables -A FORWARD -i vlan2 -p icmpv6 -j ICMPfilter

# preventing Routing Header type 0
ip6tables -I INPUT 1 -m rt --rt-type 0 -j DROP
ip6tables -I FORWARD 1 -m rt --rt-type 0 -j DROP
ip6tables -I OUTPUT 1 -m rt --rt-type 0 -j DROP
```

A.4: Example of ip6tables rules set recommended for LAN Hosts.

```
# Interface assumptions: eth0 is used for router connectivity
# IP LAN subnet: 192.168.1.0/24
# delete any old default chain rules and remove ICMPfilter
ip6tables –F INPUT
ip6tables –F OUTPUT
ip6tables –F ICMPfilter
ip6tables –X ICMPfilter

# set chain policy
ip6tables –P INPUT DROP
ip6tables –P OUTPUT ACCEPT

# allow all traffic to loopback
ip6tables –A INPUT –i lo –j ACCEPT

# allow all traffic from source LAN subnet to local host
ip6tables –A INPUT –s 192.160.1.0/24 –j ACCEPT
# allow specific traffic from WAN to local host
ip6tables –A INPUT –i eth0 –m state --state ESTABLISHED,RELATED –j ACCEPT

# create ICMPv6 chain
ip6tables –N ICMPfilter

# allow specific ICMPv6 types in ICMPfilter chain
# allow basic connectivity
ip6tables –A ICMPfilter –p icmpv6 --icmpv6-type 1 –j ACCEPT
ip6tables –A ICMPfilter –p icmpv6 --icmpv6-type 2 –j ACCEPT
ip6tables –A ICMPfilter –p icmpv6 --icmpv6-type 3 –j ACCEPT
ip6tables –A ICMPfilter –p icmpv6 --icmpv6-type 4 –j ACCEPT

# allow neighbor discovery
ip6tables –A ICMPfilter –p icmpv6 --icmpv6-type 133 –j ACCEPT
ip6tables –A ICMPfilter –p icmpv6 --icmpv6-type 134 –j ACCEPT
ip6tables –A ICMPfilter –p icmpv6 --icmpv6-type 135 –j ACCEPT
ip6tables –A ICMPfilter –p icmpv6 --icmpv6-type 136 –j ACCEPT

# allow echo for debugging or if host is a tunnel node
# using rate limiting for suppressing DoS attacks on Host
ip6tables –A ICMPfilter –p icmpv6 --icmpv6-type 128 –m limit --limit 5/sec
--limit-burst 10 –j ACCEPT
ip6tables –A ICMPfilter –p icmpv6 --icmpv6-type 129 –j ACCEPT

# adding the ICMPfilter chain to default chains for WAN interface
ip6tables –A INPUT –i eth0 –p icmpv6 –j ICMPfilter

# preventing Routing Header type 0
ip6tables –I INPUT 1 –m rt --rt-type 0 –j DROP
ip6tables –I OUTPUT 1 –m rt --rt-type 0 –j DROP
```

TRITA-ICT-EX-2012:118