

KYMENLAAKSON AMMATTIKORKEAKOULU
Information Technology / Network Engineering

Rikhard Tjeder

IPv6 SECURITY

Thesis 2014

TIIVISTELMÄ

KYMENLAAKSON AMMATTIKORKEAKOULU

Tietotekniikka

TJEDER, RIKHARD

IPv6 Security

Opinnäytetyö

48 sivua + 32 liitesivua

Työn ohjaaja

Lehtori Martti Kettunen

Toimeksiantaja

Kyamk, Kymp OY

Maaliskuu 2014

Avainsanat

IPv6, Security, Neighbor Discovery, Extension Headers

Tämän opinnäytetyön aihe on IPv6 protokollan turvallisuus. Työ keskittyy suurimaksi osaksi ongelmiin, jotka esiintyvät Neighbor Discovery protokollassa sekä IPv6 Extension Headereissä. Lisäksi käsitellään IPv6 tiedustelu ja IPv6 Bogonien tietoturvaongelmia sekä näiden ongelmien vaikutusten lieventämistä.

Tämän opinnäytetyön tavoitteena on kerätä tietoa IPv6 protokollan turvallisuusongelmista yleistä käyttöä varten. Tavoitteena olisi myös vähentää kyseisten ongelmien vaikutusta ja myös testata, että tehdyt toimenpiteet toimivat. Tietoturvaongelma käsitellään ja arvioidaan tarkemmin, jos kyseessä on vaativa ongelma tai sen vaikutusten vähentämiseen on useita vaihtoehtoja.

Tutkimuksissa on käytetty luotettavista lähteistä saatua materiaalia, kuten alan kirjallisuutta ja luotettavien osapuolten internet-sivuja. Merkittävä osa materiaalista on Cisco Systemsin julkaisemaa. Opinnäytetyön toteutus koostuu puoliksi kirjallisuustutkimuksesta ja puoliksi käytännön kokeista.

Opinnäytetyössä päädyttiin johtopäätökseen, että vaikka IPv6 on turvallisempi kuin IPv4, löytyy siitä vielä haavoittuvuuksia, joita joudutaan torjumaan. IPv6-protokollan turvallisuuden syvällisempi tarkastelu vaatisi huomattavasti enemmän aikaa, koska aihealue on niin laaja.

ABSTRACT

KYMENLAAKSON AMMATTIKORKEAKOULU

University of Applied Sciences

Information Technology

TJEDER, RIKHARD

Ipv6 Security

Bachelor's Thesis

48 pages + 32 pages of appendices

Supervisor

Martti Kettunen, Principal Lecturer

Commissioned by

Kyamk, Kymp OY

March 2014

Keywords

IPv6, Security, Neighbor Discovery, Extension Headers

The subject of this thesis was security in the IPv6 protocol. The subject areas within it that were examined in this thesis mostly revolve around the problems within Neighbor Discovery and the IPv6 Extension Headers. In addition, other problems were taken into account such as reconnaissance of IPv6 networks and IPv6 Bogons. The mitigation for these problems was also discussed.

The aim of this thesis was to generally gather information about some of the security threats involved in IPv6 for general purpose use and to be informational. Moreover, the objectives were implementing some of the mitigations and actually testing that they work. The problems themselves were evaluated on how they should be dealt with if problematic subjects were encountered or multiple choices were present.

The research was conducted by going through material on trusted media such as books published by trusted parties, the same applies to any information collected from the internet. Mostly the information was gathered from Cisco sources. The empirical part consisted of half research and half practical work.

To conclude, while better than IPv4 security-wise IPv6 still poses many threats and requires them to be mitigated. It would be also wise to research the subject further considering the size of the subject itself, for example further research into IPv6 Bogons and new ways to secure Neighbor Discovery.

LIST OF ABBREVIATIONS

IP	Internet Protocol; A communication protocol.
IPv6	Internet Protocol version 6; The latest revision of the Internet Protocol.
IPv4	Internet Protocol version 4; The predecessor of IPv6
IETF	Internet Engineering Task Force; Organization in charge of the standardization of the Internet Protocol.
IPng	Internet Protocol Next Generation; A precursor or early version of IPv6
MAC	Media Access Control; A data communication protocol.
IPsec	Internet Protocol Security; A protocol suite for securing IP communication.
Node	Nodes is a catch-all term for devices in a network that are either intermediate devices such as routers or end-point devices such as computers.
Router	A device used for forwarding packets across a network.
Host	A device that is not a router, such as a computer.
QoS	Quality of Service; a way to control what traffic is considered important by increasing their priority in the network.
TTL	Time-To-Live; the lifespan of a packet traversing through the network, usually decremented by 1 for every hop.
GiB	Gigabyte; a unit used for digital information storage, 1 gigabyte = 1073741824 bytes.
PMTU	Path Maximum Transmission Unit; value which indicates the largest datagram size allowed on the path from source to destination without being fragmented.

RH0	Routing Header Type 0; part of the Routing extension header, works similarly to IPv4 source routing option.
RH2	Routing Header Type 2; part of the Routing extension header, used in MobileIPv6
DoS	Denial of Service; an attack designed to deny a network or computer resources they require to work.
CPU	Central Processing Unit; hardware in devices such as computer and routers used for processing information.
ND	Neighbor Discovery (Protocol); a protocol within IPv6 used in helping nodes to be aware of their neighboring nodes in attached links and to auto-configure hosts with addresses and prefixes.
RA	Router Advertisement; a message sent by a router to a host to inform about the link so they can auto-configure themselves.
RS	Router Solicitation; a message sent out by a host node when it requests a router to send it a RA message.
SEND	Secure Neighbor Discovery; a security protocol used to secure the Neighbor Discovery Protocol
CGA	Cryptographically Generated Addresses;
DDoS	Distributed Denial of Service, the same as DoS attacks but done by multiple people or in some cases botnets.
RIR	Regional Internet Registry; an organization that manages the allocation and registration of Internet number resources within a particular region.
IANA	Internet Assigned Numbers Authority; nonprofit private American corporation which oversees global IP address allocation.

DAD	Duplicate Address Detection; verification mechanism in ND that checks that the auto-configured address is no present on any other device.
NUD	Neighbor Unreachability Detection; a mechanism in ND used to inform neighbors when a neighboring node is unreachable.
Bogon	A bogus address from an unallocated address space.
ACL	Access Control List; a feature used in devices to filter network traffic.
OUI	Organizational Unique Identifier; unique identifier for gear by specific vendors for different devices.

TABLE OF CONTENTS

TIIVISTELMÄ

ABSTRACT

LIST OF ABBREVIATIONS

1	INTRODUCTION	9
2	INTERNET PROTOCOL VERSION 6	9
2.1	IPv6 FEATURES	10
2.2	IPv6 HEADER	11
2.3	ADDRESSING	13
2.3.1	GLOBAL UNICAST ADDRESSES	14
2.3.2	LINK-LOCAL	14
2.3.3	MULTICAST ADDRESSING	14
2.4	EXTENSION HEADERS	14
2.4.1	HOP-BY-HOP OPTION	15
2.4.1.1	ROUTER ALERT	15
2.4.1.2	JUMBOGRAMS	15
2.4.2	ROUTING	16
2.4.2.1	TYPE 0 ROUTING HEADER	16
2.4.2.2	TYPE 2 ROUTING HEADER	16
2.4.3	FRAGMENT	17
2.4.4	ENCAPSULATION SECURITY PAYLOAD	18
2.4.5	AUTHENTICATION HEADER	18
2.4.6	NO NEXT HEADER	19
2.4.7	DESTINATION OPTION	19
3	NEIGHBOR DISCOVERY PROTOCOL	19
3.1	ROUTER ADVERTISEMENTS AND SOLICITATIONS	20
3.2	NEIGHBOR SOLICITATION AND ADVERTISEMENTS	20
3.3	DUPLICATE ADDRESS DETECTION	20
3.4	NEIGHBOR UNREACHABILITY DETECTION	21
3.5	NEIGHBOR CACHE	21

4	SECURE NEIGHBOR DISCOVERY	22
5	IPV6 RA GUARD	23
6	BOGONS	23
7	RECONNAISSANCE	24
	7.1 TECHNIQUES	24
	7.1.1 REGISTRIES	25
	7.1.2 AUTOMATED SCANNING	25
	7.1.3 MULTICAST RECONNAISSANCE	26
	7.1.4 NEIGHBOR DISCOVERY RECONNAISSANCE	26
	7.2 RECONNAISSANCE TOOLS	27
	7.3 PROTECTION AGAINST RECONNAISSANCE	27
8	EMPIRICAL PART	28
	8.1 EXTENSION HEADERS	29
	8.1.1 ROUTING HEADER TYPE 0 FILTERING	30
	8.1.2 ROUTER ALERT FILTERING	34
	8.1.3 FRAGMENTATION FILTERING	36
	8.1.4 CONCLUSION TO EXTENSION HEADERS TESTING	38
	8.2 SEND IMPLEMENTATION	38
	8.2.1 SEND CONFIGURATION	39
	8.2.2 CONCLUSION TO SEND CONFIGURATION	44
9	CONCLUSION TO THESIS	45
	REFERENCES	47
	APPENDICES	

1 INTRODUCTION

As IPv6 starts to become more common with the IPv4 address pool becoming exhausted, security issues within the IPv6-protocol must be dealt with. As service providers and enterprises start to use more IPv6-addressing they must also start using more IPv6-security. The security aspect is still developing and ways of attacking networks and devices keep evolving. Despite this, there are security options and mitigations for issues within the IPv6-protocol.

This study was aimed towards researching and testing IPv6-security as IPv6 is starting to become more common by the day. Various different vulnerabilities were researched in both how they work and how they are mitigated if it is possible to even mitigate. Solutions to spot or find possible vulnerabilities within the network were researched.

The problems that were researched in this study involved reconnaissance, Bogons, Extension Headers, Neighbor Discovery and possible tools used by attackers in trying to take advantage of the weaknesses in each of the areas. Mitigations for before-mentioned problems were addressed both in simply discussing them and implementing them in practical work, mostly in the ND and Extension header areas.

This thesis was commissioned by Kymp Oy. The practical testing was mostly conducted within the Kymenlaakso University of Applied Sciences SimuNet-Lab and also some of the equipment located in the Cisco-Lab residing in the same building. The tests were conducted on various equipment located both in the SimuNet-Lab network and in the Cisco-Lab equipment.

SimuNet-lab within Kymenlaakso University of Applied Sciences in its simplest form is a small Internet service provider environment mostly used for research, testing and new product development. With the help of the SimuNet-network those difficult tasks which Internet service providers have with testing new network solutions can be simulated.

2 INTERNET PROTOCOL VERSION 6

Internet Protocol version 6 (IPv6) is an Internet Protocol (IP) developed to be the successor of Internet Protocol version 4 (IPv4) as the IPv4 address pool has become exhausted. The Internet Engineering Task Force, which is in charge of defining the In-

ternet Protocol standards, could not foresee the global expansion that the Internet would have and could not predict what kind of impact this could have on the security of the Internet. Security was in fact not considered that important in the original design of IPv4. (Hogg & Vyncke, 2008, 3.)

As the Internet grew from just being formed from different organization's networks into what it is today, security and the vulnerabilities in IPv4 started to become a major concern. In the beginning of the 1990s, the IETF started developing a new version of IP, because they not only noticed the security hazards within IPv4 but also the fact that they were going to run out of addresses form the IPv4 address pool. They started developing IP Next Generation (IPng) which later became IPv6. (Hogg & Vyncke, 2008, 3.)

2.1 IPv6 FEATURES

IPv6 had many features which outperformed and improved on concepts from IPv4 but also introduced new features. The main features introduced in IPv6 as listed by Blanchet:

- **Larger addresses.** IPv4 had 32 bit address space which was increased to 128 bit address space in IPv6. This enabled the possibility of addressing all nodes increasing features such as security.
- **More levels of addressing hierarchy.** With the added level of addressing hierarchy it provided better aggregation of routes, allocation of addresses to downstreams became easier as did the scalability of the global routing table.
- **Scoping in the addressing.** Made filtering easier at boundaries, also increased link layer protocol security against remote attacks.
- **Simple and fixed address architecture.** Making /48 prefix for sites and /64 for links made it easier to manage addressing plans.
- **Privacy addresses.** End-user IP address cannot be used for tracking traffic.
- **Multiple addresses on an interface.** Enabled multiple use, virtual hosting easier, renumbering and multihoming.

- **Autoconfiguration of nodes.** Nodes now auto-configured themselves based on advertisements and inserting their MAC address into the host part of the IPv6 address making configuration fast and reliable.
- **No address conflicts on links.** Due to MAC addresses being used in the auto-configuration process, duplicate addresses would no longer be a problem.
- **Simpler and more efficient IP header.** Packets were processed more efficiently making forwarding easier for the routers.
- **Extension headers.** Similar to the IPv4 Options Header the extension headers in IPv6 allowed for more options within the IPv6 packets and because most of them do not have to be processed, it made forwarding faster.
- **Mandatory IP security.** IPsec was mandatory in IPv6, securing traffic for all nodes if any underlying key infrastructure is present.
- **Source routing.** An extension header option which made it easier to implement source routing.
- **Labeling flows for QoS.** Flow labels in the basic header made it easier for the routers to determine labeling and policing without having to check the application payload.
- **Multicast for discovery and link-local interaction.** Broadcasting was not used in IPv6. Only the nodes that should receive a request would receive it.
- **Mobility.** Mobility options were integrated into IPv6 headers, stacks and implementations making implementing mobility for network access easier.
- **Private but unique address space.** Unlike the private addresses in IPv4 the private addresses in IPv6 were unique to the site making connecting two or more private networks together. (Blanchet, 2006, 30 – 32.)

2.2 IPv6 HEADER

IP headers are used to give the devices a great deal of information on the Internet Layer. There are 8 fixed fields in the IPv6 Header which amount to 40 bytes. Shown is a depiction of the IPv6 Header (figure 1).

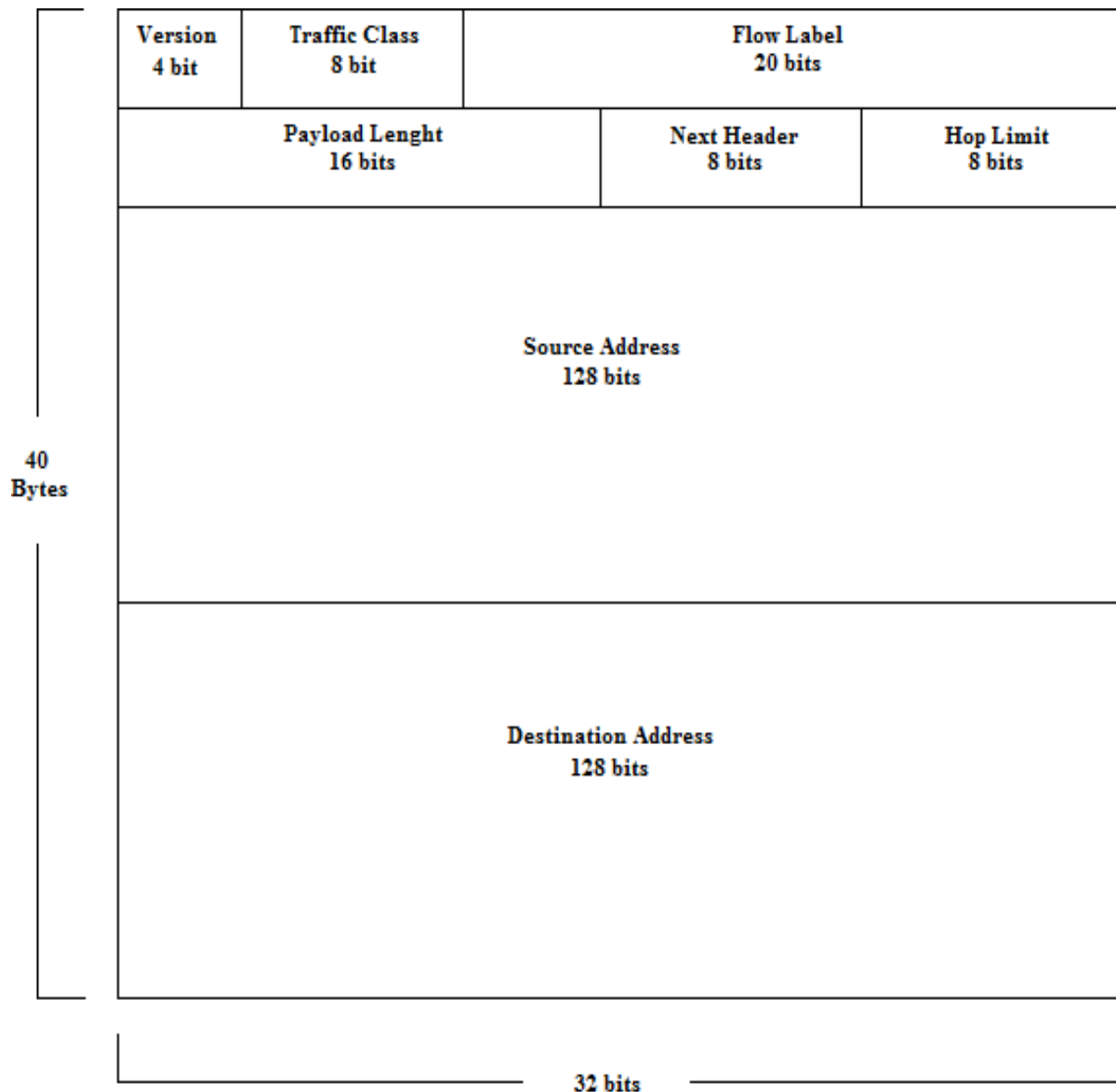


Figure 1. IPv6 Header

The fixed fields are as follows:

- **Version Field.** The Version field is 4 bits long and is used to identify which IP protocol version is in use. In IPv6 the value is “6” to indicate that IPv6 is in use.
- **Traffic Class Field.** The Traffic Class field is 8 bits long and is used to classify what sort of traffic the packets are, making it easier for devices if packets are high or low priority traffic.
- **Flow Label Field.** The Flow Label field is 20 bits. Unlike in IPv4 where routers had to specifically process, based on the IP header and transport header, what the flows for the sequence of incoming packets were. IPv6 implement the

Flow Label, which makes it easier for routers to process which flows the packets should follow.

- **Payload Length.** The Payload Length field is 16 bits long. It has the information regarding the length of what comes after the IP header, which means it tells the length of the transport, application data and possible extension headers.
- **Hop Limit.** The Hop Limit field is 8 bits long. Similar to the TTL (Time-To-Live) field in IPv4 it defines how long a time a datagram has to “live” until it is “killed”. This basically means when the packet traverses through the network its hop limit is decremented and when value hits 0 the device sends back a 'Time exceeded' ICMP message to inform the sender that the packet has been discarded due to the hop limit being exceeded.
- **Next Header.** The Next Header field is 8 bits long. This field is used for identification of the data inside the payload of the IP datagram. This is usually different sort of transport protocols- or encapsulation. The Next Header field is also in charge of telling the routers what Extension Headers are used up ahead. (Blanchet, 2006, 47 – 49.)

2.3 ADDRESSING

IPv6 addresses are 128 bits long which basically means there are 3.4×10^{38} unique addresses. This provides for a much larger address space than IPv4 also unlike IPv4 the address notations are not in decimals such as 192.0.0.1 but in hexadecimals such as 2001:0::1234:c1c0:abcd:243. The netmasks have also changed into prefix lengths, which are noted for example as /64 which means identifies an address 64 first bits. For example in the previous IPv6 address 2001:0::1234:c1c0:abcd:243 it would mean the 1234:c1c0:abcd:234 part.

2.3.1 GLOBAL UNICAST ADDRESSES

Global Unicast Addresses are addresses used to communicate between two nodes over the Internet. These addresses range from 2000:: to 3fff:ffff:ffff:ffff:ffff:ffff:ffff:ffff or in other terms 2000:/3. All Global Unicast Addresses use the prefix length of /64, this is because the leftmost 64 bits are for the network prefix and the rightmost 64 bits are used for the host part of the address. IPv6 also requires that addresses used are provider-based, which means one cannot acquire an IPv6 address block from a registry but has to get it from an upstream provider. (Blanchet, 2006, 64.)

2.3.2 LINK-LOCAL

Link-local addresses are scoped addresses, they are used on the specific link on the interface. This means they are used between the connection of two nodes on the same link. Link-local addresses use the structure fe80::<interface identifier>. They also happen to be automatically configured on every IPv6-enabled interface, this allows for communication between nodes without having to manually configure the interfaces. Routers never forward Link-local addresses. (Blanchet, 2006, 66.)

2.3.3 MULTICAST ADDRESSING

Multicast addressing is used to send a datagram to many nodes in a network. Unlike Broadcast in IPv4, where one node sends out a datagram to all nodes in the network and even the nodes it does not concern have to listen to it, IPv6 multicast send out the datagrams in specific multicast addresses specifying which nodes should listen said datagram. A multicast address is assigned to be temporary for a specific time or permanently. The lifetime and scope are united into the multicast address, this causes the multicast addresses to look as following ff<L><S>:<multicast group identifier> where <L> is the lifetime and <S> for the scope. (Blanchet, 2006, 71.)

2.4 EXTENSION HEADERS

Extension headers, similar to Options Field in IPv4 are not fixed to the IP header such as the IPv4 Options Field but since options and exceptions might be needed they are added as a non-mandatory feature. All the different extension headers are identified via unique Next Header values, so when an Extension Header is called upon the Next

Header in the IP header points to next Extension header. Now when the router processes the extension header it encounters another next header which points to the next extension header if one exists. (Blanchet, 2006, 50.)

2.4.1 HOP-BY-HOP OPTION

The hop-by-hop option extension header is always read first and has to be looked at by every device on the path to a packet's destination. There are two uses for the hop-by-hop option which are router alert and jumbogram. (Blanchet, 2006, 51.)

2.4.1.1 ROUTER ALERT

This function in the hop-by-hop option is to tell or alert all the devices in the route to the destination of the packet to process this certain datagram. An example would be RSVP (Resource Reservation Protocol) which sends only control messages to all routers of a path in one datagram. Because routers usually just view the transport data to check if a datagram is important, which causes a lot of load on processing by the router, router alerts will inform the router to take a look at this datagram so only the tagged datagrams are processed. (Blanchet, 2006, 51.)

Since the Router Alert option can be used to force routers to take a closer look at packets this can be used by attackers to their advantage. They will bombard the router with packets with the Hop-by-hop header with the Router Alert option perhaps with a combination with the Npad option to cause a DoS attack by forcing the router to use its CPUs resources to go through all the files. All depends on the network and the person in charge of administrating the network. They should block Router Alerts on routers since if they are using protocols that actually require Router Alerts simply blocking them would cause issues. However if this does not happen to be a problem and one can simply block the packets, this can be done using ACLs. (Hogg & Vyncke, 2008, 33.)

2.4.1.2 JUMBOGRAMS

Because the upper limit of the IP datagram is limited, to 65536 bytes due to the 16 bits in the Payload Length field an extension header was needed to allow for bigger datagrams that require special processing, the Jumbograms extension header. The Jumbogram allows for a maximum upper limit of 4 GiB. The Jumbogram extension header is mostly used for just for fast and large links between supercomputers. (Blanchet, 2006, 56 - 57.)

2.4.2 ROUTING

The Routing extension header allows for repathing of packets when they reach intermediate nodes to their destination, thus being able to shift their paths at will. There are currently two relevant types of Routing headers, RH0 (Routing Header Type 0) and RH2 (Routing Header Type 2).

2.4.2.1 TYPE 0 ROUTING HEADER

RH0 as in the Type 0 Routing Headers allow packets routing paths to be modified to the liking of the sender by the use of the Routing extension header. The Type 0 routing header is very simplistic and thus can be abused by the sender. For instance there is no limit to how many intermediate nodes the packet can be redirected to, even if it is back to the one it just came from. This allows for DoS (Denial of Service) attacks by causing an amplification attack. The amplification attack is performed by using RH0 to tell the intermediate routers to send the same packet back and forth, during this the attacker is obviously sending in more of the same packets to cause the routers to be unable to handle to load. RH0 can also be used to set a path that avoids specific intermediate nodes, in particular firewalls or nodes that could block RH0. (Hogg & Vyncke, 2008, 36.)

One option to mitigate this problem is disabling IPv6 source routing in routers to disallow packets from using source routing in the routers, however this will also disable Type 2 routing headers from being source routed by the router which if you need would be troublesome, on top of this disabling the IPv6 source routing does not stop the packet itself so if you happen to have another router in the network that allows the source routing on the packets path to its destination it will be able to use source routing at that node. That is why one must stop the packet using an ACL, which will identify any packets using RH0 and block it. (Hogg & Vyncke, 2008, 37.)

2.4.2.2 TYPE 2 ROUTING HEADER

RH2 as in the Type 2 Routing Header is used for source routing in MobileIPv6. Just like RH0, RH2 allows routers to redirect packets to other intermediate nodes before reaching their destination. However the difference is that they learned from Type 0 Routing Headers not making it as simplistic. Now the routing header is only allowed to have one address it can be redirected too before reaching its destination. The destination IPv6 address for the packet has to be from a network interface of the node that

owns the intermediate address. This allows the source routing to be only used with the recipient's addresses. Shown below is how source routing could be used (figure 2) (Koodli & Perkins, 2007, 130.)

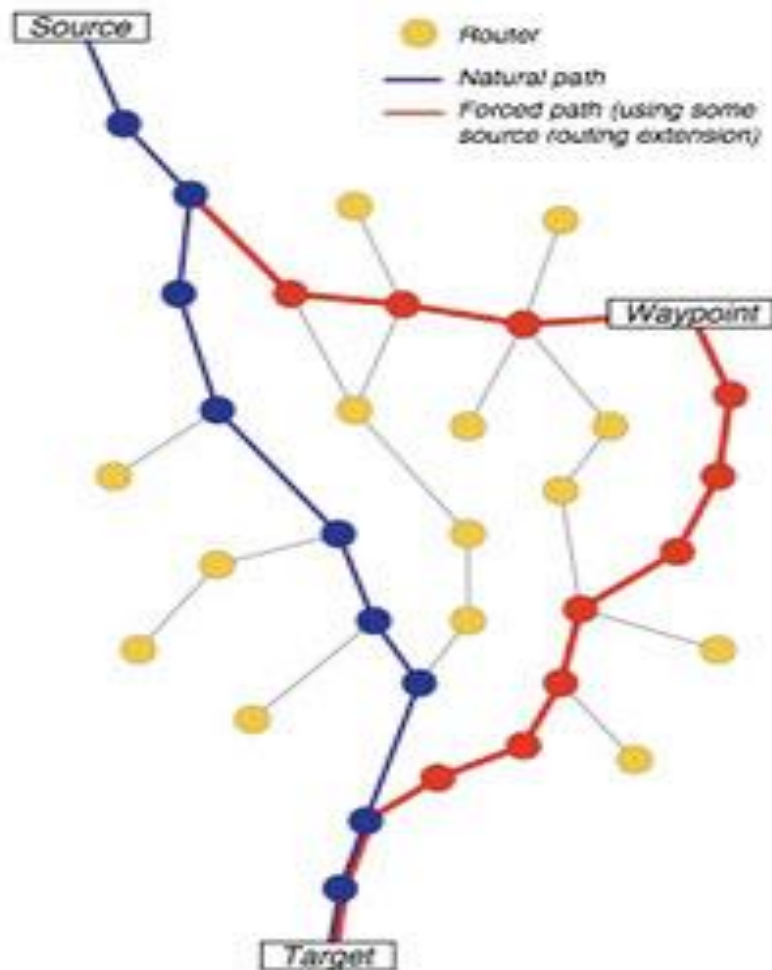


Figure 2. Packet being manipulated to take a specific path using source routing. The Internet Society (2010).

2.4.3 FRAGMENT

If a source node notices that the PMTU is smaller in size than the datagram being sent, then it must be fragmented. In IPv6 the source node is in charge of fragmentation and not the router, the destination node reassembles the fragments on delivery. This is done by using the Fragment extension header which splits the fragment into pieces, that are smaller or the same size as the PMTU and they all share the original IP header of the original datagram but also have an additional fragment header. To identify

which fragments are a part of a specific datagram they have a 32 bit identification field in the fragment header. Because the order the fragments will arrive at the destination is not linear they also specify which order the fragments are supposed to be re-assembled by using a 13 bit field called “fragment offset”, they also use a flag to indicate which fragment is the last fragment of the datagram. (Blanchet, 2006, 55-56.)

Attacks can use fragmentation to their advantage when trying to get past firewalls by fragmenting files into very small pieces in an attempt to pass through the firewall undetected. However, filtering these packets might be troublesome since filtering cannot determine which fragments might be harmful and which might not be. This is why the person in charge of the network has to decide if filtering should be implemented in some point in the network as no fragmented packets should travel that way. The filtering is done by using ACLs that block packets which have the extension header pointing to a packet being fragmented. (Hogg & Vyncke, 2008, 43.)

2.4.4 ENCAPSULATION SECURITY PAYLOAD

The Encapsulation Security Payload extension header is used by Ipv6 to encapsulate packets. It provides confidentiality, integrity for the encapsulated packet, authentication of the packets source and anti-replay protection. Unlike the authentication header it only provides protection for the data segment of the packet but the protection it gives is encryption of the data. (Blanchet, 2006, 239.)

2.4.5 AUTHENTICATION HEADER

The Authentication Header is used by Ipv6 for authentication. It provides integrity in the whole packet, helps authenticate the source and has replay protection. It provides this protection to the Version field, payload length field, next header field, the source address field, the destination address field, the extension header field and the Data section of the packet. It does not provide protection for the Traffic Class, Flow Label or Hop Limit headers. It uses a cryptographic checksum over the fields protected. (Blanchet, 2006, 235-236.)

2.4.6 NO NEXT HEADER

The no next header extension header informs that there is no payload, which means that all the relevant information in the datagram is in the header or extension header. This is used when control protocol messages are using extension headers and do not add extra information in the payload. (Blanchet, 2006, 52.)

2.4.7 DESTINATION OPTION

The Destination Options are used to carry optional information to the destination node. It uses different sort of binary values which correspond with different predetermined options which it calls upon. One of the options used is the Home Address Destination Option Extension Header. This option is used in mobility support for IPv6. (RFC2460.) (RFC6265.)

There are also two other Destination Options called Pad1 and PadN options. Pad1 inserts one octet of padding into the option portion of the header while the PadN option allows you to insert a variable amount. The PadN option can be used by attackers in a combination with Router Alert to make a packet that is forced to be viewed by the router that had a large Npad option junk data. Even on their own they could be quite effective. That is why it would be best for firewalls to drop packets with multiple padding options and padding option exceeding 5 bytes on top of dropping padding that has anything besides 0s in the datafield. (Hogg & Vyncke, 2008, 28.)

3 NEIGHBOR DISCOVERY PROTOCOL

ND (Neighbor Discovery) protocol is a feature in IPv6 which allows nodes to determine the link-layer address for other nodes connected them via links. Hosts use this to locate routers that would be willing to forward for them. It is also used by nodes to determine if their neighbors are currently reachable and if their link-layer addresses have changed. (RFC2460.)

3.1 ROUTER ADVERTISEMENTS AND SOLICITATIONS

RA(Router Advertisement) messages are ICMPv6 messages that are used to help neighboring hosts to get information about the link so they can auto-configure themselves. The important information carried in these messages is the prefix or possible prefixes and the default router or default routers. These messages are sent every 5 minutes. Because these messages are sent at 5 minute intervals this could cause problems if a node has to reboot or restart as it would not receive the information to auto-configure itself, so when the device is booting it sends a RS (Router Solicitation) message to request a RA message to be sent to it so it can auto-configure itself. If it just receives a regular RA message from two different routers it auto-configures both prefixes. However, if there are two routers both sending a host node RA messages claiming they are the default router, it chooses either or starts doing round robin between the two routers competing for being the default router. (Blanchet, 2006, 81-84.)

3.2 NEIGHBOR SOLICITATION AND ADVERTISEMENTS

A Neighbor Solicitation message is used when a node wants to send a packet to a neighbor on the same link, this is sent to the solicited-node multicast address which sends it out to all the neighbors on the link which then view the solicitation to see if the destination IPv6 address matches their address. If it does, they send back a Neighbor Advertisement to tell the soliciting node what the link layer address of the node is. (Blanchet, 2006, 116.)

3.3 DUPLICATE ADDRESS DETECTION

DAD (Duplicate Address Detection) is used in the ND protocol to make sure nodes are not using the same address causing problems in the network working correctly. This is checked with a node wanting to use a specific link-local address. It then sends out a Neighbor Solicitation message to the multicast address to ask around if that specific address would respond. If there is no response by any of the other nodes, it concludes that this address is not being used by anyone else and starts using it. However, if a node responds with a Neighbor Advertisement, the node concludes the address is already in use and does not use that address. (Blanchet, 2006, 117.)

While this is a useful function in ND it can be exploited by attackers. They could cause a DoS attack by configuring a node to always respond to solicitation with a Neighbor Advertisement. This would cause a situation where a node could not auto-configure itself because the attacker's node would keep responding to the advertisement with a Neighbor Advertisement claiming that it is currently using the address.

3.4 NEIGHBOR UNREACHABILITY DETECTION

If there is no traffic between nodes for a long period of time nodes would not know if a neighbor is reachable or not. This is why ND has a feature called Neighbor Unreachability Detection. NUD (Neighbor Unreachability Detection) is used by nodes to determine if their neighboring nodes are still reachable. This feature comes in handy to determine unreachable neighboring nodes, especially if there are multiple default routers being advertised helping the node to switch over to the other default router if the path to the other default router is unreachable. (Blanchet, 2006, 118.)

NUD activates automatically when there is no traffic between nodes for a while, this helps ND to know the states of neighboring nodes to see if they are up or not. It sends a NA message over to the node a few times to check for responses and if it receives none it determines that the node is unreachable and deletes it from the Neighbor cache. Because this is done by using NS messages where the node responds with a NA message, it can be leveraged by an attacker to cause a DoS attack via responding to the NS messages over and over with forged NA messages. (RFC3756)

3.5 NEIGHBOR CACHE

The neighbor cache is the list of neighboring nodes. The cache tells you what state the neighbors are in currently. The states in which nodes can be in are the following.

- **Incomplete.** This state indicates that a neighbor solicitation message has been sent to the neighbor but no neighbor advertisement has been sent back yet.
- **Reachable.** Indicates that a neighbor advertisement has been received from this neighbor recently.
- **Stale.** Indicates that a neighbor advertisement has been received but the expiration time has been reached. The neighboring node will stay in this state until a new message is sent to said node and the neighbor solicitation is restarted.

- **Delay.** Similar to the stale state but a state before a neighbor solicitation is sent again via the probe state.
- **Probe.** Neighbor solicitation has been sent to the neighboring node.

(Blanchet, 2006, 117.)

4 SECURE NEIGHBOR DISCOVERY

SEND (Secure Neighbor Discovery) is a security measure for Neighbor Discovery that uses authentication in the links formed between nodes. This means a Certificate Server must be set up which grants a Router its certificate letting it act as a Router. When nodes are connected to the network they must be configured with a trust anchor. This means the router must have a certification path to the certificate server so that the node can make the router its default router. SEND also uses CGA (Cryptographically Generated Addresses) to make sure that the node sending out a Neighbor Discovery message actually is the node that owns the address it claims it has. (RFC3971.)

CGA requires a RSA public keypair which it uses to generate an address for the node using a cryptographic hash function. This means the node can now claim ownership to the address CGA creates because it is associated with the RSA keypair that was generated on the node. This means no other node can claim they have this address without actually having the RSA keypair at their disposal. CGA also has a security value called the Sec which has 8 different values ranging from 0 to 7. These are used to increase the strength of the encryption to avoid attackers from simply cracking the encryption by brute-forcing the address. (RFC3972.)

This way SEND allows for Neighbor and Router discovery messages to be protected by either using certificates, using CGA or in the best case possibly using both. This has the benefits of protecting Neighbor Solicitation and Advertisement spoofing because SEND requires RSA Signatures and ,prepver. the CGA option in the solicitation which means if they do not correspond to what the router requires they will be disregarded. DAD Dos attacks are also mitigated by the fact that when a SEND node is generating its first address it listens for non-SEND nodes for possible duplicate addresses but on the 2nd attempt if a non-SEND node responds yet again it disregards the message and starts using the generated address anyway. (RFC3971.)

Even though SEND mitigates some of the possible DoS and intrusion attempts SEND itself also allows for attackers to launch DoS attacks against some of SENDs features. Because SEND uses CGA, certificates or in some cases both for validation it has to verify each packet. This can cause a situation where CGA will be first in line for verification which means the nodes could possibly cause unnecessary work for the devices processor by simply verifying the incoming packets. Especially routers can also be targeted by using multiple trust anchors to request multiple certification paths forcing the routers to use resources in said process. With certificates it would be wise to also configure nodes so that too many resources are not used for verification of files to prevent similar situations. (RFC3971.)

5 IPV6 RA GUARD

IPv6 RA Guard is a feature that allows one to filter out unwanted RA messages by rogue routers. RA Guard announces to the network what kind of device there is at the end of the port and allows a network administrator to set up a safe environment with trusted ports allowing in RA messages while untrusted ports are denied. RA Guard allows a device to be either configured to be a host or a router. It also allows for other optional configurations that increase the security of RA Guard such as adding custom hop-limits, matching the sender to specific IP address using access lists or checking that the prefix being advertised matches the correct prefix-list. RA Guard is slightly lighter than SEND, but it only protects against rogue RA messages. (Cisco, IPv6 RA Guard)

However RA-Guard may be spoofed by Extension Headers depending on how the message is read. An attacker could make an RA message that uses an Extension Header, making use of Hop by Hop or Destination Options Header, allowing Router Advertisement messages to slip by RA Guard due to how the packet is read and processed. Also using this in combination with fragmentation would allow the RA message to slip past a Layer 2 device. (v6ops, 2011.)

6 BOGONS

Bogons deriving from Bogus addresses are IP addresses or network spaces that should never appear on the Internet routing table and are normally used as a bogus source address in DDoS attacks. These addresses are address spaces that have not been allocat-

ed to RIR (Regional Internet Registry) by IANA(Internet Assigned Numbers Authority). These apply to both the Ipv4 address space and the IPv6 address space. (Hogg & Vyncke, 2008, 87.)

There are various ways to block Bogons in a network such as using Unicast RPF. Probably the most useful way to do this is using RTBH (Remotely Triggered Black Hole) filtering. RTBH routes unwanted traffic based on either the source address or the destination address into a null0 interface which means the packet is being routed into “nowhere”. The general problem with Bogon filtering is the fact that especially in IPv6 the address space that is unallocated will keep on changing as they are given out, which will require for filters to be updated so that no actual traffic from previously unallocated address spaces is blackholed or filtered. This will cause an issue where the network administrator has to weigh their options depending on if they want to set up an automatically updateable filter or a manually managed filter. There are some companies willing to offer automatic updates through IBGP but this might not be an option for some administrators because the companies claim not to be eligible for taking any responsibility in any possible problems resulting from the service they have provided. On the other hand, manual upkeep of up-to-date Bogon filters requires additional work and management on the network. It is best to choose the option that would be the optimal for that network. (Hogg & Vyncke, 2008, 90.)

7 RECONNAISSANCE

Attackers will generally want to scout out for information about a network before reconnaissance launching a possible attack or trying to get control of the network, looking for possible vulnerabilities. Just like in Ipv4, IPv6 has similar problems with some of the features related to IPv6, even though some of them happen to be diminished by the larger address space.

7.1 TECHNIQUES

During reconnaissance the attacker would generally like anonymity in case of detection so having a bogus source address would be beneficial in avoiding authentication. Depending on what technique the attacker is going to use a less efficient technique could be used to avoid detection, sacrificing speed to gain stealth. (Hogg & Vyncke, 2008, 56.)

7.1.1 REGISTRIES

Registries are a part of many so called “tools” for attackers to use to check for information about an IP address or a network. Such as whois for checking information about addresses or domains, nslookup to look up DNS information and IP addresses, using trace route or even simply using search engines to gain information about a specific IP address. In IPv6 especially DNS is a prime suspect for reconnaissance because it can contain information about all the target's IPv6 systems. (Hogg & Vyncke, 2008, 56.)

7.1.2 AUTOMATED SCANNING

Because of the size of the IPv6 subnets it is simply inefficient to brute force scan through the entire network looking for nodes starting from the first possible address and ending at the last possible address of that subnet. It is also quite troublesome since host nodes auto-configure their IPv6 addresses which takes out the chance of there being any predictability about the interface identifier portion of the address. However, port-scanning for known IPv6 addresses is still possible, so if an attacker somehow managed to get their hands on a larger list of IPv6 addresses, the attacker could run it through an automated port-scan. (Hogg & Vyncke, 2008, 56.)

However, if an attacker found out the OUI (Organizational Unique Identifier) for devices such as the routers or the Ethernet cards for the host computers and went under the assumption that they were only using EUI-64 addresses, they could make a scan that simply needed to scan through the last 24 bits of the address. This would need some additional reconnaissance about the actual hardware of the target but could yield better results. On the other hand, on routers things might get easier for attackers because addresses may be configured by administrators for nodes. Because of human error and laziness this might result in routers having easy to remember IP addresses that either resemble words, words could be something like ::f001 or ::abcd. This could give the attacker an opportunity to make a scanning tool which goes through all these easy to remember addresses which the administrator is obviously using to aid themselves in managing the network. Also simply making an automated scan that goes through the lower and higher end of the subnets could yield results in finding possible laziness on the part of the administrator. (Hogg & Vyncke, 2008, 58.)

7.1.3 MULTICAST RECONNAISSANCE

Multicast addresses can be used by attackers in reconnaissance of a network by making connections to multicast addresses which return information about the nodes. This would however, mean that the attacks were done inside the internal LAN which would either mean the attacker had to be inside the organization being able to attach his devices in positions which would yield an opportunity to do this. Other option would be a user being infected by malware enabling the attacker remote access to be able to do these scans. Some of the Multicast addresses would be able to find or receive information about all the nodes in the network or all routers in the network. (Hogg & Vyncke, 2008, 59.)

7.1.4 NEIGHBOR DISCOVERY RECONNAISSANCE

ND can be leveraged for reconnaissance also since if ND is not using SeND it will make nodes vulnerable of sending information to the attacker's node if the attacker somehow is able to connect into the network. The attacker could also possibly infect a node inside the network, perhaps a computer, with a sniffer. The sniffer would listen for NDP messages. (Hogg & Vyncke, 2008, 61.)

Otherwise if the attacker happened to have node in the network, which they either connected into it themselves or if they happened to get remote access to a node, they could simply view the Neighbor Cache. The Neighbor Cache of course holds Layer 2 information about the neighboring nodes in the network, including their MAC addresses. (Hogg & Vyncke, 2008, 62.)

7.2 RECONNAISSANCE TOOLS

There are some tools that are already in use for IPv6 reconnaissance, perhaps the best-known being NMAP (Network Mapper). NMAP is a free security scanner most commonly used for scanning both Ipv4 and IPv6 address spaces and for possible open ports, it also works quite well for single host addresses as well. It can also determine what operating systems hosts are running on and what firewalls or filters are currently being used. It is most commonly used for checking the security of a network for any possible holes and by administrators for upkeep of their network. NMAP works on both Linux distributions and different Windows OSes. (NMAP.)

THC-IPv6 is a toolkit designed to scan IPv6 networks in multiple different ways the kit containing many different features used for different sort of reconnaissance attempts. It has various tools for either scanning specific addresses, advertising yourself as a part of some protocol, faking your address and so on. It also happens to have about 55 tools that can be used in attacks against a network. (THC-IPv6.)

Scapy is a tool that allows you to create and capture packets from both different protocols as well as different requests and replies. It works with both Ipv4 and IPv6 creation and allows for a much more sophisticated attack because of the freedom the attacker receives from creating their own packets. Scapy is available mainly for Linux but has some Windows support, however the IPv6 spectrum of Scapy on Windows OSes is very limited so using it for creating IPv6 packets is mostly limited to Linux distributions. (Scapy.)

7.3 PROTECTION AGAINST RECONNAISSANCE

The options for protecting a network against reconnaissance yields one the benefit of protecting oneself against all other following attacks, the less information the attacker has the better. This is why it is up to the administrator in charge to do whatever they can to mitigate this. The basics would be not to use any obvious addresses for nodes starting from the beginning or the end parts of a subnet. This also holds true for using the "word" type addresses such as ::F001 or common combinations such as ::abcd. It would be wisest for the administrators to use randomized IPv6 addresses for their nodes if they have to manually configure an IPv6 address for them. The Auto-configuration of hosts should also be taken into account, ensuring that the operating

systems on the nodes do support this and actually generate them randomly.(Hogg & Vyncke, 2008, 63-64.)

SEND would also ensure giving an added layer of security versus reconnaissance in the ND department, but because it can cause a load on some nodes and is not actually supported by many Computer OSes and in the OSes that they are very poorly or made complicated, this might be an issue just to ensure further protection versus reconnaissance. Administrators should also take into consideration scanning their networks nodes for possible vulnerable nodes. When found, they should be taken care of by giving them the proper security they need. Also making sure that nodes have their software up-to-date especially any security software it would lower the chances of Malware infecting the nodes enabling the attackers to snoop around the network. (Hogg & Vyncke, 2008, 64.)

Dual-stacking should also be taken into consideration because IPv6 will not be only used natively by all networks. This means that Ipv4 could still be leveraged for reconnaissance so that has to be taken into account. Brute force scanning is quite difficult in the IPv6 address space, getting information about the network would be quite valuable to the attackers, meaning any device that could hold this information should be well protected. This includes devices such as DHCP servers or possibly Logs. (Hogg & Vyncke, 2008, 64.)

8 EMPIRICAL PART

The testing environment chosen for testing the mitigations for the subject's issues that were going to be examined was the Cisco Lab at Kymenlaakso University of Applied Sciences. This was because the lab itself had gear set up conveniently for use both in general and for the specific tasks that were going to be performed. The lab was also convenient because it allowed for the testing of these things without it having any sort of negative effect on any other network. Previously in the original project, subjects were tested in the Simunet environment but due to the complexity of the Simunet network it would be smarter to do it in a separate network at least first before it was implemented in any way in the Simunet network.

8.1 EXTENSION HEADERS

For the empiric part of the thesis there was time to work with both mitigating some of the troublesome Extension Headers that would be good to filter out and mitigated.

These would be the Router Header Type 0, Router Alert and Fragmentation extension headers. These are easily dealt with via implementing ACL filtering on the devices allowing them to block any packets containing the above-mentioned extension headers.

Because there needed to be a way to create and send and receive responses based on extension headers sent to the target address, a program enabling the creation and sending packets with the extension headers in them was needed. It was decided that Scapy, which was mentioned in many sources related to the subject as a very reliable option for creating both IPv4 and IPv6 packets, would be used. The person doing the testing had already used it during their previous project so they were somewhat familiar with how it worked. What was also learned about the program was that it had very slim support for Windows OSes so a Linux distribution had to be used that would allow one to create the IPv6 packets.

It was decided that Fedora 64-bit distribution would be installed on VMware, VMware being a virtual machine, this would allow the freedom of both safely testing and downloading items on it without risking any infection if any other tools would be tested, considering some of the more questionable attack tools came from quite questionable sites. Scapy however is mainly used for administrators for testing, like most tools are nevertheless it was more convenient to install it as a virtual machine in the testing environment being used.

For the “attack” setup itself two routers were used, one being a Cisco 2911 that acted as the filtering part of the setup. This is where the ACL would block the packets that had the extension header attached to it and possibly log when the packets were encountered and blocked for ease for the administrator to track down the attack whenever it may happen. The other router was a Cisco 2801 that acted as the target of the “attack”. This is where all the attack packets were sent, first passing through the filtering router and then arriving here, the loopback address was to be used as the “Target” IP address for the attacks. Below can be seen the setup for the environment in figure 3.

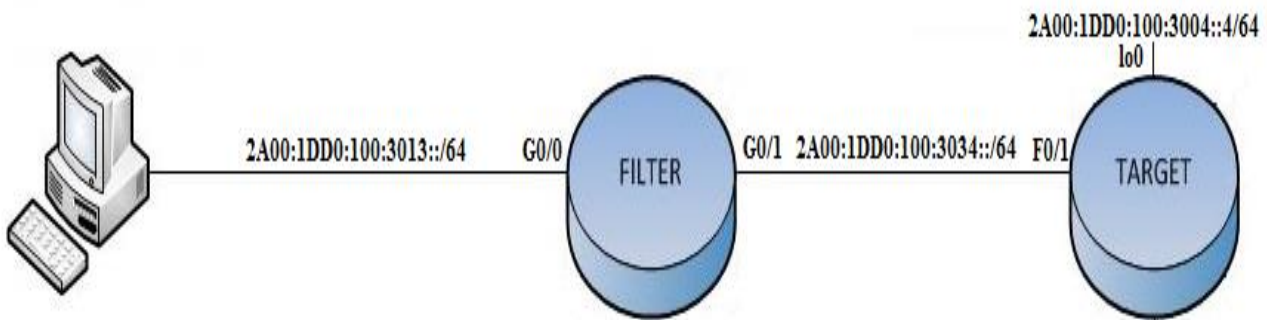


Figure 3. Setup for Routing Header Labs

8.1.1 ROUTING HEADER TYPE 0 FILTERING

As mentioned before in the theory on Extension headers, RH0 allows for DoS attacks and could be used for avoiding specific paths that would possibly lead the packets to firewalls or devices which would filter them out. This is why they should be filtered out introducing safety in one's network.

The first thing that had to be done was creating a RH0 packet that would be sent to the Target IP. This was done by creating the packet in Scapy, the creation of packet looked as following.

```
>>> i=IPv6()
>>> i.dst="2A00:1dd0:100:3004::4"
>>> h=IPv6ExtHdrRouting()
>>> h.addresses=["2A00:1dd0:100:3034::3","2A00:1dd0:100:3034::4"]
>>> p=ICMPv6EchoRequest()
>>> pa=(i/h/p)
>>> ans=sr1(pa)
>>> ans.show()
```

To break it down briefly the steps will be explained. The first thing done was `i=IPv6()` which means that it is a IPv6 packet, then the packet's destination address was set by doing `i.dst=" 2A00:1dd0:100:3004::4"` making the destination address of the packet that of the target routers Loopback address.

After this was done the next step was to do `h=IPv6ExtHdrRouting()` which was the RH0 extension header and on top of that add `h.addresses=["2A00:1dd0:100:3034::3","2A00:1dd0:100:3034::4"]` that were the addresses it should go through before going to the destination address. In a larger test environment or actual network one could make it go any path you wanted too.

As the last step the command `p=ICMPv6EchoRequest()` was used. It is a simple ICMPv6 Echo Request sending back an echo after the packet has reached its target letting one inspect the received packet.

After doing steps the actual packet was created by doing `pa=(i/h/p)` creating the packet where the `i` variable is first, `h` second and `p` last. If one tries to make a packet with where specific portions come in different orders the program will simply tell them this could not be done, since even if they were able to send it, it would be discarded due to the devices not being able to process it.

Then as the next step the packet was sent by doing `ans=srl(pa)`, which sent out the packet and then records the response into the `ans` variable. One could also do `ans,unans=srl(pa)` since Scapy seemed to be very specific on how it reads some packets so one might have to read different packets in different ways.

The final step was to read what the response was by doing `ans.show()` it gives information on the packet that was sent there but the important part.

ICMPv6 Parameter Problem

type= Parameter problem

code= erroneous header field encountered

cksum= 0xe0a7

ptr= 42

From studying different examples of using extension headers with Scapy from other sources this means that the packet went through even though it gave this parameter problem text. This meant the packets were not dropped and actually reached their destination.

The way one blocks this sort of packet from entering a network is by creating an access-list on a device that supports the making of ACLs and can handle them. To create the IPv6 ACL to block RH0 packets one has to use the following commands. The following commands have to be done in the Global Configuration Mode.

IPv6 access-list FILTER

```
deny IPv6 any any routing-type 0
```

```
permit IPv6 any any
```

```
exit
```

This creates the ACL itself but that is not enough, it has to also be applied to the interface it is going to filter. The commands themselves within in the ACL are basically telling to deny any packets from any source to any destination containing the RH0 extension header. The lower command simply states that all other traffic is permitted. The ACL is applied to the G0/0 port, the port leading to the machine sending the packets, this means all traffic coming from the G0/0 will be filtered using the access-list. The commands to apply the access-list to the port are as following, these have to be preformed in the Global Configuration Mode.

interface GigabitEthernet0/0

```
IPv6 traffic-filter FILTER in
```

```
exit
```

Now the access-list is applied as a traffic-filter for the G0/0 port and is filtering all the traffic through the access-list. This should mean that RH0 packets no longer are allowed to pass through. The packet was sent from the machine once more using Scapy and got a different response from in part of the output.


```
###[ ICMPv6 Destination Unreachable ]###
```

```
type= Destination unreachable
```

```
code= Communication with destination administratively prohibited
```

```
cksum= 0xe3f1
```

```
unused= 0x0
```

As can be seen above it noted that the packet was administratively prohibited from communicating with the Target router due to the access-list filtering the packets containing RH0 extension headers. To confirm that the packets were actually filtered by the Filter router the following command was used in the privileged EXEC mode.

```
show IPv6 access-list
```

This command will display information about the access-lists that have been created such as which ACLs are currently on the devices and if there has been any matching packets to said ACLs. The Filter router gave the following prompt after entering the command.

```
IPv6 access list FILTER
```

```
deny IPv6 any any routing-type 0 (1 match) sequence 10
```

```
permit IPv6 any any (20 matches) sequence 20
```

As can be seen, there had been 1 match for packets containing routing-type 0 headers and 20 which have not contained any routing-type 0 headers.

There is another way of also avoiding RH0 being used and that is by disabling IPv6 source routing from the devices. However, this will not actually stop the packets but just not allow them to use the source routing on that device, which mean somewhere along the line some device might allow it to use it and then start using the routing

header destinations again. It can also cause problems if RH2 is in any way used in the network because that is required for it to work. The command to turn off IPv6 source routing is as follows. The command is performed in the Global Configuration Mode.

no IPv6 source-route

In most devices it is most likely that it has already been set to be off by default, but it would be good to know if it is on or not depending on if the network requires it to be used. All in all the smartest option is to filter out unwanted RH0 packets from the network by the use of ACLs.

8.1.2 ROUTER ALERT FILTERING

As stated before about Router Alert it could possibly be used in DoS attacks versus routers, where they are told to view the packets more carefully. An attacker could possibly make use of this by adding a massive amount of data for the router to process causing a denial-of-service attack.

Using the same setup with the Scapy machine as the attacker, the filtering router and the target router, the following packets were created by using Scapy.

```
>>> i=IPv6()
>>> i.dst="2A00:1dd0:100:3004::4"
>>> h=IPv6ExtHdrDestOpt(nh=6, options=[RouterAlert()])
>>> p=ICMPv6EchoRequest()
>>> pa=(i/h/p)
>>> ans=sr1(pa)
>>> ans.show()
```

This packet is very similar to the RH0 packet made earlier with Scapy. However, this packet had the h value, which was used for the header in the previous packet, change

to use an option header which was using the Router Alert option. It also happens to contain a next header value which points to TCP thus nh=6. This is because the next header value for TCP is 6. The return packets cannot be read because there are no return packets but they are getting through.

The way to block this is via ACL on the device to filter out any packets containing the RA option header option. This is performed by doing the following commands.

IPv6 access-list BLOCKRA

```
deny IPv6 any any dest-option-type 5 log  
permit IPv6 any any  
exit
```

interface GigabitEthernet0/0

```
IPv6 traffic-filter BLOCKRA in  
exit
```

As can be seen in the ACL configuration it denies any packets containing destination option type 5, which is the RA option and logging was also turned on. If any packets containing RA pass through the interface they are both blocked and logged.

After applying the filter, another packet was sent containing the RA header from the Scapy machine, then once again the ACL was checked to see if it has encountered any packets that it would have blocked. The prompt received was the following:

IPv6 access list BLOCKRA

```
deny IPv6 any any dest-option-type 5 log (1 match) sequence 10  
permit IPv6 any any (4 matches) sequence 20
```

As can be seen, 1 such packet was encountered and filtered out as the ACL detected the packet containing the RA header.

8.1.3 FRAGMENTATION FILTERING

Fragmentation could be used in a network to get past firewalls without detection and thus could be a major security threat. However, filtering out fragments may also cause problems in transporting data in the network possibly blocking fragmented packets from a non-hostile host. This is why fragmentation filtering should only be applied in places it is really required.

Using the same setup as before a packet was created containing a fragmentation header with Scapy using the following commands:

```
>>> i=IPv6(nh=44)
>>> i.dst='2A00:1dd0:100:3004::4'
>>> h=IPv6ExtHdrFragment(nh=6, offset=100, id=2, m=1)
>>> t=TCP(sport=1080, dport=80, flags="S")
>>> pa=(i/h/t)
>>> ans,unans=sr(pa)
>>> print(ans)
```

This packet had to be a little more exact due to how Scapy apparently handled the fragmentation header. The TCP section had to be added because of this, which the fragmentation header was pointing to as its next header. There were also problems with reading the reply packet which was done by using the command `show` `show.ans()` so the command `print(ans)` was used instead which gives a somewhat different prompt. However, it gave the same text to confirm that the packet had actually passed through to its target destination.

<ICMPv6ParamProblem type=Parameter problem code=erroneous header field encountered cksum=0x291 ptr=4

As usual to block packets containing the fragmentation header one had to use a ACL. This had to be applied on the packet passed through to reach its destination. The commands were the following:

IPv6 access-list FRAGBLOCK

deny IPv6 any 2A00:1DD0:100:3004::/64 log fragments

permit IPv6 any any

interface GigabitEthernet0/0

IPv6 traffic-filter FRAGBLOCK in

exit

As mentioned before because blocking fragmentation could cause problems in the network it would be wise to more precisely filter out the packets containing fragmentation so the filter only blocks packets with the destination of the target IP. Logging was also enabled for the packets containing the fragmentation header destined to the target IP.

As a new packet was sent from the Scapy machine and then using the command **print(ans)** a different reply was received containing the following line:

<ICMPv6DestUnreach type=Destination unreachable code=Communication with destination administratively prohibited cksum=0x5b5 unused=0x0

Not only this confirms that the packet has been blocked since enabling logging also prompted the device to display the following text as the packet was filtered out.

Mar 11 14:11:47.351: %IPv6_ACL-6-ACCESSLOGP: list FRAGBLOCK/50 denied tcp 2A00:1

DD0:100:3013:20C:29FF:FE4D:473B(0) -> 2A00:1DD0:100:3004::4(0), 1 packet

As can be seen, the device logged the event of the packet containing the fragmentation header being captured and filtered.

8.1.4 CONCLUSION TO EXTENSION HEADERS TESTING

Because of what threats some of the extension headers cause it seems like a very good idea to mitigate the problems they pose. Also taking into the account the ease in which they are mitigated there seems to be no reason why they should not be filtered out unless it would cause problems in the general network, which blocking most of them would not cause.

8.2 SEND IMPLEMENTATION

As described before, SEND is a protocol designed to mitigate threats aimed at the Neighbor Discovery protocol. However, there are a few problems with it. Besides the fact that the protocol itself is vulnerable to attacks it also has very poor support by operating systems. Microsoft does not support it at all, Linux however supports it but it requires you to edit the kernel itself to enable it to actually use it. The main reason why Microsoft has not implemented SEND into their OS is because it is a very complex protocol to manage, as well as the fact that it can require a great deal of resources to upkeep.

However, to further study and examine the SEND protocol and to see how complicated it is to set up and any problems that may arise along the way it was decided that setting up a SEND network would quite possibly be beneficial to see how it worked. In the setup for making the test environment, three Cisco 2911 were used running on Version 15.2(2)T1 of the IOS since they all supported SEND, one which was the certificate servers answering to any certificate requests from devices, a router that would work as the default router also advertising out a prefix that has been certified by the

certificate server and lastly a host simply to work with the router and receive the prefix. There were some oversights during configuration which will be addressed in the upcoming sections. Below can be seen the setup for the lab (figure 4).

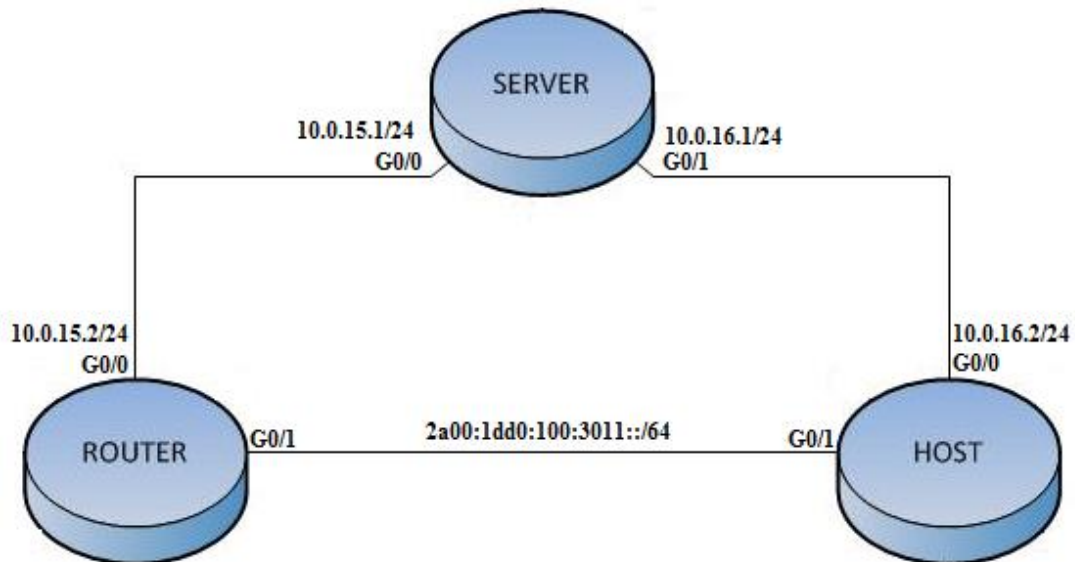


Figure 4. SEND test environment

8.2.1 SEND CONFIGURATION

The first step was configuring the Certificate Server or the CA (Certificate Authority) by making the “Server” router the Certificate Authority by using the following commands in Global Configuration mode.

```
crypto key generate rsa label ca modulus 1024
```

```
ip http server
```

```
crypto pki trustpoint CA
```

```
ip-extension prefix 2A00:1DD0:100:3011::/64
```

```
revocation-check crl
```

```
rsakeypair CA
```

```
crypto pki server CA
```

```
grant auto
```

```
hash sha1
```

```
cdp-url http://10.0.15.1
```

```
no shutdown
```

```
exit
```

First a RSAkeypair was generated which will be used in the certification process. After that SCEP (Simple Certificate Enrollment Protocol) was enabled by enabling IOS HTTP server on the device. Then a crypto pki trustpoint was created which will allow routers to only advertise the specific prefix mentioned in SEND, setting the revocation-check to use CRL(Certificate Revocation List) and binding the rsakeypair created to the trustpoint.

After that the server itself was created, granting certification was made to be automatic in this case because it is a safe testing environment. The hashing of the certificates was set to use sha1 encryption, the cdp-url name was set to http://10.0.15.1 used for the CRL and then the no shutdown command was used to start the server.

After enabling the server the following commands were used on the "Router" router:

```
crypto key generate rsa label SEND modulus 1024
```

```
crypto pki trustpoint ROUTER
```

```
enrollment url http://10.0.15.1:80
```

```
revocation-check crl
```

```
rsakeypair SEND
```

```
exit
```

```
crypto pki trustpoint authenticate ROUTER
```

```
% Do you accept this certificate? [yes/no]: yes
```


crypto pki enroll ROUTER

An RSAkeypair was generated labeled SEND, then a trustpoint was created called ROUTER which set the enrollment URL for the certificate to the CA. Set the revocation-check to use CRL and bound the RSAkeypair SEND to the trustpoint. Then authenticated the CA and accepted the certificate and enrolled it, it will ask for a password also for revoking the certificate later and asked if the router serial number should be in the subject name and if an IP address should be in the subject name. It will then send the certificate request to the CA which would auto grant it since it was set to do so.

After being done with the “Router” the “Host” device needed to be configured.. Because this was all slightly confusing and figuring out how all this worked the decision was made to turn the device into a host after receiving the certificate. So the following commands were used.

crypto key generate rsa label SEND modulus 1024

crypto pki trustpoint CLIENT

enrollment url http://10.0.15.1:80

revocation-check none

rsakeypair SEND

exit

crypto pki trustpoint authenticate CLIENT

% Do you accept this certificate? [yes/no]: yes

crypto pki enroll CLIENT

no ip routing

This was very similar to the router except the revocation-check was not set to be CRL so it was not allowed to operate as the default router and does not receive the rights to advertise the prefix certified by the CA. The device was also turned into a host device

by issuing the **no ip routing** command. After the certificates had been enrolled the following commands were be done on the router:

```
IPv6 cga modifier rsakeypair SEND sec-level 1
```

```
interface GigabitEthernet0/1
```

```
IPv6 cga rsakeypair SEND
```

```
IPv6 address FE80:: link-local cga
```

```
IPv6 nd prefix 2A00:1DD0:100:3011::/64
```

```
IPv6 nd secured trustpoint ROUTER
```

```
IPv6 nd secured timestamp delta 1000
```

```
exit
```

The RSAkeypair is bound with the CGA hashing for the encryption of the addresses and ND. Then in the interface pointing to the client host and bind the RSAkeypair in the interface as well as generate a link-local address encrypted with CGA, advertised the prefix out to the host, sat a secured trustpoint and add a secured time stamp. Then on the Client device the following commands were performed:

```
IPv6 cga modifier rsakeypair SEND sec-level 1
```

```
interface GigabitEthernet0/1
```

```
IPv6 cga rsakeypair SEND
```

```
IPv6 address autoconfig
```

```
IPv6 nd secured trustanchor CLIENT
```

```
IPv6 nd secured timestamp delta 1000
```

On the Client device a very similar sequence of commands was configured except a link-local address was not generated and a trustanchor made rather than a trustpoint. However, at this point some problems arose as SEND would not start operating correctly. After some investigation it was discovered that the device clocks were not synched which meant the clocks had to be set manually. The best alternative however would have been to set a NTP server but because of time restraints this was the best option at the moment. After the clocks were set SEND started to operate as it should. Using some checks it could be confirmed that the auto-configured device on the Client was using SEND by using the following command:

```
show IPv6 cga address-db
```

```
2A00:1DD0:100:3011::/64 ::2CBF:5542:1173:93A5 - table 0x0
```

```
interface: GigabitEthernet0/1 (4)
```

```
modifier: SEND
```

```
collisions: 0
```

The IPv6 neighbors were also checked, before the clocks were fixed nothing was showing up in the neighbor cache but after fixing the clocks the following prompt was shown.

```
show IPv6 neighbors
```

```
IPv6 Address                Age Link-layer Addr State Interface  
FEA1:1111:1111:1111:38AF:9590:64CE:4999  17 c464.13f5.dbe1 STALE Gi0/1
```

Interface G0/1 was also viewed to see if it had set the default router correctly which it had.

```
SENDCLIENT#show IPv6 int g0/1
```

```
GigabitEthernet0/1 is up, line protocol is up
```

IPv6 is enabled, link-local address is FE80::C664:13FF:FEF6:161

No Virtual link-local address(es):

Stateless address autoconfig enabled

Global unicast address(es):

**2A00:1DD0:100:3011:2CBF:5542:1173:93A5, subnet is
2A00:1DD0:100:3011::/64 [C**

AL/PRE]

valid lifetime 2591913 preferred lifetime 604713

Joined group address(es):

FF02::1

FF02::2

FF02::1:FF73:93A5

FF02::1:FFF6:161

MTU is 1500 bytes

ICMP error messages limited to one every 100 milliseconds

ICMP redirects are enabled

ICMP unreachable are sent

ND DAD is enabled, number of DAD attempts: 1

ND reachable time is 30000 milliseconds (using 30000)

ND NS retransmit interval is 1000 milliseconds

Default router is FEA1:1111:1111:1111:38AF:9590:64CE:4999 on Giga-bitEthernet0/

8.2.2 CONCLUSION TO SEND CONFIGURATION

This is the point the testing of SEND was halted. It was very hard to find any information about the subject and getting it to work was also quite the hassle considering one had to learn how CAs work and figuring out some of the problems that were en-

countered along the way. While SEND does provide additional security to ND it seems as if it should be only implemented in the most serious situations where security is a must. This would possibly be to a server running on Linux or devices which really need to be secured for ND. For general use it is hard to think of a way how this will become popular unless Microsoft backs it.

9 CONCLUSION TO THESIS

The aim of this study was to generally research IPv6 security. In the subject areas the most time was put into researching problems with both Extension Headers and Neighbor Discovery. Time was also put into researching other aspects such as reconnaissance and Bogons. Also the mitigations for both Extension Headers and ND were thoroughly researched, finding out how Extension Headers are filtered out and what options were for securing some of the functions in Neighbor Discovery Using RAGuard or SEND.

As a whole IPv6 seems more secure than IPv4 yet it still has problems which need to be addressed. An intelligent guess would be that most of these problems would not be apparent if when designing IPv6, the architects would have known about all the problems within IPv4 more thoroughly, since similar problems arise here also. These could have probably been avoided.

The problematic Extension Headers should be filtered out if they are not required in a network, considering the low cost of filtering them. The filtering still has to take into account the fact that it might cause problems within the network somehow and thus should be implemented by taking into consideration how it is done.

ND seems quite problematic and the current ways of securing it seem quite “weak”. RAGuard only protects from one aspect and has possible way of spoofing it yet SEND could secure it quite well but is not supported by most operating systems natively and is quite troublesome to even to get to work.

While reconnaissance can be mitigated by intelligent choices on managing a network and possibly even implementing SEND and configuring devices to not respond to different kinds of messages, reconnaissance would most likely always be a problem when it comes to networks. The only thing to be done is to make it as difficult for the attackers as possible.

All in all more attention and time should be spent in the security aspects of IPv6 considering the size and the benefits it reaps if done correctly. However, the fact that security may cripple a network should always be taken into account.

REFERENCES

Hogg, S. & Vyncke, E. 2008. IPv6 Security. Indianapolis: Cisco Press.

Blanchet, M. 2006. Migrating to IPv6: A practical guide to implementing IPv6 in Mobile and Fixed Networks. Chichester: John Wiley & Sons Ltd.

RFC2460, Internet Protocol, Version 6 (IPv6) Specification. IETF (Internet Engineering Task Force). Available: <http://www.ietf.org/rfc/rfc2460.txt> (referenced 30.1.2014).

RFC6265, Mobility Support in IPv6. IETF. Available: <http://www.ietf.org/html/rfc6275.txt> (referenced 30.1.2014).

Koodli, R. & Perkins, C. 2007. Mobile Inter-networking with IPv6: Concepts, Principles and Practices: New Jersey: John Wiley & Sons Ltd.

RFC2460. 1998. Neighbor Discovery for IP Version 6 (IPv6), Available: <http://www.ietf.org/rfc/rfc2461.txt> (referenced 5.2.2014).

RFC3971. 2005. Secure Neighbor Discovery (SEND). Available: <http://tools.ietf.org/html/rfc3971.txt> (referenced 20.2.2014).

RFC 3972. Cryptographically Generated Addresses (CGA). Available: <https://tools.ietf.org/rfc/rfc3972.txt> (referenced 20.2.2014).

Cisco. 2011. Implementing First Hop Security in IPv6. Available: http://www.cisco.com/c/en/us/td/docs/ios/IPv6/configuration/guide/12_4t/IPv6_12_4t_book/ip6-first_hop_security.html (referenced 20.2.2014).

Cisco. 2012. IPv6 RA Guard. Available: <http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/IPv6/configuration/15-2s/ip6-15-2s-book/ip6-ra-guard.html> (referenced 10.3.2014).

v6ops. 2011. IPv6 Router Advertisement Guard (RA-GUARD) Evasion. Available: <http://tools.ietf.org/id/draft-gont-v6ops-ra-guard-evasion-00.txt> (referenced 10.03.2014).

NMAP. Available: <http://nmap.org/book/man.html#man-description> (referenced 21.2.2014).

THC-IPv6. 2013. THC-IPv6 Available: <https://www.thc.org/thc-IPv6/> (referenced 21.2.2014).

Scapy. SecDev. Available: <http://www.secdev.org/projects/scapy/> (referenced 21.2.2014).

RFC3756. 2004. Available: <http://tools.ietf.org/html/rfc3756> (referenced 10.03.2014).

RFC6104. 2011. Available: <http://tools.ietf.org/html/rfc6105> (referenced 10.03.2014).

Appendices

APPENDIX 1: Extension Header Practical Work Filtering Router Configuration.

```
Building configuration...
```

```
Current configuration : 2292 bytes
```

```
!
```

```
! Last configuration change at 14:03:18 UTC Tue Mar 11 2014
```

```
version 15.2
```

```
service timestamps debug datetime msec
```

```
service timestamps log datetime msec
```

```
no service password-encryption
```

```
!
```

```
hostname Filter
```

```
!
```

```
boot-start-marker
```

```
boot-end-marker
```

```
!
```

```
!
```

```
!
```

```
no aaa new-model
```

```
!
```

```
!
```

```
ipv6 unicast-routing
```

```
ipv6 cef
```

```
ip auth-proxy max-login-attempts 5
```

```
ip admission max-login-attempts 5
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```



```
!  
  
interface Embedded-Service-Engine0/0  
  
no ip address  
  
shutdown  
  
!  
  
interface GigabitEthernet0/0  
  
no ip address  
  
duplex auto  
  
speed auto  
  
ipv6 address 2A00:1DD0:100:3013::3/64  
  
ipv6 nd prefix 2A00:1DD0:100:3013::/64  
  
ipv6 ospf 1 area 0  
  
ipv6 traffic-filter FRAGBLOCK in  
  
!  
  
interface GigabitEthernet0/1  
  
no ip address  
  
duplex auto  
  
speed auto  
  
ipv6 address 2A00:1DD0:100:3034::3/64  
  
ipv6 ospf 1 area 0  
  
!  
  
interface GigabitEthernet0/2  
  
no ip address  
  
shutdown  
  
duplex auto  
  
speed auto  
  
!  
  
interface Serial0/1/0  
  
no ip address  
  
shutdown  
  
!  
  
interface Serial0/1/1  
  
no ip address  
  
shutdown
```

```
clock rate 2000000

!

!

ip forward-protocol nd

!

no ip http server

no ip http secure-server

!

!

ipv6 router ospf 1

router-id 10.0.0.3

auto-cost reference-bandwidth 100000

passive-interface default

no passive-interface GigabitEthernet0/1

!

!

!

!

ipv6 access-list BLOCKRA

deny ipv6 any any dest-option-type 5 log

permit ipv6 any any

!

ipv6 access-list FILTER

deny ipv6 any any routing-type 0

permit ipv6 any any

!

ipv6 access-list FRAGBLOCK

permit 88 any any

permit 103 any any

permit icmp any any router-advertisement

permit icmp any any router-solicitation

deny ipv6 any 2A00:1DD0:100:3004::/64 log fragments

permit ipv6 any any

!
```

```
control-plane
!
!
!
line con 0
line aux 0
line 2
no activation-character
no exec
transport preferred none
transport input all
transport output lat pad telnet rlogin lapb-ta mop udptn v120 ssh
stopbits 1
line vty 0 4
login
transport input all
!
scheduler allocate 20000 1000
!
end
```

APPENDIX 2: Extension Header Practical Work Target Router Configuration.

Building configuration...

Current configuration : 1632 bytes

```
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
```

```
hostname Target
!
boot-start-marker
boot-end-marker
!
logging message-counter syslog
!
no aaa new-model
memory-size iomem 10
!
dot11 syslog
ip source-route
!
!
ip cef
!
!
ipv6 unicast-routing
ipv6 cef
!
multilink bundle-name authenticated
!
!
!
!
!
!
!
!
!
!
!
!
```

```
!  
!  
!  
!  
!  
!  
!  
!  
!  
!  
voice-card 0  
!  
!  
!  
!  
!  
archive  
log config  
hidekeys  
!  
!  
!  
!  
!  
!  
!  
!  
!  
!  
interface Loopback0  
no ip address  
ipv6 address 2A00:1DD0:100:3004::4/64  
ipv6 ospf 1 area 0  
!  
interface FastEthernet0/0
```

```
no ip address

shutdown

duplex auto

speed auto

!

interface FastEthernet0/1

no ip address

duplex auto

speed auto

ipv6 address 2A00:1DD0:100:3034::4/64

ipv6 ospf 1 area 0

!

interface Serial0/1/0

no ip address

shutdown

no fair-queue

clock rate 2000000

!

interface Serial0/1/1

no ip address

shutdown

clock rate 2000000

!

interface ATM0/3/0

no ip address

shutdown

no atm ilmi-keepalive

!

interface wlan-controller1/0

no ip address

shutdown

!

ip forward-protocol nd

no ip http server
```



```
no ip http secure-server
!
!
!
ipv6 router ospf 1
router-id 10.0.0.4
log-adjacency-changes
auto-cost reference-bandwidth 100000
passive-interface default
no passive-interface FastEthernet0/1
!
!
!
!
!
!
!
!
control-plane
!
!
!
!
mgcp fax t38 ecm
mgcp behavior g729-variants static-pt
!
!
!
!
!
!
line con 0
line aux 0
line 66
no activation-character
```

```
no exec

transport preferred none

transport input all

transport output pad telnet rlogin lapb-ta mop udptn v120 ssh

line vty 0 4

login

!

scheduler allocate 20000 1000

end
```

APPENDIX 3: SEND CA Configuration

Building configuration...

Current configuration : 3309 bytes

!

! Last configuration change at 11:50:08 UTC Mon Feb 24 2014

version 15.2

service timestamps debug datetime msec

service timestamps log datetime msec

no service password-encryption

!

hostname CA

!

boot-start-marker

boot-end-marker

!

!

!

no aaa new-model

!

!

ipv6 unicast-routing

ipv6 cef

ip auth-proxy max-login-attempts 5

ip admission max-login-attempts 5

!

!

!

!

!

ip cef

!

multilink bundle-name authenticated

!

!

!

crypto pki server CA

grant auto

hash sha1

cdp-url http://10.0.15.1

crypto pki token default removal timeout 0

!

crypto pki trustpoint CA

ip-extension prefix 2A00:1DD0:100:3011::/64

revocation-check crl

rsakeypair CA

!

!

crypto pki certificate chain CA

certificate ca 01

3082021B 30820184 A0030201 02020101 300D0609 2A864886 F70D0101 05050030

0D310B30 09060355 04031302 4341301E 170D3134 30323234 31313130 31325A17

0D313730 32323331 31313031 325A300D 310B3009 06035504 03130243 4130819F

300D0609 2A864886 F70D0101 01050003 818D0030 81890281 8100CEEC B1CD3357

DD4058C4 6219E3F0 0A1830F5 6A7FAEAE 7100C979 EDD99359 4DE7905F EDE4582C

6CFF85AD 8C604561 2C2CB6C1 BAA7F961 D62C6DEE 243AD19F 47E50A3B A0A4B8C6

C324888F 461F203F DDA5DF5B F3516043 F61BC4A7 706D4910 BA2E9660 86EA38A8

DF199714 65E5AD57 F418D7AA 7154A869 4BD57BFC 85BBB4AC E31B0203 010001A3

818A3081 87300F06 03551D13 0101FF04 05300301 01FF300E 0603551D 0F0101FF

04040302 01863024 06082B06 01050507 01070101 FF041530 13301104 02000230

0B030900 2A001DD0 01003011 301F0603 551D2304 18301680 14778F17 8F2C60C0

122C6E5A 187FEDE3 79475AAD 59301D06 03551D0E 04160414 778F178F 2C60C012

2C6E5A18 7FEDE379 475AAD59 300D0609 2A864886 F70D0101 05050003 81810018

0BA244E6 17BB7F66 C98DD384 954C9E4A 571348F8 21D13533 0E0E2ECA BA80D876

916CED89 53E43109 3862D285 A580D798 9E28ACF1 2EE02E70 8CD5260C 4CC49157

33467C58 90FE9862 C32A85D7 A1D9C540 2A39A5BE AC2DA077 73D924B4 03123E1F

EF722378 B9DDAACF C3FA619E 6F96C346 8D68AC87 53330A80 14993BC6 693D1E

quit

license udi pid CISCO2911/K9 sn FCZ160570LU

license boot module c2900 technology-package securityk9

license boot module c2900 technology-package datak9

!

!

vtp mode transparent

!

redundancy

!

!

!

!

!

!

!

!

!

!

!

!

!

!

interface Loopback0

ip address 10.0.0.1 255.255.255.255

!

interface Embedded-Service-Engine0/0

no ip address

shutdown

!

interface GigabitEthernet0/0

ip address 10.0.15.1 255.255.255.0

duplex auto

speed auto

!

interface GigabitEthernet0/1

ip address 10.0.16.1 255.255.255.0

duplex auto

speed auto

!

interface GigabitEthernet0/2

```
no ip address
```

```
shutdown
```

```
duplex auto
```

```
speed auto
```

```
!
```

```
interface Serial0/1/0
```

```
no ip address
```

```
shutdown
```

```
clock rate 2000000
```

```
!
```

```
interface Serial0/1/1
```

```
no ip address
```

```
shutdown
```

```
clock rate 2000000
```

```
!
```

```
!
```

```
router ospf 1
```

```
router-id 10.0.0.1
```

```
auto-cost reference-bandwidth 100000
```

```
passive-interface default
```

```
no passive-interface GigabitEthernet0/0
```

```
no passive-interface GigabitEthernet0/1
```

```
network 10.0.0.0 0.255.255.255 area 0
```

!

ip forward-protocol nd

!

ip http server

no ip http secure-server

!

!

!

!

!

control-plane

!

!

!

line con 0

line aux 0

line 2

no activation-character

no exec

transport preferred none

transport input all

transport output lat pad telnet rlogin lapb-ta mop udptn v120 ssh

stopbits 1


```
line vty 0 4
```

```
login
```

```
transport input all
```

```
!
```

```
scheduler allocate 20000 1000
```

```
!
```

```
end
```

APPENDIX 4: SEND ROUTER Configuration

Building configuration...

Current configuration : 4696 bytes

```
!
```

! Last configuration change at 11:30:32 UTC Mon Feb 24 2014

```
version 15.2
```

```
service timestamps debug datetime msec
```

```
service timestamps log datetime msec
```

```
no service password-encryption
```

```
!
```

```
hostname DefRouter
```

```
!
```

```
boot-start-marker
```

```
boot-end-marker
```

```
!
```

!

!

no aaa new-model

!

memory-size iomem 10

!

ipv6 unicast-routing

ipv6 cga modifier rsakeypair SEND sec-level 1 90A4:C546:7C30:6236:8B14:7624:6D1D

:70F8

ipv6 cef

ip auth-proxy max-login-attempts 5

ip admission max-login-attempts 5

!

!

!

!

!

ip cef

!

multilink bundle-name authenticated

!

!

crypto pki token default removal timeout 0

!

crypto pki trustpoint ROUTER

enrollment url http://10.0.15.1:80

revocation-check crl

rsakeypair SEND

!

!

crypto pki certificate chain ROUTER

certificate 02

3082020F 30820178 A0030201 02020102 300D0609 2A864886 F70D0101 05050030

0D310B30 09060355 04031302 4341301E 170D3134 30323234 31313138 35355A17

0D313530 32323431 31313835 355A301A 31183016 06092A86 4886F70D 01090216

09446566 526F7574 65723081 9F300D06 092A8648 86F70D01 01010500 03818D00

30818902 818100CC DEAD399B 696909E6 11A6723B EBD64591 2B99C375 59F47864

656ADC7E 3DDD6BF2 6E5205AE 38664153 59326365 371D3C0B 8F73DEEA EF908B63

18F3FB14 CA7E2943 C7B7F9D9 F4D69531 F90CAA32 4F67402B C0C7CA97 40A81DE1

06F2E0DC CBF5363F 75799DD6 D915970B 735E603E DB270EB6 5135CF81 3D312151

87985807 8A03E702 03010001 A3723070 30210603 551D1F04 1A301830 16A014A0

12861068 7474703A 2F2F3130 2E302E31 352E3130 0B060355 1D0F0404 030205A0

301F0603 551D2304 18301680 14778F17 8F2C60C0 122C6E5A 187FEDE3 79475AAD

59301D06 03551D0E 04160414 6ED83007 80426980 07750303 5DCCB9A4 1E3BCFEE

300D0609 2A864886 F70D0101 05050003 8181002C 122F6D06 49E5B17D BB34A078

B17E81DB F9CAE84C 1324C9A9 D63725FF 7096057E 1B47AD54 053FB1B9 CA0641EE

F4C8D14A 7CDE29F6 475E9B65 A12DA926 24B33231 1B237967 FB342829 5028620D

DDB05D10 E0F3C6F5 E12003AE 0BD4C4A2 3345F4E1 1721DF2F A7A19A66 2E725C62

FEE1A7D1 EA0B8938 9EE318C9 CD2A932F 6C3155

quit

certificate ca 01

3082021B 30820184 A0030201 02020101 300D0609 2A864886 F70D0101 05050030

0D310B30 09060355 04031302 4341301E 170D3134 30323234 31313130 31325A17

0D313730 32323331 31313031 325A300D 310B3009 06035504 03130243 4130819F

300D0609 2A864886 F70D0101 01050003 818D0030 81890281 8100CEEC B1CD3357

DD4058C4 6219E3F0 0A1830F5 6A7FAEAE 7100C979 EDD99359 4DE7905F EDE4582C

6CFF85AD 8C604561 2C2CB6C1 BAA7F961 D62C6DEE 243AD19F 47E50A3B A0A4B8C6

C324888F 461F203F DDA5DF5B F3516043 F61BC4A7 706D4910 BA2E9660 86EA38A8

DF199714 65E5AD57 F418D7AA 7154A869 4BD57BFC 85BBB4AC E31B0203 010001A3

818A3081 87300F06 03551D13 0101FF04 05300301 01FF300E 0603551D 0F0101FF

04040302 01863024 06082B06 01050507 01070101 FF041530 13301104 02000230

0B030900 2A001DD0 01003011 301F0603 551D2304 18301680 14778F17 8F2C60C0

122C6E5A 187FEDE3 79475AAD 59301D06 03551D0E 04160414 778F178F 2C60C012

2C6E5A18 7FEDE379 475AAD59 300D0609 2A864886 F70D0101 05050003 81810018

0BA244E6 17BB7F66 C98DD384 954C9E4A 571348F8 21D13533 0E0E2ECA BA80D876

916CED89 53E43109 3862D285 A580D798 9E28ACF1 2EE02E70 8CD5260C 4CC49157

33467C58 90FE9862 C32A85D7 A1D9C540 2A39A5BE AC2DA077 73D924B4 03123E1F

EF722378 B9DDAACF C3FA619E 6F96C346 8D68AC87 53330A80 14993BC6 693D1E

quit

```
license udi pid CISCO2911/K9 sn FCZ160570LV
```

```
license boot module c2900 technology-package securityk9
```

```
license boot module c2900 technology-package datak9
```

```
!
```

```
!
```

```
!
```

```
redundancy
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
interface Loopback0
```

```
ip address 10.0.0.2 255.255.255.255
```

!

interface Embedded-Service-Engine0/0

no ip address

shutdown

!

interface GigabitEthernet0/0

ip address 10.0.15.2 255.255.255.0

duplex auto

speed auto

!

interface GigabitEthernet0/1

no ip address

duplex auto

speed auto

ipv6 cga rsakeypair SEND

ipv6 address FE80:: link-local cga

ipv6 nd prefix 2A00:1DD0:100:3011::/64

ipv6 nd secured trustpoint ROUTER

ipv6 nd secured timestamp delta 1000

!

interface GigabitEthernet0/2

no ip address

shutdown

```
duplex auto
```

```
speed auto
```

```
!
```

```
interface Serial0/1/0
```

```
no ip address
```

```
shutdown
```

```
clock rate 2000000
```

```
!
```

```
interface Serial0/1/1
```

```
no ip address
```

```
shutdown
```

```
clock rate 2000000
```

```
!
```

```
!
```

```
router ospf 1
```

```
auto-cost reference-bandwidth 100000
```

```
passive-interface default
```

```
no passive-interface GigabitEthernet0/0
```

```
network 10.0.0.0 0.255.255.255 area 0
```

```
!
```

```
ip forward-protocol nd
```

```
!
```

```
no ip http server
```

```
no ip http secure-server
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
control-plane
```

```
!
```

```
!
```

```
!
```

```
line con 0
```

```
line aux 0
```

```
line 2
```

```
no activation-character
```

```
no exec
```

```
transport preferred none
```

```
transport input all
```

```
transport output lat pad telnet rlogin lapb-ta mop udptn v120 ssh
```

```
stopbits 1
```

```
line vty 0 4
```

```
login
```

```
transport input all
```

```
!
```



```
scheduler allocate 20000 1000
```

```
!
```

```
end
```

APPENDIX 5: SEND CLIENT Configuration

```
Building configuration...
```

```
Current configuration : 3371 bytes
```

```
!
```

```
! Last configuration change at 11:53:06 UTC Mon Feb 24 2014
```

```
version 15.2
```

```
service timestamps debug datetime msec
```

```
service timestamps log datetime msec
```

```
no service password-encryption
```

```
!
```

```
hostname SENDCLIENT
```

```
!
```

```
boot-start-marker
```

```
boot-end-marker
```

```
!
```

```
!
```

```
!
```

```
no aaa new-model
```

```
!
```

memory-size iomem 10

!

ipv6 cga modifier rsakeypair SEND sec-level 1 4F9D:C1C6:6690:A8E7:1D99:1B96:FE3F

:D995

no ipv6 cef

no ip routing

ip auth-proxy max-login-attempts 5

ip admission max-login-attempts 5

!

!

!

!

!

no ip cef

!

multilink bundle-name authenticated

!

!

crypto pki token default removal timeout 0

!

crypto pki trustpoint CLIENT

enrollment url http://10.0.15.1:80

revocation-check none

rsakeypair SEND

!

!

crypto pki certificate chain CLIENT

certificate ca 01

3082021B 30820184 A0030201 02020101 300D0609 2A864886 F70D0101 05050030

0D310B30 09060355 04031302 4341301E 170D3134 30323234 31313130 31325A17

0D313730 32323331 31313031 325A300D 310B3009 06035504 03130243 4130819F

300D0609 2A864886 F70D0101 01050003 818D0030 81890281 8100CEEC B1CD3357

DD4058C4 6219E3F0 0A1830F5 6A7FAEAE 7100C979 EDD99359 4DE7905F EDE4582C

6CFF85AD 8C604561 2C2CB6C1 BAA7F961 D62C6DEE 243AD19F 47E50A3B A0A4B8C6

C324888F 461F203F DDA5DF5B F3516043 F61BC4A7 706D4910 BA2E9660 86EA38A8

DF199714 65E5AD57 F418D7AA 7154A869 4BD57BFC 85BBB4AC E31B0203 010001A3

818A3081 87300F06 03551D13 0101FF04 05300301 01FF300E 0603551D 0F0101FF

04040302 01863024 06082B06 01050507 01070101 FF041530 13301104 02000230

0B030900 2A001DD0 01003011 301F0603 551D2304 18301680 14778F17 8F2C60C0

122C6E5A 187FEDE3 79475AAD 59301D06 03551D0E 04160414 778F178F 2C60C012

2C6E5A18 7FEDE379 475AAD59 300D0609 2A864886 F70D0101 05050003 81810018

0BA244E6 17BB7F66 C98DD384 954C9E4A 571348F8 21D13533 0E0E2ECA BA80D876

916CED89 53E43109 3862D285 A580D798 9E28ACF1 2EE02E70 8CD5260C 4CC49157

33467C58 90FE9862 C32A85D7 A1D9C540 2A39A5BE AC2DA077 73D924B4 03123E1F

EF722378 B9DDAACF C3FA619E 6F96C346 8D68AC87 53330A80 14993BC6 693D1E

quit

```
license udi pid CISCO2911/K9 sn FCZ160570LJ
```

```
license boot module c2900 technology-package securityk9
```

```
license boot module c2900 technology-package datak9
```

```
!
```

```
!
```

```
vtp mode transparent
```

```
!
```

```
redundancy
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
interface Loopback0
```

```
ip address 10.0.0.3 255.255.255.255

no ip route-cache

!

interface Embedded-Service-Engine0/0

no ip address

no ip route-cache

shutdown

!

interface GigabitEthernet0/0

ip address 10.0.16.2 255.255.255.0

no ip route-cache

duplex auto

speed auto

!

interface GigabitEthernet0/1

no ip address

no ip route-cache

duplex auto

speed auto

ipv6 ega rsakeypair SEND

ipv6 address autoconfig

ipv6 nd secured trustanchor CLIENT

ipv6 nd secured timestamp delta 1000
```

!

interface GigabitEthernet0/2

no ip address

no ip route-cache

shutdown

duplex auto

speed auto

!

interface Serial0/1/0

no ip address

no ip route-cache

shutdown

clock rate 2000000

!

interface Serial0/1/1

no ip address

no ip route-cache

shutdown

clock rate 2000000

!

!

ip forward-protocol nd

!

```
no ip http server
```

```
no ip http secure-server
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
control-plane
```

```
!
```

```
!
```

```
!
```

```
line con 0
```

```
line aux 0
```

```
line 2
```

```
no activation-character
```

```
no exec
```

```
transport preferred none
```

```
transport input all
```

```
transport output lat pad telnet rlogin lapb-ta mop udptn v120 ssh
```

```
stopbits 1
```

```
line vty 0 4
```

```
login
```

```
transport input all
```

!

scheduler allocate 20000 1000

!

end