Escola Superior de Tecnologia e Gestão
de Oliveira do Hospital

Master in Applied Informatics

Analysis of the security impact of making mShield an IPv4 to IPv6 converter box

Master's Thesis
By
Luís Rosa

November 2012

Thesis submitted for the partial fulfillment of the requirements for the degree of Master of Science in Applied Informatics at the Higher School of Technology and Management of the Polytechnic Institute of Coimbra

Under supervision of:

Prof. Dr. Luís Veloso - Escola Superior de Tecnologia e Gestão de Oliveira do Hospital

Prof. Dr. Marco Veloso - Escola Superior de Tecnologia e Gestão de Oliveira do Hospital

Internship Supervision:

Holger Mikolon - Philips Medical Systems DMC GmbH Hamburg

Erich J. Heins - Philips Medical Systems DMC GmbH Hamburg

To my Mom.

Acknowledgements

I would like to express my sincere gratitude to my family. To my parents, Luís and Isabel, for their support and encouragement. They are and will be both present in all steps of my life. To Ana, my future wife, I would like to thank for her love, support and patience during all these years. To my sister, Alexandra, for her friendship and to her daughter, Camila, my little niece, for showing me the meaning of life.

I would like to thank my professors Luís Veloso and Marco Veloso, my supervisors at school, for their useful guidance, reviews and feedback. Thank to my supervisor at Philips, Erich, for giving me the opportunity to learn in a fruitful environment. Thank to Holger, more than my supervisor at Philips, for his continous support and motivation as well as the insightful discussions and reviews of my work. Thanks to Sami and Clara for proof reading part of this document. I also would like to thank Philips for the financial support.

For all aforementioned, other colleagues and friends, thank you.

Abstract

Resumo

Contents

List of Figures

List of Tables

List of Listings

One of the greatest gifts you can give to anyone is the gift of attention.

Jim Rohn

1 Introduction

1.1 Background

Nowadays, computers communicate over the network mainly via Internet Protocol version 4 (IPv4). This protocol was proposed in 1981 and has been the standard for communications using TCP/IP architecture. In the last years, there was a significant increase in demand for IPv4 addresses as a result of new network devices and the amount of connected people. The most significant limitation of IPv4 is probably the narrow address space resulting from the limited possible combinations of a 32-bit address. Internet Assigned Numbers Authority (IANA) already assigned the last IPv4 addresses block[1]. In the meanwhile, several Request for Comments (RFC) and documents have been written describing its successor, Internet Protocol version 6 (IPv6).

It is not reasonably expected that all IPv4 devices can be easily changed to IPv6. Therefore, significant efforts are being made, not only in IPv6 specification, but also in defining strategies and mechanisms for a graceful transition phase. One of those transition strategies, provides a framework to translating addresses and packets between both versions.

This thesis, written under an internship in Philips Healthcare, Hamburg, covers the problem of enabling IPv6 communications in IPv4 legacy medical devices. mShield is a physical firewall developed by Philips to operate in a hospital environment. Its role is to protect medical devices from network based attacks. Currently it only supports IPv4.

Given the need for IPv6 transition, the next releases of mShield should also perform IPv4 to IPv6 conversion. This allows legacy medical devices to communicate with IPv6 hospital devices. Developing a medical device requires significant effort and costs, and therefore, it is not feasible to update each legacy device to IPv6.

Nevertheless, the problem is not just about enabling the converter function. It is important to understand that medical devices are operated in environments where it is necessary to protect sensitive patient data.

[1] Full announcement by the Number Resource Organization (NRO) [2]

1.2 Problem Statement

In this thesis, the need to move towards IPv6 is assumed. Moreover, in a medical context it is important to analyse the security implications of a new protocol. This study provides a comprehensive answer to the question:

What is the security impact of making mShield an IPv4 to IPv6 converter box ?

The change involves not only the converter role but also the support for both Internet Protocol (IP) versions (e.g. IPv6 to IPv6 communications). It is necessary to analyse the security issues arising from the change and compare them with the current design.

Even though restricted to mShield and its context, this study can be used as a reference in other studies regarding IPv4 to IPv6 translation covering the topic from the theory to practical implementation.

1.3 Structure of the thesis

In addition to this chapter, the thesis is composed of the following chapters:

Chapter 2 is composed of two parts: IPv6 and security theory reviews. It describes the main topics of IPv6 as well as some of the differences of IPv4 version. One of the focus regards the transition mechanisms, and particularly, the translation strategy used, NAT64. Additionally, a discussion about the vantages and disadvantages of the use of several mechanisms are also provided (e.g. stateless versus stateful address translation). At the end, an overview of the security concepts used in the security analysis is also presented.

Chapter 3 is composed of tree parts: the methodology used, the environment description and the security analysis itself. The methodology includes a description of all performed steps as well as the terminology used during the security analysis. In the environment section, mShield, the medical devices network and customer network as well as features and constraints of the environment are described. Finally, in the security analysis, a security discussion of the issues that affect mShield is provided. Namely the effect of introducing a new IP version. The analysis takes into account the requirements

for the converter box and discusses the security effect of the possibilities for a new design. The assets and threats identified are presented in this chapter. At the end of the analysis, a risk assessment describing and measuring the impact and the likelihood of each threat is also presented. The analysis is based on a quantitative and qualitative analysis relating to each threat with known security threats and most common attacks.

Chapter 4 provides a detailed description of a new mShield design. It is proposed, based on the previous security discussion and the requirements for the converter process. It also contains a detailed description of its implementation. In adition to that, based on the implementation, practical tests are discussed.

Chapter 5 provides a picture of each chapter, highlighting the most relevant conclusions. An overall conclusion of the research question is also provided. In the conclusion, the main contributions and the further work are described. The thesis ends with some final thoughts about the developed work.

Reading without reflecting is like
eating without digesting.

Edmund Burke

2

IPv6 and Security Theory Review

2.1 IPv6 in a nutshell

The differences between IPv4 and IPv6, IPv6 addressing architecture or IPv6 header format are explained in this section. Those IPv6 basics topics will help to understand the next topics and chapters of this thesis.

2.1.1 Overview

The IPv6 specification is defined as a draft standard [3] by Internet Engineering Task Force (IETF). The main differences between IPv4 and IPv6 are:

- Increase of addressing possibilities (due to the 128 bits address length compared with 32 bits in IPv4).
- Header simplification, the number of field in the base header was reduced increasing the processing speed of packets.
- New concept of *Extension Headers* to carry additional options used by non-IP but IP-related protocols.
- New Field, *Flow Label*, allows to group a set of packets as a flow, improving the packet processing for the same flow. This processing improvement is only valid for infrastructure devices, like routers. Some applications or infrastructure devices do not make any use of this field resulting in a normal processing speed.
- Authentication and Privacy features are defined as part of the base protocol using Internet Protocol security (IPsec).

2.1.2 Addressing

The most visible change, regardless the address length, is probably related to broadcast which is not used in IPv6. It was replaced by multicast functions using pre-defined addresses.

Syntax

An IPv6 address can be written by eight groups of four hexadecimal digits separated by colons. In IPv4 the representation is made in dotted decimal notation using four groups between 0 to 255, e.g. *255.255.255.0*.

In order to simplify the address configuration from an user perspective, the addresses can also be written in a short format omitting leading or trailing zeros. The prefix notation syntax is similar to IPv4 CIDR (prefix / prefix-length). Some IPv6 addresses examples are provided in table 2.1.

Table 2.1: IPv6 text representation

Form	Example
IPv6 address	2012:abcd:1234:4321:0000:0000:0000:0001
IPv6 Short-form	2012:abcd:1234:4321::1
Prefix Notation	2012:abcd:1234:4321::1/64

Notice that a single IPv6 address assignment, with at least 64 bits as prefix, has more host possibilities than the entire IPv4 address space. Its prefix information[1] is used to group IPv6 address types as show in table 2.2. An IPv6 address type can represent a group of addresses or a single address. For example the loopback type represents a single address, the loopback address *::1*.

Table 2.2: IPv6 addressed types based on the prefix, adapted from [1]

	Notation	
Address type	Binary	Prefix
Unspecified	00...0 (128 bits)	::/128
Loopback	00...1 (128 bits)	::1/128
Multicast	11111111	FF00::/8
Link-Local Unicast	1111111010	FE80::/10
Global Unicast	others	

[1]The prefix information are the left-most n bits of the IPv6 address, where n is the prefix length.

Address scope

An address scope defines the span where the address can identify uniquely an interface [4]. The scope zone represents the area of a group of nodes using the same address scope [4]. The following list explains the unicast address scopes. Those concepts are important to understand each IPv6 type. Multicast address scopes are out of scope of this thesis.

Interface-local The span of the packets is limited to the same node/interface of the origin (e.g. communications between processes using a socket with the same IP address).

Link-local Equivalent to IPv4 broadcast domains, packets using these addresses are not forwarded by routers. Nevertheless, they can be used to create a fully functional network within that scope.

Global The most wide scope used to identify an interface over several networks (e.g. Internet).

Unicast

In IPv6, different addresses sub-types of unicast were introduced as characterized below.

Link-local It is composed of the well-known prefix, *FE80::/10* plus a 64 bits interface ID in the 64 right-most bits. The remaining bits of the prefix are filled with zeros.

The interface ID identifies a specific network card inside a network, this means that an interface ID is unique per network prefix. Although it is possible that the same interface ID identifies different network cards in different networks, this is not likely, since in most cases this value is generated randomly, or based on a constant link-layer address.

The IPv6 link-local addresses can only be used with on the same a link. These addresses have an important role in several protocols such as Neighbor Discovery (ND). Before being generated, the global address, are used to exchange messages between nodes on

the same link. They can also be used to deploy an internal and non routable network.

Global It is a routable address built locally, in the node, using an ISP assigned prefix plus the Interface ID. Those addresses are globally unique and therefore, used for internetwork communication (e.g. between two or more IPv6 networks). Each interface can have more than one global address.

Site-local A routable address type restricted to an organization network scope, already deprecated [5].

Multicast

IPv6 multicast addresses are used in one-to-many communication playing an important role in several IP related protocols as a replacement of broadcast. Table 2.3 illustrates some of the most used and pre-defined multicast addresses.

If multicast needs to be supported by a v4-to-v6 converter box, like mShield, this device should be aware of pre-defined multicast groups and the incompatibilities with the previous IP version.

Table 2.3: Most common and pre-defined IPv6 addresses

IPv6 Address	Description
FF02::1	All Nodes address (link-local scope)
FF02::2	All Routers address (link-local scope)
FF01::1	All Nodes Address (interface-local scope)
FF01::2	All Routers Address (interface-local scope)

2.1.3 Header Format and Extensions

The IPv4 header carries all IP related information and has a variable size, whereas the IPv6 header has a fixed size and less fields. Figure 2.1 shows the IPv6 header next to the IPv4 header for easy comparison. In most cases not all IP options are used, so shifting them to *Extension Headers* improves packet processing.

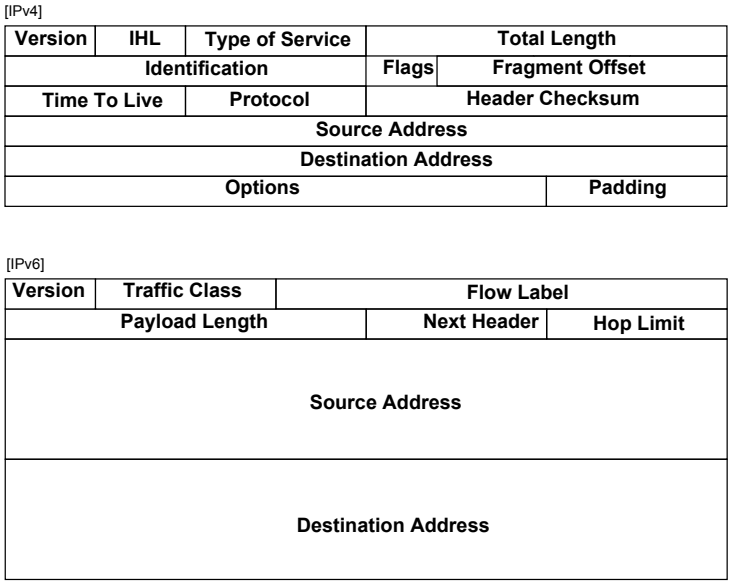

Figure 2.1: IPv4 versus IPv6 Header Format

The field *Next Header* is used to identify the next header, which could be another extension header or a protocol like, Internet Control Message Protocol version 6 (ICMPv6). It is possible to have a sequence of different extension headers, each pointing to the next one. Routers must process the headers in the same order as they appear [3]. The IPv6 specification defines that IPv6 implementations should have support for Authentication Header (AH) and Encapsulated Security Payload (ESP) used to provide secure communications at network layer [3]. Nevertheless, their use is not mandatory.

Both, v4- and v6- headers share the same base fields, an useful feature during the conversion between them[2].

In IPv6, the *Quality of Service (QoS)* feature can be provided using the field *Traffic Class*, similar to the field *Type of Service* in IPv4 [7]. Additionally, in IPv6 exists the *Flow Label* field. This field can be used to identify the flow of packets, corresponding for instance to a VoIP session, which provides a more efficient routing process. Due the differences between the two IP versions since this is a new field without correspondence in IPv4, this feature can be lost (e.g. during a v6- to v4- header conversion [6]).

[2]The IP/ICMP Translation Algorithm [6] defines how v4- and v6- fields should be mapped to the other version (e.g. *Time To Live* in IPv4 to *Hop Limit* in IPv6)

An uniform format for the IPv6 extensions header [8], showed in figure 2.2, was defined as an update to the IPv6 specification. According to that specification, a node can skip the processing of an unknown header.

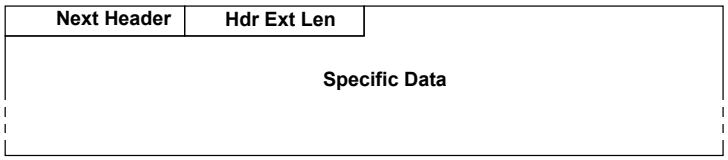

Figure 2.2: IPv6 Extension Header Uniform Format

2.2 Core Protocols

In this section, several protocols like ICMPv6 or Stateless Address Autoconfiguration (SLAAC) are presented. They are used to support IPv6 in functions like neighbor communication or address configuration. Their understanding is necessary for the security analysis and the design proposal (e.g. how the ICMPv6 messages can be used to trigger local network attacks).

2.2.1 ICMPv6

The successor to Internet Control Message Protocol version 4 (ICMPv4) is called ICMPv6 [9]. It defines a new set of control messages for informational and error purposes. The ICMPv6 header is added immediately after the IPv6 header, as outlined in figure 2.3.

Most of the ICMPv6 messages are derived from ICMPv4, like echo request/reply or time exceed. Nevertheless, new ICMPv6 messages were defined to cover new functions, like neighbor discovery and stateless autoconfiguration. It is also possible to define additional messages following a general message format as described in figure 2.4.

2.2.2 Neighbor Discovery

Neighbor Discovery [10] can be viewed as a mixture between Address Resolution Protocol (ARP) and ICMPv4. Technically it defines an additional set of ICMPv6 messages (e.g. *router advertisement / solicitation*). Using

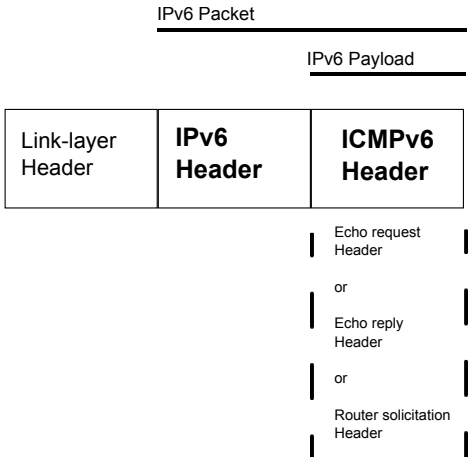

Figure 2.3: ICMPv6 Header in TCP / IP Architecture

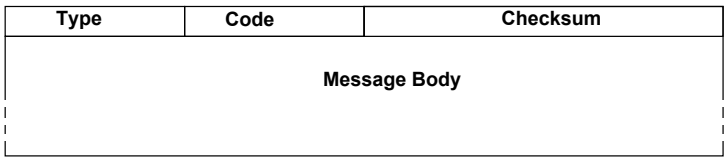

Figure 2.4: ICMPv6 Header Format

those messages, the nodes can interact with each other at the same link. The following topics describe some of the problems covered by ND:

- ND can be used to discover a list of routers, the network prefix or other parameters.
- Stateless Address Autoconfiguration allows to configure IPv6 addresses at network layer.
- Additional functions like Duplicate Address and Neighbor Unreachable Detection.

The following ICMPv6 messages were defined:

- *Router Solicitation / Advertisement* They are used to discover a router and learn IP settings (e.g. network prefix).
- *Host Solicitation / Advertisement* They are used to discover hosts on the same link. It is useful for instance to avoid duplicate IPv6 addresses.

Those messages, triggered by the ND algorithms, are sent as replies to nodes, or sent in an unsolicited way to rapidly propagate changes in the network.

Usually, in unsolicited messages a reserved multicast address (*all-routers* or *all-nodes*) is used as a destination IP address.

Stateless Auto-configuration

SLAAC [11] is used to generate link-local addresses and global addresses. The IPv6 addresses are composed of a prefix advertised by the routers and an interface ID generated based on the link-layer address. SLAAC eliminates the need of a Dynamic Host Configuration Protocol version 6 (DHCPv6) [12], also called stateful address autoconfiguration.

Stateless and stateful modes have the same goal, IP settings configuration. Nevertheless, they are defined at different layers of TCP/IP. Stateless mode is used at network layer whereas DHCPv6 runs at application layer.

The main disadvantage of the stateful mode is the need of a dedicated Dynamic Host Configuration Protocol (DHCP) server. Nevertheless, in a dual stack environment in order to support dynamic IPv4 and IPv6 address configuration it is necessary to have a DHCP server, at least for an IPv4 address configuration.

Stateless option is an advantage for a set of devices on the same link since they can use only link-local addresses to communicate between each other without the need of a router or even a DHCP server (e.g. a group of sensors or an ad-hoc network). Nevertheless, the SLAAC definition [11] limits the parameters advertised by the router and does not cover the use of Domain Name System (DNS) options[3]. Additionally, routers do not keep entries of hosts in the network in stateless mode.

SLAAC begins generating an IPv6 link-local address locally. In the next step the node sends a message to that IPv6 address and waits for a reply to check whether that IPv6 address is already used (Duplicate Address Detection (DAD)). If successful, the assigned link-local address is used in the next steps[4]. A Router Solicitation (RS) message is sent to *all-routers multicast IPv6 address* and a Router Advertisement (RA) message containing the prefix information is expected. Finally, the host can use the network prefix together

[3]IPv6 router advertisement options [13] can be used to configure DNS settings.
[4]The next steps are just performed by hosts

with the Interface ID to construct a global and routable IPv6 address.

Generating the interface ID from the Media Access Control (MAC) address raises security concerns since the value is predictable allowing host scanning in a remote network. The first 24 bits of Interface ID identifies the network card vendor that could be guessable and the next 16 bits are fixed reducing the scanning space to 24 bits [14]. Since the interface ID remains constant, this allows also for host-tracking in different networks. One possible solution is the use of SLAAC privacy extensions [15], generating temporary and random values for the Interface ID. However using temporary values makes it hard to manage IP assignments within a network. A proposal solution [16] defines a way of random generating the Interface ID value based on several inputs, including a private key and network prefix. In this approach the value remains constant within a network since all inputs remain constant, making it easy to manage and, at the same time, difficult to scan since it is randomly generated having a private as input.

SEND

A proposed standard, SEcure Neighbor Discovery (SEND) [17] defines a secure ND against attacks that include Denial of Service (DoS) and address spoofing [18]. The proposed solution involves:

- Certification Paths, a mechanism to establish trusted anchors to the authority of a router.
- Cryptographically Generated addresses (CGAs) [19] using a public-key scheme.
- Signature option in ND messages to guarantee the integrity and authenticity of the messages.

2.3 NAT

One of the proposed solutions in IPv4 to minimize the limitation of the amount of available addresses is called Network Address Translation (NAT) [20]. This technique is used to translate the source IP address of a packet. Commonly,

it uses an association of a private IP address and a port mapped to a global IP address, so multiple internal IP addresses can be translated to one single external IP address. Other terms such as NAT44 or NAPT44 are often used to describe the same technique.

In IPv6 the availability of free addresses is not a problem anymore. Nevertheless, several other reasons, such as hiding internal devices in a network or renumbering purposes, can justify the use of NAT with IPv6. NAT breaks the end-to-end connectivity model and, without additional strategies, it will break several protocols. A discussion about the use of NAT in the design is presented in chapter 4. The security implications are discussed in chapter 3.

An experimental NAT66 document [21] defines a stateless mechanism to translate between IPv6 prefixes. There are no more RFCs referring to NAT in IPv6. Even that document does not define how to translate from multiple IPv6 addresses to a single IPv6 address.

Nevertheless, several implementations already support stateful NAT in IPv6 in the same way as used in IPv4, see appendix B. In the remainder of the document, the terms NAT44 and NAT66 are used to refer to a stateful NAT in IPv4 and IPv6 respectively. The use of NAT in transition scenarios is described in the following sections.

2.4 Transition Mechanisms

The term *Transition* adopted in IETF documents refers to the mechanisms used to migrate from IPv4 to IPv6. Although another term could be used, *Coexistence*, since those mechanisms describe scenarios of coexistence between the two IP versions, it is commonly accepted that *Transition* better describes the goal of those mechanisms [22]. On the other hand, the term *Coexistence* is associated with dual stack technique. The term *Conversion* refers to the translation technique used to convert between two IP versions.

2.4.1 Mechanisms

Generally, the transition mechanisms are grouped into three categories: dual-stack, tunneling and translation. In the following subsections, an overview of

each one is presented.

Dual Stack

Dual stack means full support for both IP architectures, IPv4 and IPv6 in the same host. This is the recommended approach for most scenarios [23]. However, this is not always possible (e.g. on legacy devices).

Tunneling

This technique refers to the encapsulation of IPv6 packets inside IPv4 packets. A tunnel between two end-points is established where an IPv4 header is added and extracted. The intermediate devices between the end-points of the tunnel do not need to know IPv6. One example of this implementation is Teredo [24].

Translation

Translation means convert IP packets to the other IP version. This translation occurs in IP and Internet Control Message Protocol (ICMP) headers, as well as in the transport header in some modes as described in the next topics. It allows IPv4-only devices to communicate with IPv6-only devices.

2.4.2 Translators in detail

A dual stack approach implies changes in all devices to use both IP versions, and that change could not be desirable (e.g. legacy medical devices). The tunnels approach does not solve the problem of enabling communication between two IP versions since it is used to enable communication between end nodes using the same IP version. Others solutions like A+P [25], NAT444 [26] or DS-Lite [27] aim to solve the problem of scarcity of IPv4 addresses. However, they do not enable the communication between both versions. In this thesis, the only approach considered regards translators.

One of the first attempts by IETF to standardize the translation process is Stateless IP/ICMP Translation Algorithm (SIIT) [28] and NAT-PT [29]. In

SIIT, a stateless mode of translating IP and ICMP headers is defined, whereas NAT-PT defines the routing process between both IP versions. Nevertheless, due to technical reasons [30], NAT-PT was obsoleted and the SIIT algorithm was updated by a new IP/ICMP translation algorithm [6] as part of a new translation framework, NAT64 [22].

A stateless translation approach called IVI Translation [31] deployed in China Education and Research Network (CERNET) is presented as an informational RFC. In this approach, the SIIT algorithm is used with a limited subset of IPv6 addresses and with a fixed prefix. Those constraints allow the support of one-to-one address translation in both directions providing scalability. The differences between stateless and stateful modes can be found in the next topics and appendix A.

To the best of author's knowledge, no other relevant studies refer to practical deployments of the translation approach and the use of NAT64 framework either using the stateless or stateful mode. Nevertheless, some recent studies provide feasibility analyses referring to the behavior of the translators with the most common upper-layers protocols [32] [33]. No other studies were found mentioning security implications resulting from a practical deployment of NAT64. Nevertheless, several documents can be found mentioning theoretical security considerations regarding the transition mechanisms. These security considerations are covered by chapter 3. Looking at recent implementations, it is also expected that in the near future new studies will appear mentioning analysis vectors like security or performance.

NAT64 Framework

The NAT64 framework [22] includes the following components:

- IPv6 / IPv4 Address translation [34]
- IP and ICMP translation [6]
- Support for stateful mode [35]
- Support for DNS64 [36]
- Application Layer Gateway (ALG), additional definitions to support incompatible protocols due to its constrains (e.g. ftp active mode)

IPv6 / IPv4 Addressing Translation

A proposed standard [34] defines an algorithm to automatically compute an IPv6 address from an IPv4 address plus a well-defined prefix[5] (or a manually defined one). The same definition also specifies the reverse algorithm to extract the IPv4 from an IPv6 address.

According to the terminology in the specification:

IPv4-translated IPv6 addresses Subset of IPv6 addresses used by IPv6 nodes that can be automatically translated to IPv4, used in stateless translation.

IPv4-converted IPv6 addresses IPv6 addresses representing the IPv4 addresses from IPv6 network perspective.

The IPv4 embedded IPv6 addresses (IPv4-converted IPv6 addresses and IPv4-translated IPv6 addresses) described in figure 2.5 contain the following elements:

Prefix Well-known prefix or statically assigned. The definition restricts the prefix length to 32, 40, 48, 56, 64 or 96 bits. The well-known prefix cannot be used with private IPv4 addresses.

Null Octet A null octet for prefixes smaller than 96 bits is included between bits 64 and 71. It is represented in the figure 2.5 by the letter *u*.

Suffix An all-zero suffix at the end till bit 128.

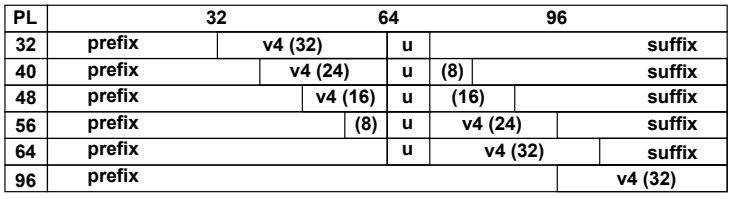

Figure 2.5: IPv4 embedded in IPv6 addresses format

[5]The well-known prefix used by the mapping algorithm is *64:ff9b::/96*

IP/ICMP Translation

IP / ICMP Translation Algorithm [6] defines how a specific IP or ICMP header can be translated to the other version. The document specifies the fields correspondence and defines what should be done in case of incompatibilities.

New IPv6 features are discarded during the translation (e.g. *Flow Label* field). The IP addresses are translated using *address translation algorithm* [34].

Some of the ICMP message types do not have correspondence in the other version. For instance, *information request/reply* messages in IPv4 are obsoleted in IPv6 and should be dropped out during the translation process. For those ICMP messages having correspondence in the other version, the fields are updated [6].

The IP/ICMP translation algorithm also specifies that the converter device should be capable of distinguishing when a packet should be translated or not. This is important and makes it possible to have both IP versions at the same side of the translator device which only performs the translation when needed.

Operation Modes

The major problem of NAT64 is how to represent any IPv6 address with a single IPv4, due to the difference in their lengths. This problem is visible in IPv4-initiated connections where an unknown IPv6 needs to be represented by an IPv4 address. In IPv6-initiated connections, the problem can easily be solved by representing each IPv4 address by its own value plus some prefix or using another appropriate scheme. Nevertheless, this process consumes one IPv4 address per each IPv6 and it is not suitable to solve the lack of IPv4 addresses.

The framework also defines two modes of operation:

Stateless The same algorithm is used to translate the source and destination IP addresses in both directions (IPv6 to IPv4 and vice versa). The stateless mode is limited to IPv4-translatable IPv6 addresses. In the stateless mode, the change only occurs in the IP header and it is not expected to interfere with the upper layers.

Stateful A state table is maintained and used to translate the destination IP address. The state mantains the relation between the destination IP address and port to the translated IP address and port. After the translation, the packet is sent with the source IP address of the converter interface (i.e. IPv6 or IPv4 depending on the direction).

The stateful mode allows the use of any IPv6 address without any manual configuration, but also cannot solve the problem of IPv4-initiated connections. Nevertheless, in this approach each address translation can be manually specified. Restricting the IPv6 address space to a subset of IPv4-translatable IPv6 addresses simplifies the configuration since it can be based just on a prefix.

In this mode, the transport header is modified, for instance by changing the source address port. Other upper-layer protocols that include IP addresses in their own headers such DNS, need additional considerations and the use of an ALG. Moreover, the current specification [35] only supports ICMP, Transmission Control Protocol (TCP) and User Datagram Protocol (UDP).

The Appendix A presents an overview of the translation steps (with emphasis on addressing) and compares stateless and stateful modes.

DNS64

One of the functions of DNS is a mapping between IP addresses and domain names using DNS records. The *A* records map an IPv4 address to a domain name whereas the *AAAA* records are used to map an IPv6 address to a domain name. DNS64 [36] permits the translation between an *A* record to an *AAAA* record.

Whenever an IPv6-only device needs an *AAAA* record, it sends a DNS request to the DNS64 device, that can be implemented or not in the same device of the NAT64 translator. If the DNS64 device does not know the *AAAA* record requested, it makes a new request to a DNS server for an *A* record using the same domain requested. If the DNS64 gets the answer, it synthesizes an *AAAA* record using the IPv4 address mapped in the *A* record plus a

configured prefix and forwards it to the IPv6- only device.

In theory DNS46 should perform the reverse process, by synthesizing the A records from $AAAA$ records and returning the A records to IPv4 hosts. Currently, no valid drafts exist covering this process.

2.5 Security Theory

In this section, security concepts like malicious code or DoS attacks are explained. Those concepts are refered in the security analysis. Therefore, their understanding is necessary (e.g. the impact of a DoS attack in the network communication).

2.5.1 Security properties

The STRIDE model [37], used in security analysis, defines a threat classification (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege) according to the effect on the following security properties:

Authentication Provide evidence of identity. Consists of a method where the identity is compared against some known and trusted values.

Integrity The term is often related with ensuring that specific data is not manipulated during a transfer. It can also mean that a system is free of malicious code.

Non-repudiation Guarantees that an action cannot be denied later by its actor.

Confidentiality Ensures that the information remains private. This means the use of cryptography to encrypt the data.

Availability Refers to whether a resource can be accessed. A resource can be a service running on a device or the entire network.

Authorization Narrows the access to a resource based on privileges.

2.5.2 Malicious code

Malicious code defines the group of all types of programs coded to cause damage on a system or device. The term *compromised* is used to define a device that contains any type of malicious code. There are several types of malicious code such as virus, worms or trojans. Despite of their individual definitions, not covered by this thesis, they share some common properties:

- Reproduce itself via network by exploiting vulnerabilities or by human intervention
- Destroy, tamper or steal data by sending it to a remote machine
- Used to trigger attacks (e.g. Distributed Denial of Service (DDoS) attacks)

Figure 2.6 gives a generic overview of the steps needed to compromise a system over a network and steal confidential informations using a worm. A typical scenario involves the following steps:

Reconnaissance A pre-defined range or a random number of IP addresses is used to discover potential victims. Several discovery techniques are applied to gather information like open ports and information about the services behind these ports. This helps to find potential targets that have vulnerabilities.

Exploiting a vulnerability The next step is to explore the vulnerability. The target can be any service reachable by network or even the protocol used during the communications, like IPv6. Figure 2.6 shows a simplified buffer overflow, where an arbitrary code can be executed as a result of lack of input validation.

Remote code Execution During this step, generally two actions are implemented. Repeat the cycle so another device can be compromised and, optionally, send any kind of data back to the original attacker. Other arbitrary actions are also possible.

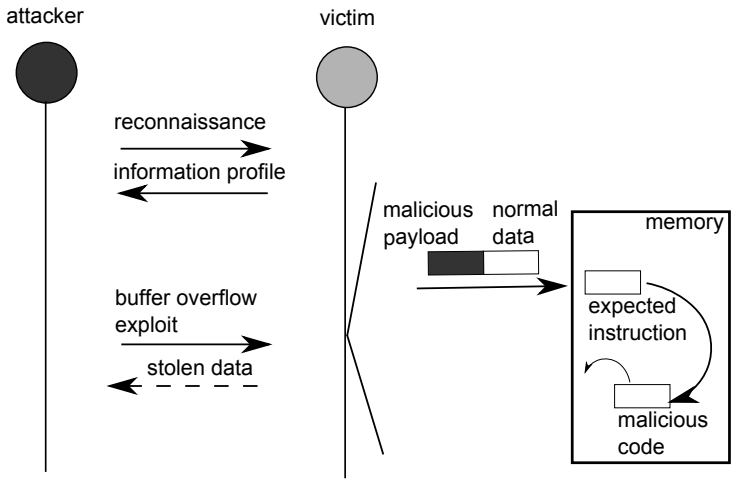

Figure 2.6: Steps of a buffer overflow attack

2.5.3 Spoofing

IP Spoofing

The basic function of IP spoofing is to manipulate a packet header in order to change the IP source address, either IPv4 or IPv6. By applying this technique, an attacker can send a request from his machine to an intermediate device and expect that a reply will be sent to the victim machine, turning the intermediary into a reflector. This technique can be used in DoS attacks as described in the next topics. Another common usage is to bypass a firewall that implements filtering based on a source IP address.

Neighbor Spoofing

For unicast communications, whenever a host needs to sent a packet to another host, besides the IP values it also needs to know the MAC address to send the packet. In IPv4, the mapping of MAC to IP is usually done dynamically using ARP. In IPv6, ARP is not used anymore but the need of mapping IPv6 to MAC addresses remains. That function is performed using Neighbor Solicitation (NS) and Neighbor Advertisement (NA). An attacker can send unsolicited NA messages claiming a spoofed IP address causing the wrong association between the IP and MAC address on the victim.

2.5.4 DoS

The term *DoS* is associated with a class of attacks aiming at the availability of a service, a host or an entire network by consuming the available resources. An elevated number of packets per second may cause a network congestion or memory exhaustion and prevent others services of using those resources.

Although these types of attacks do not cause a direct damage on systems, they have a strong impact on companies, since presently a significant number of resources are network dependent. In chapter 3, the impact of those attacks on mShield and its environment is discussed.

There are two classes of DoS: massive packet flooding or special forged packets in order to explore protocol issues or poor implementations. In the case of massive flooding attacks, one of the key aspects is the factor of amplification. A remote controlled structure of compromised devices, zombies, or a group of trusted servers, reflectors, are often used in order to amplify the attack. The reflectors are regular servers that reply to requests from a forged source IP address. This way, the replies are sent to the victims instead of the real source of the request. Different terms are applied according to the use of distributed compromised devices, DDoS, the used of reflective hosts, Reflective Denial of Service (RDoS), or combining both, Distributed Reflective Denial of Service (DRDoS) as outlined in figure 2.7. The compromised devices can also be organized in a multi-layer structure in order to amplify the attack factor and difficult the traceability of the attacker. The attacks may explore one or more protocols such as ICMP, TCP or UDP.

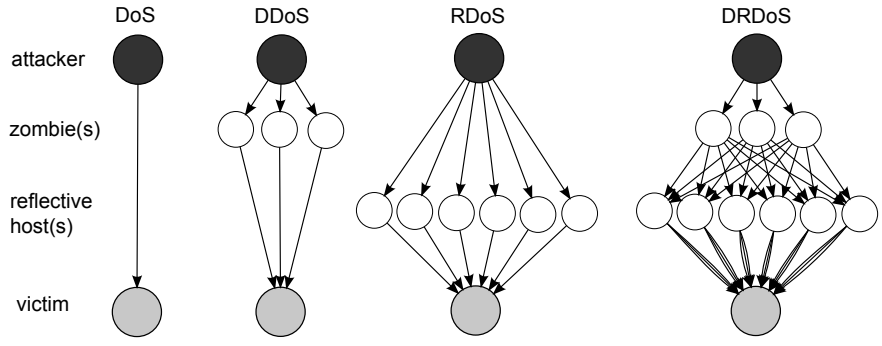

Figure 2.7: Behavior of different types of DoS attacks

DoS attacks rely on the use of forged source IPs, taking advantage of incorrect

filtering, and a prior spread of malicious code to create a "network" of compromised devices. An example of these attacks is shown in figure 2.8.

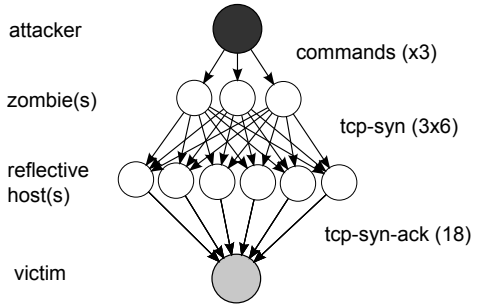

Figure 2.8: DRDoS attack using invalid TCP connections

2.5.5 Packet Filtering

Ingress filtering One of the security mechanisms against IP spoofing based DoS attacks is ingress filtering [38]. As a current best practice all packets coming from an unallocated space or any specially reserved IP should be blocked. Those IP addresses are usually called *bogons*[6].

Egress filtering The egress filtering mechanism is used to filter spoofed packets from the inside.

These filtering techniques are specially useful if applied by an Internet Service Provider (ISP) in order to prevent DDoS attacks. However, since some ISP do not provide those types of filtering some of the techniques employed in those attacks are still valid.

[6]Team Cymru provides an updated list of bogons IP addresses including IPv6 [39]

Make your life easy.

Holger Mikolon

3 Security Analysis

3.1 Methodology

The methodology used includes a set of steps as outlined in figure 3.1. It is composed of two parts: a theoretical and a practical part. The security analysis of making mShield an IPv4 to IPv6 converter box is provided in the theoretical part, whereas in the practical part, a design proposal and its implementation is presented. Additionally, it also includes a set of practical tests and a discussion of their results. The current mShield design does not have any IPv6 related support. Therefore, it is useful to understand how mShield can be redesigned to enable IPv6 support.

In the proposed design, all the design changes from current design are discussed. The practical tests support the theoretical security discussion and the proposed design. The terminology and a detailed description of each step are provided in the next topics.

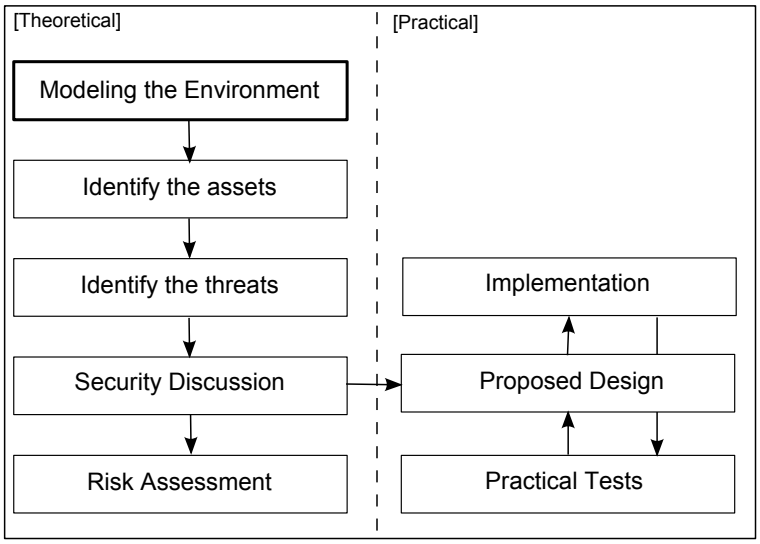

Figure 3.1: Steps of the Methodology

3.1.1 Terminology

Asset An asset represents a process, data or physical resource that belongs to the information system.

Threat A threat can be defined as the damage that an action can

cause to assets. The damage results from an action of someone or something in the presence of a vulnerability. Even if a vulnerability is fixed, the threat persists and can be triggered in the presence of other vulnerability.

Vulnerability Vulnerability refers to a specific known weakness of a system. Can be referred to hardware or software weaknesses. The scope of this document is related to software.

Risk Risk is commonly expressed as the value of several inputs like the impact of a specific threat combined with the likelihood.

3.1.2 Modeling the Environment

This step involves gathering information about mShield, costumer network and Philips medical devices in order to understand the system. The main characteristics and requirements of those elements are described in section 3.2. A full description of the translation process and implementation details is presented in appendix B.

3.1.3 Identify the assets

At this step, a discussion about each asset is presented. The resulting list of assets will be used to match against the threats. Each asset is expressed in together with its importance, expressed by the terms *high*, *medium* or *low*. The definitions used to evaluate each asset are shown in table 3.1 whereas the assets are discussed in section 3.

Table 3.1: Assets Definition

Classification	Definition
High	It is mandatory to protect the asset.
Medium	It is not mandatory but the asset should be protected.
Low	It is not mandatory but it could be desirable to protect the asset.

3.1.4 Identify the threats

In this step a list of network threats are discussed. Each threat is grouped according the STRIDE model [37]: Spoofing identity, Tampering with data, Repudiation, Information disclosure, Denial of service, Elevation of privilege. The threats discussion is presented in section 3.

3.1.5 Security Discussion

In this step, a security analysis about the differences of both IP versions is provided. It includes a discussion about the needed changes to the design such the translation approach or the security implications of the IP address allocation method. The security discussion is presented in section 3.

3.1.6 Risk Assessment

In this step, a risk evaluation based on Risk Management Guides for Information technology Systems [40] is performed. It is important not only discuss the effects but also measure them. The risk assessment allows to quantify the risk associated with each threat and helps to decide the next steps (accept the risk or mitigate it). Two inputs are used in the risk evaluation: *impact* and *likelihood* [40].

The *impact* measures the consequences of a materialized threat whereas the *likelihood* measures the probability of the threat being realized, considering the existence of known vulnerabilities, design flaws or the complexity / expertise required. The impact discussion and the risk assessment is provided in section 3.

3.1.7 Proposal Design

In this step, a new mShield design is presented in chapter 4. All changes are discussed and explained including the role of mShield in the translation process. Those changes are based on the security analysis.

3.1.8 Implementation

In this step, the implementation of all design changes is provided. It contains all the steps, commands and scripts used to enable NAT64 translation on mShield. The implementation details are covered by the chapter 4. Nevertheless, its details are presented in appendix B.

3.1.9 Practical tests

Several types of tests are part of this step. They are intended to support the proposed design and the theoretical security discussion. The converter box requirements are tested and the packet translation debugged. Finally, a discussion about their results is provided. The practical tests and their discussion are presented in the chapter 4.

3.2 Environment

An IPv4 to IPv6 converter box represents a network device capable of translating communications initiated by IPv4 devices towards IPv6 devices. The basic requirement includes support for unicast and IP / ICMP packets translation. Additionally, IPv6 to IPv4 translation must be also supported. The mShield device plays the role of a converter box.

In this section, the entire environment surrounding mShield is described. It includes the medical devices and customer networks as well as the characteristics of the environment. Finally, the role of mShield in the translation process is also described.

All the scenarios involving IPv4 legacy devices and the need of translation between IP versions were considered as outlined in figures 3.2 to 3.5. All options are important to consider since they affect the security analysis. For example in figure 3.2, the only one with just one IP architecture in both sides, the threats of IPv6 extensions headers are not considered since they are not translated by IP / ICMP translation algorithm [6].

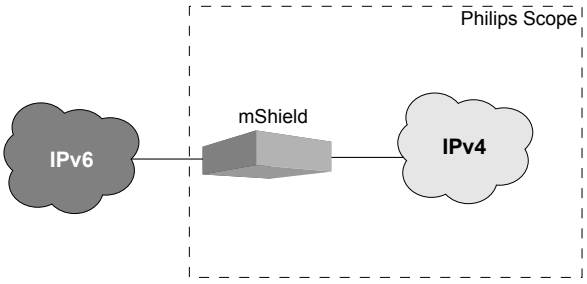

Figure 3.2: IPv6 Costumer Network and IPv4 Medical Devices

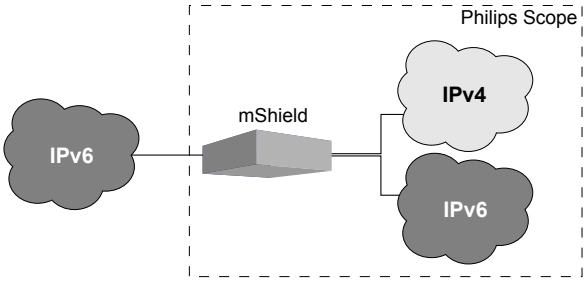

Figure 3.3: IPv6 Costumer Network and IPv4/IPv6 Medical Devices

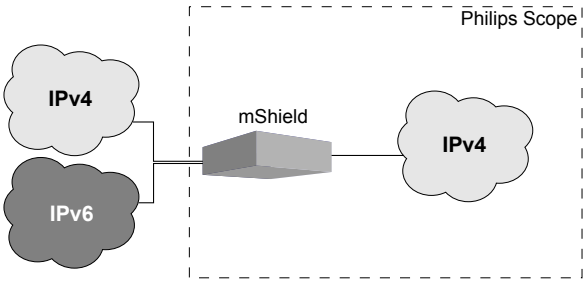

Figure 3.4: IPv6/IPv4 Costumer Network and IPv4 Medical Devices

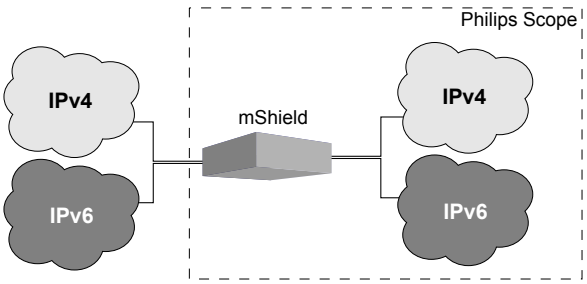

Figure 3.5: IPv6/IPv4 Costumer Network and IPv4/IPv6 Medical Devices

Features

- The converter box should support communications in both directions. It includes incoming connections, initiated by customers devices, and outgoing connections, initiated by medical devices.
- The converter box should be able to translate IP / ICMP headers in unicast communications.
- Generic application layer protocols using TCP or UDP at the transport layer should be supported.

Constraints

- IPv4 legacy devices should be supported without the need of adapting the devices already developed.
- The new design should not impose any constraints in customers networks. Those networks are not managed by Philips. Therefore, it should be possible to use any IP address in the customer side.

3.2.1 Elements

Medical Devices

Those devices represent the group of all medical devices deployed by Philips. The protocols used, run at the top of TCP or UDP (e.g. HyperText Transfer Protocol (HTTP) or Digital Imaging and Communications in Medicine (DICOM)). IPsec or multicast communications are not used. Therefore, it is not necessary the use of any ALG. Currently, medical devices uses only IPv4. Nevertheless, it is expected that new devices already support IPv6.

Customer Network

Since those networks are not managed by Philips, they should be considered as insecure. Nevertheless, in the most of the cases they have a perimeter device to filter traffic from the Internet. That consideration is necessary to the security analysis. Customer networks can be IPv4, IPv6 or both. The

external interface of mShield, part of customer network, can be statically or dynamically assigned (e.g. using SLAAC). The security implications of using a dynamic assignment are discussed in section 3.

mShield

The mShield device represents the intermediate device in all communications between the customer and the medical devices network. Its first goal should be to ensure the security of the medical devices network. The security should be enforced before the efficiency. The conversion function is a requirement caused by the existence of legacy devices. Nevertheless, in the future, the communications are expected to be IPv6 to IPv6. Those concepts are important during the design and implementation of a new mShield release as a converter box.

In terms of software, mShield is a minimal modified version of OpenBSD. That choice can be justified with the main goal of security in mind. It is based on one of the most secure Operating System (OS) in the market. As a bonus, its reduced size make it an easily manageable software. The configuration can be done using a remote Secure Shell (SSH) connection or using serial connection via a configuration tool.

mShield is composed of 4 physical interfaces, one external interface connected to the costumer network, and the other internal interfaces, connected to medical devices. Additional functions such as IP translation or cryptographic operations could exceed the limited hardware capacity of mShield. Therefore, the hardware requirements should be reconsidered for the new design.

3.3 Security Discussion

In this section, a security analysis of the environment and its needed changes are presented. It is composed of the assets and threats identification, a discussion about their impact and a risk assessment. It is useful for the next phase in order to improve the security of the new mShield design.

3.3.1 Assets

In this subsection, the security related assets are identified and discussed. Their importance and the role of mShield are necessary in order to understand what kind of threats surround the environment. A detailed discussion about the threats and their impact is provided in the next sections and the identified assets are shown in table 3.2.

Availability

Medical devices like x-ray systems are used in a critical environment. Those devices can be used to save lives, and therefore it is necessary to assure its availability. By filtering the traffic from the outside mShield should assure that they can still be used in case of network availability attacks (e.g. DoS attack from the outside).

Due to the actual design, all communications are via mShield exposing it as a single disruption point. Nevertheless, during network unavailability, it is always possible to physically access and use a medical device.

On the other hand, a successful attack against medical devices availability will cause a serious impact. Protecting the systems availability is therefore classified as of high importance. It is acceptable that mShield becomes unavailable but that is not acceptable for a medical device.

Integrity

The patient data, used by medical devices, is critical and should not be tampered or destroyed by malicious code.

The mShield role is limited to the traffic filtering, which means it cannot detect malicious code, like an anti-virus software. Nevertheless, through traffic filtering, mShield can reduce the propagation and effect of malicious code.

The loss of integrity has also an effect on the systems availability. For instance, a virus detection will likely cause a downtime for a device or the entire medical

devices network. Since it is important to assure the systems availability it is also necessary to assure their integrity.

Table 3.2: Assets evaluation and description

ID	Security Property	Asset	Importance
1	Availability	Ensure availability of the systems by protecting the network against attacks.	High
2	Integrity	Reduce the effect and propagation of malicious code over the network	Medium

Further Considerations

Additional considerations are important to understand the scope of this analysis and the security evaluation. Nevertheless, they are not covered by the threats and risk discussion since they are not related with the converter box, IPv6 or part of current features of mShield.

Traffic filtering is based on IP addresses and ports, there is no traffic authentication. Notice that the purpose of an IP address is identify a network card and not provide evidence. As a consequence, mShield is vulnerable to spoofing attacks either in IPv4 or IPv6.

Traffic encryption, providing end-to-end confidentiality, is done using upper layer protocols such as SSH or HyperText Transfer Protocol Secure (HTTPS).

Authentication and authorization are also a security concern in the remote access to mShield (spoofing and elevation of privileges). This access, for configuration purposes, use SSH protocol, and again, it is not related with converter role.

For the same reasons end-to-end data integrity and non-repudiation were also not considered.

3.3.2 Threats

In this subsection, the threats surrounding the environment are identified. It is also explained their meaning and why they are a threat. The identified

threats are shown in table 3.3.

DoS

Two threats against the environment were considered: DoS attacks against mShield itself and DoS attacks against medical devices. The scenarios where an attack can affect medical devices but not mShield or the network communication are described in the next topics.

In both cases, the source of DoS attacks can be a device in the Internet or a compromised device in the customer network.

Those attacks are considered as threats since they affect the network availability or, in the worst case, affect the availability of a medical device.

Tampering

In this category tree types of threats were considered: Malicious code affecting either medical devices or mShield and its propagation over network.

For the network propagation scenario, the communications between customer and medical devices as well as communications between medical devices via mShield were studied. Although, medical devices are considered trusted devices, they can get compromised and spread malicious code over network.

Malicious code affecting medical devices can be used to tamper, steal or destroy patient data. Whereas malicious code affecting mShield can be used to change firewall configurations or intercept and steal patient data.

A compromised device, either a medical device or mShield can also be used to trigger other network attacks (e.g. DoS attacks).

3.3.3 Security Considerations

In this section, the security implications of the design are discussed. The use of IPv6 and the options for the new design are discussed. They are necessary to understand the impact and the likelihood of each threat presented during the risk assessment.

Table 3.3: Threats evaluation and description

ID	Category	Threat
1	DoS	DoS attacks against mShield
2	DoS	DoS attacks against medical devices
3	Tampering	Network propagation of malicious code
4	Tampering	Malicious code affecting medical devices
5	Tampering	Malicious code affecting mShield

Translation approach

The translation aproach, such as NAT64, does not raise additional security considerations rather than use the IPv6 and IPv4. Nevertheless, all headers and IP address translations need to be computed and therefore, can be used as a DoS target. Other security issues may result from the implementation used.

Address Assignment

The IPv6 address allocation method can increase the likelihood of network attacks. Dynamic assignment methods, such as SLAAC or DHCPv6, generally, produce more predictable addresses (e.g. MAC based or sequential ranges) [41]. Although, it is not expected that mShield is connected to the Internet without any filtering allowing remote scans. Nevertheless, even considering random IPv6 addresses, the local scanning is possible as discussed in the next subsections.

Routing and Switching

In the current design, the packet filtering is based on IP addresses and ports, mShield operates on data-link layer such as a switch. That means mShield is not visible in the network scans. Nevertheless, the IP / ICMP translation and packet forwarding, required for the new design, are done at the network layer. In that case, it may be desirable that mShield operate at the network layer, such as a router. The discussion about functional advantages and disadvantages are presented in chapter 4.

A router firewall means that it has assigned IP addresses on its interfaces and therefore it is also a target for network based attacks. In that case, the those attacks can be done not only against medical devices but also against mShield.

Using a router approach, from the outside perspective, medical devices are not visible. They are hidden behind external interface of mShield. Using Packet Filter (PF) it is possible to employ the same filtering policy.

On the other hand, medical devices are connected to internal mShield interfaces. In that case, mShield should behave as a switch, grouping those devices in a single network.

Communications between medical devices can occur without mShield filtering (e.g. external switch). In the case of internal communications using different IP versions, an external switch, without IP conversion is useless For the cases of a compromised medical devices, filtering internal connections can also prevent malicious code propagation via network.

Reconnaissance

Dual stack, specially in the external interface, increases the attack surface. Although, the necessary support for both IP versions in the customer side increases the number of effective malicious code and scanning methods. Excepting the common malicious code, the system is exposed to IPv4 plus the IPv6 threats. Several methods and tools have already been developed for perform IPv6 scans [42] [43] [44].

NAT in the communications between medical and customer devices difficults automated processes used by virus and worms to detect vulnerabilities and the OS fingerprint of medical systems. The same result can be achieved in the current design using traffic normalization.

In IPv6, even that the remote host scan can be difficult by using random IPv6 addresses, the local scan is simplified by the use of pre-defined multicast groups (e.g. ping to multicast link-local address of all nodes).

The services running on medical devices can be found by scanning all network range and ports either using a router or a switch approach. Using mShield as

a NAT router, the result is one host, the external interface of mShield, with all used open ports whereas in the current design the open ports are spread by several hosts. It is possible to find services either in IPv4 or IPv6.

On the other hand, the random network scans performed by virus or worms can be easily flagged since all communications are controlled and expected. This feature is not present in the current design. Several approaches have been developed to detect those scans in IPv6 [45] [46].

Vulnerabilities

Nevertheless, after a successful services recognition it is also necessary to have some vulnerability to explore. One of them, refers to the use of Routing Headers since this could be used to perform DoS attacks or bandwidth theft, particularly the type 0, already deprecated [47].

Other topics already observed in IPv4 like fragmented packet handling or stateful mechanisms are susceptible to cause DoS or used to bypass packet filtering protection. The extension headers mechanisms, used in IPv6, can also be used to bypass firewall protection (e.g. allow unknown extensions headers).

Nevertheless, there are no open IPv6 vulnerabilities affecting OpenBSD that can be used to include some malicious code in the system[1].

On the other hand, mShield is still vulnerable to neighbor spoofing and man in the middle attacks. Several IPv6 design problems were already demonstrated such neighbor spoofing [42]. ND messages can be spoofed and an attacker can therefore, intercept connections between mShield and customer devices. Despite of that, man in the middle attacks are also possible in IPv4 and in the actual design.

DoS Attacks

The DoS attacks exploring upper layer vulnerabilities are not covered by this document. Their operation mode does not change and can be easily adapted

[1] A Common Vulnerabilities and Exposures (CVE) is an open and widely used vulnerability database. It maintains a list of vulnerabilities and exposures affecting multiple OS and protocols. [48].

for IPv6.

In the case of a massive packet flooding from the outside, mShield will become unresponsive and break the network connection to medical devices. However, as outlined in figure 3.6, this consideration is not always true. A specific amount of packets per unit of time lesser than mShield maximum capability and greater than medical device capability will cause a DoS on medical device.

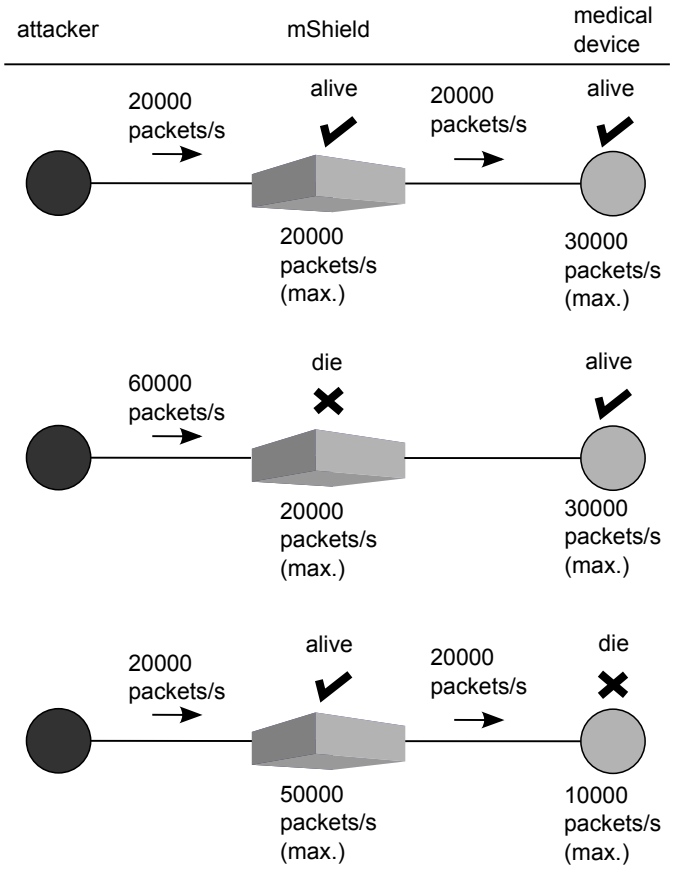

Figure 3.6: Effect of different amount of packets/s on the environment

Malicious Code

The mShield role against malicious code is limited to the network communications. It can block either in the first stage of infection or during the propagation.

Most of that code are designed to remotely control a devices or sending back

data from the compromised machines. In both cases, it is expected that mShield block most of those communications. Other code designed to local modify or destroy data or spread by removable devices, like usb sticks, are out of the scope of mShield.

3.3.4 Risk Evaluation

In this subsection, the damage and the likelihood of a threat are evaluated. For the likelihood values a qualitative analysis based on the current knowledge of security issues is presented. A discussion about the complexity and expertise needed to perform the attack is also included in the likelihood discussion. In the impact analysis, the network and business consequences are presented.

DoS Attacks

Impact A successful DoS attack against a medical device represent high impact. It have costs for customer and for Philips. The costumer needs to find or use alternatives such as other available equipment or transfer patients to another hospital. Whereas Philips needs to allocate resources to support the costumer to identify and mitigate the problem.

Nevertheless, a DoS attack against mShield will not have the same impact, since the medical devices can be physical accessed and used. It was classified as of low impact.

Likelihood DoS attacks using packet flooding are not complex. They take the advantage of a distributed network of compromised devices to flood some victim.

Other DoS attacks using special crafted packets can be more complex. Nevertheless, for some of them pre-built tools can be used without too much knowledge. For those attacks it is necessary to explore an OpenBSD vulnerability.

Despite that, DoS attacks directly against medical devices are only possible in some cases as described before. Even in those cases, specific packet rates need to be used. Attacks against mShield

are considered as more likely since the security level of customer network is unknown.

On the other hand, in both cases, if the attack comes from the Internet, the main source of massive flooding attacks, it is also necessary to bypass a border firewall in the customer side.

For DoS attacks against mShield, it was considered as having a medium likelihood of happening. Whereas the DoS attacks against medical devices were evaluated as having a low likelihood.

Malicious Code

Impact The loss of integrity affects the availability of medical devices. Those devices will likely be considered as unavailable until the problem is fixed. In those situations, the impact for customer and for Philips is the same as described above for unavailable devices.

On the other hand, a successful tampering of patient data have a significant damage for the customer. It can result in lawsuits against customer.

Considering both scenarios, the impact for damages caused by malicious code was classified as having a high impact.

Likelihood Due to its OS, malicious code affecting mShield, was evaluated as having a low likelihood. Few of vulnerabilities affected OpenBSD in the past.

Nevertheless, the case of malicious code affecting medical devices was classified as having a high likelihood. The sources of infection are more than network propagation. Considering all those scenarios and the high value, the problem of malicious code detection is out of the scope of mShield.

It is expected that mShield block most of the malicious code attempts to spread via network. Nevertheless, the allowed IP and ports can be used to explore a vulnerability on the victim.

Table 3.4: Risk classification

ID	Threat	Damage	Likelihood	Risk
1	DoS attacks against network	1	2	low
2	DoS attacks against medical devices	3	1	medium
3	Network propagation of malicious code	3	2	medium
4	Malicious code affecting medical devices	3	3	high
5	Malicious code affecting mShield	3	1	medium

3.3.5 Overview

The main security role of mShield is to protect medical devices against DoS attacks and malicious code propagation via network.

The use of an additional IP version has an impact in all threats as described in previous topics either from new IPv6 issues or due to the design changes. Nevertheless, most of security issues are present in the previous IP version.

Most of the IPv6 specific issues are local and can only affect communication between mShield and customer devices. That communication cannot be considered as trusted and should be encrypted whenever patient data is handled. This is valid either for IPv4 or IPv6.

The current design suffers from security issues such DoS attacks using specific packet rates). In the proposal design, chapter 4, it is discussed a solution for those issues.

Chapter 4 also refers to a practical set of tests in order to test behavior of the proposal design, discover IPv6 vulnerabilities and demonstrate some of the issues described above.

In the risk assessment, malicious code affecting medical devices was classified as having the highest risk. Although it has a high impact in the environment, mShield as a firewall cannot mitigate all the impact. Other measures, such as the use of an anti-virus, should be enforced in the medical devices. The current design was also considered effective against DoS attacks either by reducing its impact or the likelihood.

Never discourage anyone who continually makes progress, no matter how slow.

Plato

4

mShield64 - Design and Implementation

4.1 Background

In this chapter, a complete description of mShield64[1] is presented. It includes the design discussion and the implementation specification. Several functional tests and its results are also presented.

4.1.1 mShield

Several customers are moving to IPv6. Therefore, new mShield releases are required to have IPv6 support. In addition to that, legacy IPv4 medical devices already developed cannot be discarded or easily updated. Since those devices are required to talk each other, a IP converter box needs to translate there messages. Furthermore, the process should not interfere with the address allocation on customer side. An overview of the problem is outlined in figure 4.1.

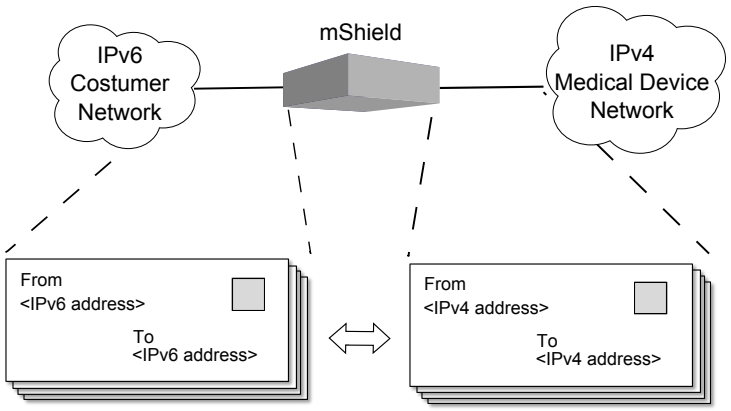

Figure 4.1: Overview of NAT64 applied to mShield

The current mShield device is a bridge firewall used to protect communications between medical and customer devices. It connects all the medical and customer devices in a single network. mShield uses OpenBSD as OS and PF as packet filtering. There is no need for additional IP configurations since mShield runs at link layer. Communications between medical devices are not filtered since it is assumed that are trusted.

[1]The mShield64 name, used during the thesis development, is composed of the *mShield* word plus the *64* number. Similar to NAT64, the *64* number means the IPv6 to IPv4 converter role added in the new design.

4.1.2 PF and NAT64

PF is the firewall system used in OpenBSD. It can be used to filter packets based on its values or other additional features like packet translation (e.g. NAT44) [49]. It is part of the OpenBSD base distribution, which mean it is maintained by the OpenBSD team, having the same development process and code quality of the OpenBSD itself.

The first open source implementation of NAT64 Framework [22], stateful NAT64 [35] and DNS64 [36] started with a project Ecdysis[2] [50]. In 2011, the project was committed to OpenBSD source tree.

In OpenBSD 5.1 release, PF starts to support NAT64 [35]. A new PF rule option, *af-to*, allows to specify the IP address translation, either based on address translation algorithm [34] or statically configured. It is possible to specify the destination address using an IP address or a network prefix [51]. The *af-to* option can also be used to filter packets based on IP address or ports.

PF behaves as stateful translator, this means it keeps a state of each initiated connection and uses it during the reverse translation (e.g. using the *af-to* option it is possible to traslate from IPv4 to IPv6 and then the other direction). The NAT44 and NAT66 features can be configured using the *nat-to* option.

4.2 Proposal Design

In the following sections, the design choices and their discussion are presented. They describe each needed change of the current design such as routing versus switching mode or address configuration.

4.2.1 Translation Approach

In addition to the IETF proposed standard, NAT64, it is possible to connect an IPv4 to IPv6 network using an application layer proxy. Figure 4.2 outline

[2]The first releases of the Ecdysis project were based on the NAT64 draft specifications. Nowadays, the specifications are classified as proposed standards.

their differences. NAT64 works at network and transport layers converting network and transport headers between two versions. Whereas the application proxy, such as squid [52], redirect application layer requests / replies. Using this kind of proxy requires dual stack support and is dependent of the protocols used. This means, the proxy must know all application layer protocols used in communications such HTTP or SSH. Whereas NAT64 is designed to support different types of upper-layers protocols. Nevertheless, if those protocols use IP literals they are incompatible with NAT64.

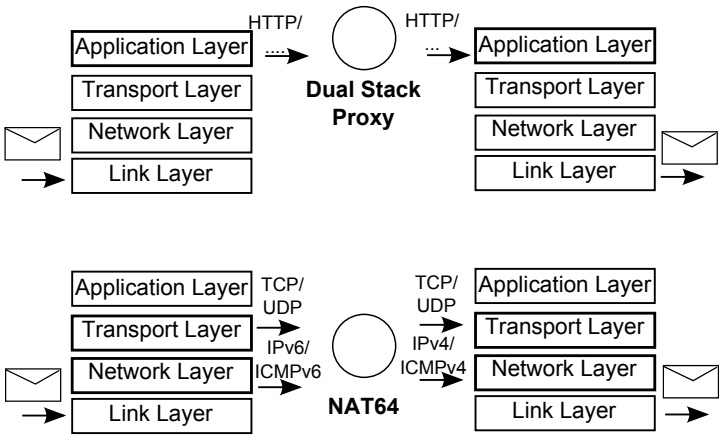

Figure 4.2: Application level Proxy versus NAT64 approach

The stateless translation mode is not considered for the new design. Despite the fact it offers a flexible and scalable design, the IPv6 addresses are limited to IPv4-translated IPv6 addresses. That condition is not acceptable since it will limit the IPv6 range used by the customer. NAT64 integrated with OpenBSD and PF rules provide a more flexible and robust solution, allowing the support for generic application layer protocols and an integrated filtering policy. Therefore, the author proposes the use of a stateful NAT64 approach [36] supported by PF rules.

4.2.2 Routing and switching

Update mShield to a router will split customer and medical devices in two separate networks. Although, it is possible to have NAT features in a switch, conceptually, they belong to the network layer. One of the reasons of using the switch approach, a mShield installation on-the-fly, without additional

IP settings such as a gateway or new addresses, is likely to change. It is necessary to configure the NAT associations either manually or implementing a dynamic scheme. On the other hand, medical devices need to be connected using a switch otherwise they will belong to separate networks. At the end, the author proposes to use a router approach between medical and customer devices. Whereas a switch approach to connect all medical devices.

4.2.3 Address Configuration

The internal interface of mShield, not visible to the customer, may has always the same value, a manually configured IP address. Whereas the external interface are dependent of the customer network. A manually and fixed value can simplify the management of the translation entries. Using a temporary IP address will require a dynamic update of PF rules. Nevertheless, the address assignment method does not interfere with translation scheme. Focusing on the NAT64 translation, in the remainder of this document, a manually and fixed values will be considered.

For the new design, the author proposes that all communications are initiatied to mShield interfaces. Then, mShield is responsible for redirecting the requests according the mapping entries. From the customer side, there is only one device to comunicate with, mShield. IP addresses of medical devices are not visable from the outside.

In addition to NAT64, it is also necessary the use of NAT44 and NAT66, both supported by PF. All different kinds of communications will behave in the same way, initiating the comunications to mShield. There is no diference from the outside of a NAT64 or NAT66 communication, it is started in both modes to the same IPv6 address. At the end, even that mShield behaves like a router[3], from an outside perspective it behaves as a host providing several services. Figure 4.3 outline the entire process, including the NAT and redirect roles. mShield receives the packets from one network segment, and redirects to the other segment the translated packet. The redirection and translation processes are performed by PF.

[3]The translator must behave as router [6].

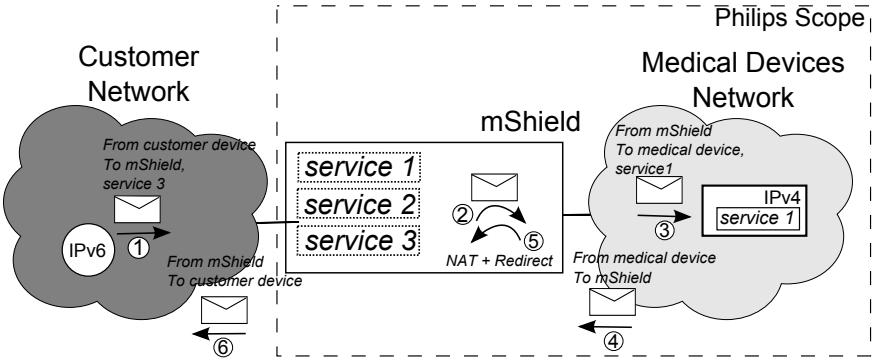

Figure 4.3: Packet translation and routing steps

4.2.4 Connection Rate Limits

As described in the security analysis, in some cases, a specific amount of packets per unit of time can cause a DoS on medical devices and not mShield. The author propose to limit the traffic rate according to each medical device capacity. It means a stateful filtering, where the connections are tracked and limited according pre-configured values. Nevertheless, traffic rate limits will not protect mShield itself against those attacks.

4.2.5 Configuration Tool

One of the significant changes from the current design is the need of additional configurations, namely NAT associations. In order to do that, they can be manually assigned using configuration tool. That tool can be used to create the address mapping entries by keeping a list of network devices and allowing a association between them. Then, it should translate those associations into rules syntax. This way, it provides an abstraction layer between the syntax of packet filter where an user is able to configure the translation by simply choosing two devices and a direction to communicate. Additionally features like backup / restore for the entire configuration are required.

Other approach, using dynamic configuration, involves exchange information between mShield and the other devices. Several protocols like Universal Plug and Play (UPnP) or NAT-PMP [53] can be used. It allows hosts to send messages to the NAT device in order to request an port / NAT association. In this case, not only mShield should support those protocols but also each device

who wants to initiate a communication. Moreover, since that scheme allows devices to control PF rules is not inherently secure. Host authentication is not part of those protocols. Another one, Port Control Protocol (PCP) [54] and its authentication mechanism [55], allow host authentication. Nevertheless, for the best of author knowledge there is no implementation available. For the sake of complexity, they will not be considered in the remainder of this document.

4.2.6 Summary

In the next release, the design of mShield should be updated to behave as a router. The packet filtering plus the address translation occurs between the external interface and the internal interfaces. The resulting internal network components used are outlined in figure 4.4 whereas the following topics summarize the proposed changes.

- Routing between internal and external interfaces and switching role limited to internal interfaces.
- Support for unicast communications using TCP and UDP for both IP versions.
- Support for ND and standard ICMPv6 messages [9]. It does not include SLAAC or other dynamic address configuration mechanism.
- Support for stateful NAT44, NAT66 and NAT64 on both directions. Mapping entries statically configured.

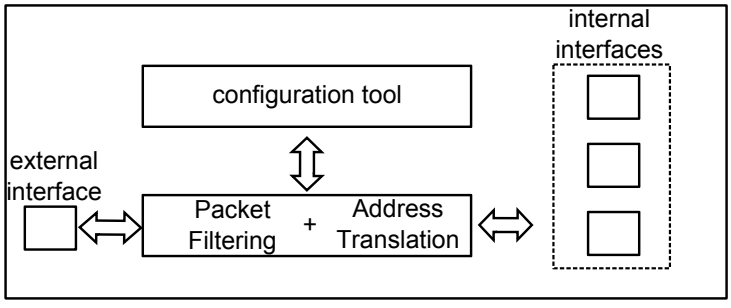

Figure 4.4: Internal mShield behavior

4.3 Implementation

All configuration description and details are provided in appendix B. Figure 4.5 shows the used scenario whereas the network settings are specified in table 4.1. It includes both IP versions in each side of mShield. Each possible communication, in each direction, is numbered in the same figure.

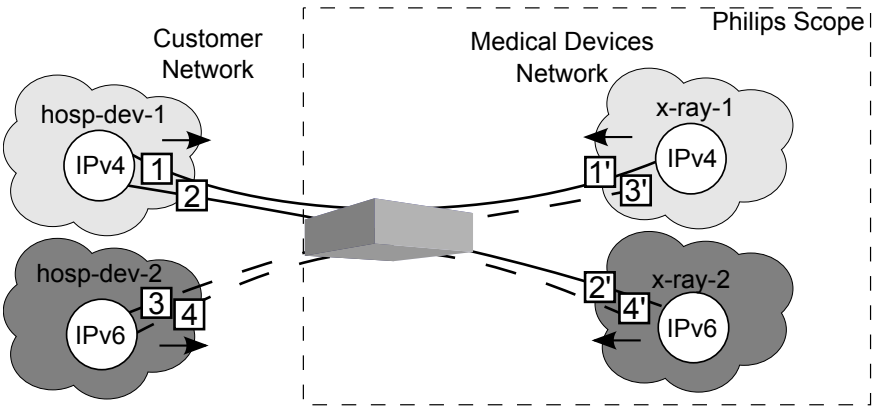

Figure 4.5: Connections between customer and medical devices

Table 4.1: Network Scenario Values

Device	Interface	IP address	Port(s)
hospital device 1	vic0	192.1.1.1/24	80
hospital device 2	vic0	2012::11/64	8080
x-ray device 1	vic0	192.2.2.2/24	90
x-ray device 2	vic0	2011::11/64	9090
mShield	vic0	192.1.1.64/24	90, 9090
mShield	vic0	2012::64/24	90, 9090
mShield	vic1	192.2.2.64/24	80, 8080
mShield	vic1	2011::64/64	80, 8080

4.4 Practical Tests

The folowing sections describe pratical scenarios where are tested the current support of PF for protocols like TCP or UDP as well as some IP related

features such as Path Maximum Transmission Unit Discovery (PMTUD) or fragmentation handling.

4.4.1 UDP and ICMP Translation

One of the requirements of mShield is to be able to handle UDP and ICMP. UDP is used for faster communications without assure reliable. Whereas the ICMP can be used for network debugging using error or informative messages. At this point, it is analysed how the converted box behaves and translate each packet, either UDP or ICMP.

In the first part, a UDP client, listing 4.1, sends to the server 5Mb of random data previously generated using the *dd* tool [56]. Whether a UDP server, listing 4.2, is responsible for creating a UDP socket to receive data. In both sides, the UDP sockets were managed by *Netcat (nc)* [57].

ICMP runs at network layer and does not use ports. Therefore, sending ICMP requests from the outside it is not possible. mShield would not know to where sent those packets. Nevertheless, according to the requirements, mShield should be able to translate ICMP messages. The ICMP translation is only possible using previous states, for instance as a result of a previous TCP or UDP translation.

In the second part to test ICMP translation, the UDP server socket was intentionally closed. Whenever the client starts to communicate, the server should reply with an ICMP *unreachable* message. Using the previous state, that message should be then translated by mShield back to the client .

```
#!/bin/sh
dd if=/dev/random bs=1M count=5 of=random.data
nc -u 192.2.2.64 80 < random.data
```

Listing 4.1: IPv6/UDP netcat client

```
#!/bin/sh
nc -ul 2011::11 80  > random.data
```

Listing 4.2: IPv4/UDP netcat server

4.4.2 TCP Connections

For testing TCP translation it was established an connection between two devices and check whether the transmited data was successful translated. The packets, handled by mShield, are converted between IP versions in both directions. At the end, the received by the server should be equal to the trasmmited by the client.

Both client and the server use the *nc* tool [57] to create the TCP sockets. The same tool is also used to transfer data between them by redirecting the input and output of the command. The client, listing 4.3, sends to the server 50Mb of random data previously generated. Additionally, the sha1 tool [58] is used to create a hash of that data. In the other side, the server, listing 4.4, receives data and, using the same hashing algorithm of client, also generates a hash.

The integrity of the received data is evaluated by comparing both hash values. Whether the packet translation is analysed using *tcpdump* tool [59] in both mShield interfaces used.

```
#!/bin/sh
dd if=/dev/random bs=1M count=5000 of=random.data
sha1 -q random.data > client.hash
nc 192.1.1.64 9090 < random.data
```

Listing 4.3: IPv4/TCP client using netcat

```
#!/bin/sh
nc -l 2011:11 9090 -l random.data
sha1 -q random.data > server.hash
```

Listing 4.4: IPv6/TCP server using netcat

4.4.3 Fragmentation and MTU

Maximum Transmission Unit (MTU) defines the maximum size that a packet can have to be forwarded to another network segment. This value may change from one segment to another. At the network layer, those differences are handled using techniques, such as fragmentation or PMTUD. Other techniques

in the upper layers, such TCP segmentation and the use of *Maximum Segment Size (MSS)* field are out of the scope of this thesis since they are not IP related.

Fragmentation allows to split a bigger packet in several parts for instance at the middle of the path. Whether PMTUD is used to discover the lowest MTU along a network path and then, if necessary, fragment the packet at the origin. In both cases, it becomes necessary for a router or a converter box such as mShield be able to handle and forward IP fragments. The idea behind PMTUD is successively probe the path with larger packets until receive the information that the packet is too big reducing the size the next packets.

In IPv4, PMTUD is optional. The IPv4 header contains a *Do not Fragment (DF)* flag, that defines whether a packet can be fragmented along the path or not, that means enable or not PMTUD [60].

Nevertheless, in IPv6 all fragmentation information was shifted to an extension header and there is no such flag. Instead, it is used an implicit flag since the intermediate devices should not fragment packets and it is recommended to implement PMTUD [61] [3]. In that case, PMTUD is used to find out the maximum packet size that a packet can have, then the sender should fragment the packet according to that value.

In the first part of the test, regarding the PMTUD, two tests were performed, IPv6 to IPv4 and the other direction. A packet bigger than the next-hop MTU was sent and the reply analysed. In the following tests, in order to have a consistent and full control about the header fields, all packets were crafted using scapy tool [62]. For IPv4 packets, in addition to a bigger payload, it was used the DF IPv4 flag to explicitly enable the PMTUD process. The scripts used are represented in listings 4.5 and 4.6.

In the second part, instead of sending one single big packet, the original packet was fragmented before send it. That way it is possible to test whether the fragments are properly handled and forwarded or not. Listings 4.7 and 4.8, shown IPv4 and IPv6 modes. Communications initiated by customer devices or the other way behave in the same way. In the scripts and results are presented communication initiated from the outside.

```
#!/usr/local/bin/python
from scapy.all import *

ifo="vic0"
sip, dip="2012::11", "2012::64"
dmac="00:50:56:00:00:cc"
sport, dport=6666, 90
plen=1450

payload="#"*plen
udp=UDP(sport=sport,dport=dport)
ipv6=IPv6(src=sip, dst=dip)
eth=Ether(src=RandMAC(),dst=dmac)
packet=eth/ipv6/udp/payload
sendp(packet,iface=ifo)
```

Listing 4.5: IPv6/UDP packet using scapy

```
#!/usr/local/bin/python
from scapy.all import *

ifo="vic0"
sip, dip="192.1.1.1", "192.1.1.64"
dmac="00:50:56:00:00:cc"
sport, dport=6666, 9090
plen=1450
flags=2 # dont fragment set

payload="#"*plen
udp=UDP(sport=6666, dport=dport)
ipv4=IP(src=sip, dst=dip,flags=flags)
eth=Ether(src=RandMAC(),dst=dmac)
packet=eth/ipv4/udp/payload
sendp(packet,iface=ifo)
```

Listing 4.6: IPv4/UDP packet with DF flag set using scapy

```
#!/usr/local/bin/python
from scapy.all import *

ifo="vic0"
sip, dip="192.1.1.1", "192.1.1.64"
dmac="00:50:56:00:00:cc"
sport, dport=6666, 9090
plen=1450

payload="#"*plen
udp=UDP(sport=sport,dport=dport)
ipv4=IP(src=sip, dst=dip)
eth=Ether(src=RandMAC(),dst=dmac)
frags=fragment(eth/ipv4/udp/payload,800)
for f in frags: sendp(f,iface="vic0")
```

Listing 4.7: IPv4/UDP fragmented packet using scapy

```
#!/usr/local/bin/python
from scapy.all import *

ifo="vic0"
sip, dip="2012::11", "2012::64"
dmac="00:50:56:00:00:cd"
sport, dport=6666, 90
plen=1450

payload="#"*plen
udp=UDP(sport=sport,dport=dport)
hdrfrag=IPv6ExtHdrFragment()
ipv6=IPv6(src=sip, dst=dip)
eth=Ether(src=RandMAC(),dst=dmac)
frags=fragment6(eth/ipv6/hdrfrag/udp/payload, 800)
for f in frags: sendp(f,iface=ifo)
```

Listing 4.8: IPv6/UDP fragmented packet using scapy

4.5 Results

The folowing sections show the packets and the results obtained in each described test. Apendix B contain a description and the entire PF ruleset. For failed tests it is also described a proposed solution.

4.5.1 UDP and ICMP Translation

Both UDP and ICMP packets were successful translated. In the first part, with server running, since it is used UDP, there is no reply or acknowledge message from the server. Nevertheless, the captured packets in mShield, listing 4.9, reveal the expected translation. The original IPv4 packet to mShield was translated and forwarded to the other side, from mShield to a medical device. The source port of the translated address is random chosen by PF and it is part of the internal state whereas the destination port are copied from the original packet.

The second part, after the server shutdown, listing 4.10, the expected ICMP error message was received. An ICMPv6 generated message by the server was then translated to a ICMPv4 messages back to the client. The PF state created during the initial UDP request was used to translate back the ICMP message.

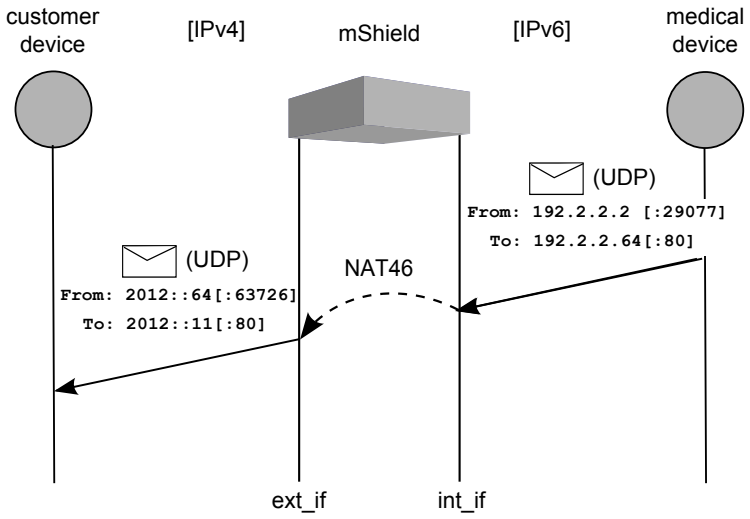

Figure 4.6: TODO:

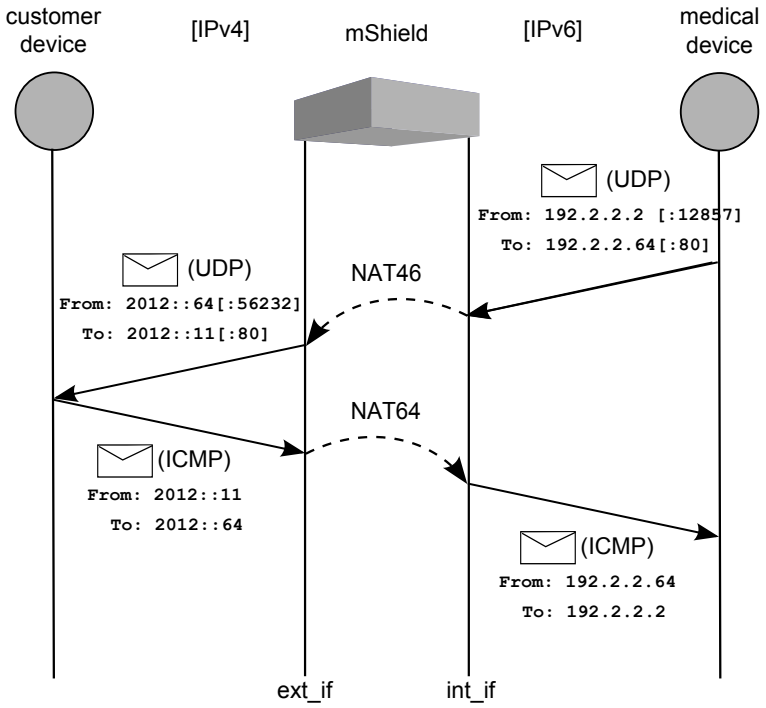

Figure 4.7: TODO:

```
#mshield
$ tcpdump -Nnti vic2
192.2.2.2.29077 > 192.2.2.64.80: udp 1031

$ tcpdump -Nnti vic0
2012::64.63726 > 2012::11.80: udp 1031
```

Listing 4.9: IPv4 to IPv6 UDP translation packets

```
#mshield
$ tcpdump -Nnti vic2
192.2.2.2.12857 > 192.2.2.64.80: udp 1031
192.2.2.64 > 192.2.2.2: icmp: 192.2.2.64 udp port 80 unreachable (DF)

$ tcpdump -Nnti vic0
2012::64.56232 > 2012::11.80: udp 1031
2012::11 > 2012::64: icmp6: 2012::11 udp port 80 unreachable
```

Listing 4.10: UDP translation with returned ICMP error messages

4.5.2 TCP connection

The TCP connection was successful translated. Listings 4.11 and 4.13 shows the establishment and finalization of the connection. Whereas the listing 4.12 shows a excerpt of the translated packets and their acknowledgments. The translations occurred in the same way previous explained for UDP.

Moreover, hash check test confirms the integrity of the data, meaning packets were successful transmitted. Since there is no fragmentation in any side the number of received packets was equal to the forwarded my mShield, excluding ARP and ND messages.

This simulation test not only IPv4 to IPv6 translation but also the other way, since the IPv6 / TCP acknowledgments need to be translated back. IPv6 initiated connections behave in the same way and are successful translated.

```
#mshield
$ tcpdump -Nnti vic0
192.1.1.1.1831 > 192.1.1.64.9090: S 238919288:238919288(0) win 16384 <mss 1460,nop,no
192.1.1.64.9090 > 192.1.1.1.1831: S 3877516634:3877516634(0) ack 238919289 win 16384
192.1.1.1.1831 > 192.1.1.64.9090: . ack 1 win 2048 <nop,nop,timestamp 3547517653 1576

$ tcpdump -Nnti vic1
2011::64.62893 > 2011::11.9090: S 238919288:238919288(0) win 16384 <mss 1460,nop,nop,
2011::11.9090 > 2011::64.62893: S 3877516634:3877516634(0) ack 238919289 win 16384 <m
2011::64.62893 > 2011::11.9090: . ack 1 win 2048 <nop,nop,timestamp 3547517653 157654
```

Listing 4.11: TCP three-way handshake translation output

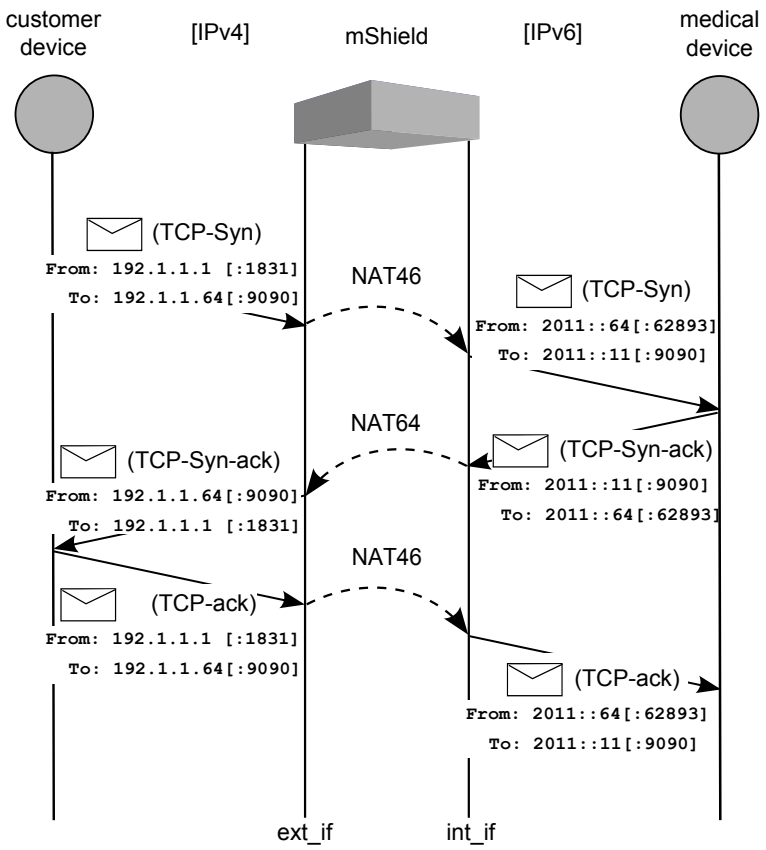

Figure 4.8: TODO:

```
#mshield
$ tcpdump -Nnti vic0
192.1.1.1.1831 > 192.1.1.64.9090: . 69165:70593(1428) ack 1 win 2048 <nop,nop,timesta
192.1.1.1.1831 > 192.1.1.64.9090: . 70593:72021(1428) ack 1 win 2048 <nop,nop,timesta
192.1.1.1.1831 > 192.1.1.64.9090: . 72021:73449(1428) ack 1 win 2048 <nop,nop,timesta
192.1.1.1.1831 > 192.1.1.64.9090: . 73449:74877(1428) ack 1 win 2048 <nop,nop,timesta
192.1.1.1.1831 > 192.1.1.64.9090: . 74877:76305(1428) ack 1 win 2048 <nop,nop,timesta
192.1.1.1.1831 > 192.1.1.64.9090: . 76305:77733(1428) ack 1 win 2048 <nop,nop,timesta
192.1.1.1.1831 > 192.1.1.64.9090: . 77733:79161(1428) ack 1 win 2048 <nop,nop,timesta
192.1.1.1.1831 > 192.1.1.64.9090: . 79161:80589(1428) ack 1 win 2048 <nop,nop,timesta
192.1.1.1.1831 > 192.1.1.64.9090: . 80589:82017(1428) ack 1 win 2048 <nop,nop,timesta
192.1.1.1.1831 > 192.1.1.64.9090: . 82017:83445(1428) ack 1 win 2048 <nop,nop,timesta
192.1.1.1.1831 > 192.1.1.64.9090: . 83445:84873(1428) ack 1 win 2048 <nop,nop,timesta
192.1.1.64.9090 > 192.1.1.1.1831: . ack 70593 win 1785 <nop,nop,timestamp 1576546072
192.1.1.64.9090 > 192.1.1.1.1831: . ack 73449 win 1428 <nop,nop,timestamp 1576546072
192.1.1.64.9090 > 192.1.1.1.1831: . ack 76305 win 1071 <nop,nop,timestamp 1576546072
192.1.1.64.9090 > 192.1.1.1.1831: . ack 79161 win 714 <nop,nop,timestamp 1576546072 3
192.1.1.64.9090 > 192.1.1.1.1831: . ack 82017 win 357 <nop,nop,timestamp 1576546072 3
192.1.1.64.9090 > 192.1.1.1.1831: . ack 84873 win 0 <nop,nop,timestamp 1576546072 354
$ tcpdump -Nnti vic1
2011::64.62893 > 2011::11.9090: . 69165:70593(1428) ack 1 win 2048 <nop,nop,timestamp
2011::64.62893 > 2011::11.9090: . 70593:72021(1428) ack 1 win 2048 <nop,nop,timestamp
```

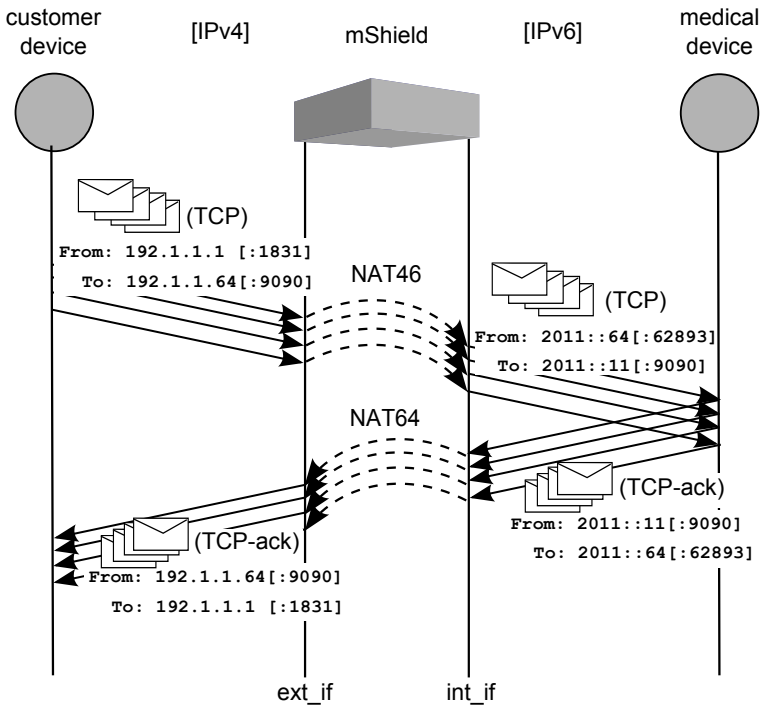

Figure 4.9: TODO:

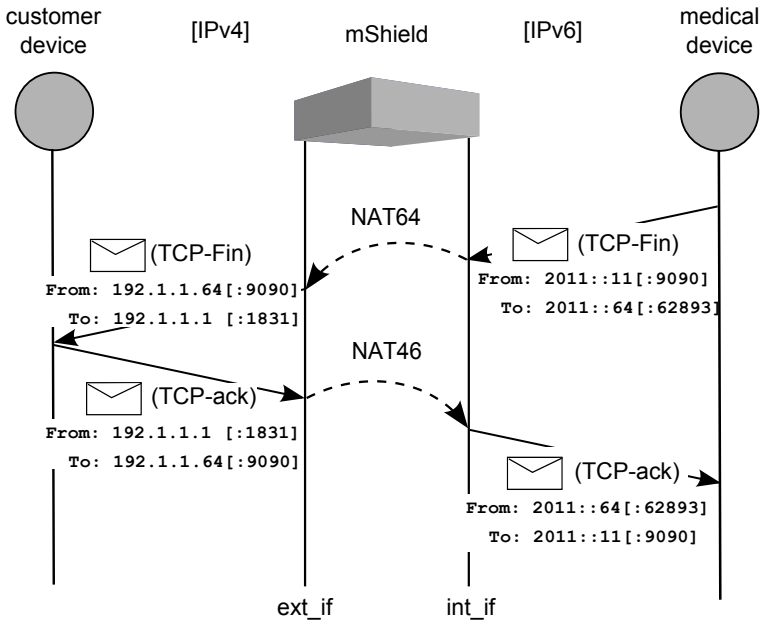

Figure 4.10: TODO:

```
#mshield
$ tcpdump -Nnti vic0
192.1.1.64.9090 > 192.1.1.1.1831: F 1:1(0) ack 52428802 win 32130 <nop,nop,timestamp
192.1.1.1.1831 > 192.1.1.64.9090: . ack 2 win 2048 <nop,nop,timestamp 3547517740 1576

$ tcpdump -Nnti vic1
2011::11.9090 > 2011::64.62893: F 1:1(0) ack 52428802 win 32130 <nop,nop,timestamp 15
2011::64.62893 > 2011::11.9090: . ack 2 win 2048 <nop,nop,timestamp 3547517740 157654
```

Listing 4.13: TCP connection termination translation output

4.5.3 PMTUD

In both address family translation modes, the PMTUD process The issue was reported to OpenBSD project [63].

In case of IPv6 to IPv4 translation,listing 4.14, ICMPv4 *Fragmentation Need* messages were sent to the loopback interface with wrong addresses. A ICMPv6 *Packet Too Big* message send back to the original sender is the expected behavior [6].

In case of IPv4 to IPv6 translation, listing 4.14, the IPv4 DF flag is ignored resulting in a IPv6 packet with a fragmentation header. A ICMPv4 *Fragmentation Need* message send back to the original sender is the expected behavior [6].

```
$ tcpdump -Nnti vic0
192.1.1.1.6666 > 192.1.1.64.9090: udp 1440 (DF)

$ tcpdump -Nnti vic1
2011::64 > 2011::11: frag (0x4c65749a:1352@0+) 63926 > 9090:  udp 1440
2011::64 > 2011::11: frag (0x4c65749a:96@1352)
```

Listing 4.14: PMTUD mShield outputs using NAT46 mode

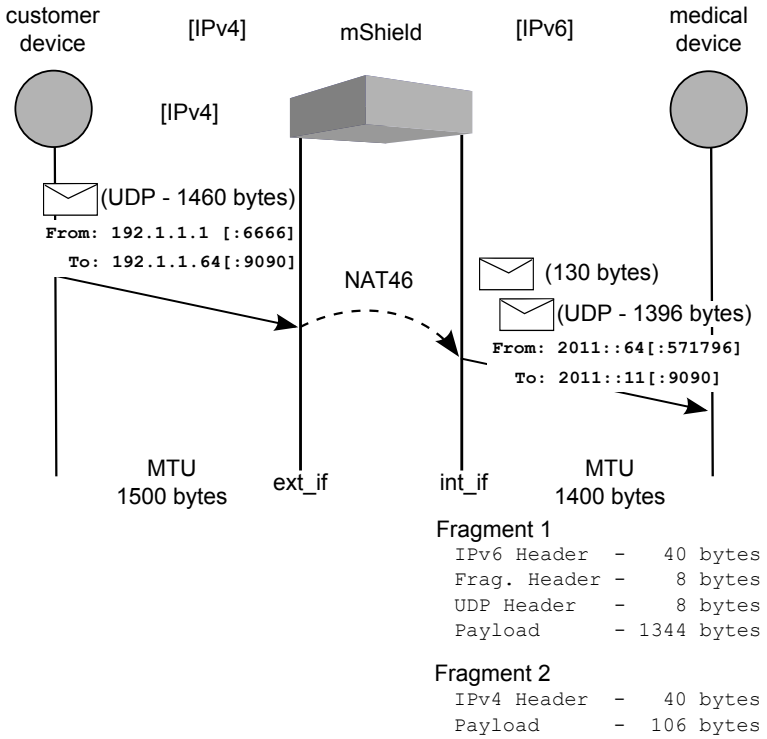

Figure 4.11: TODO:

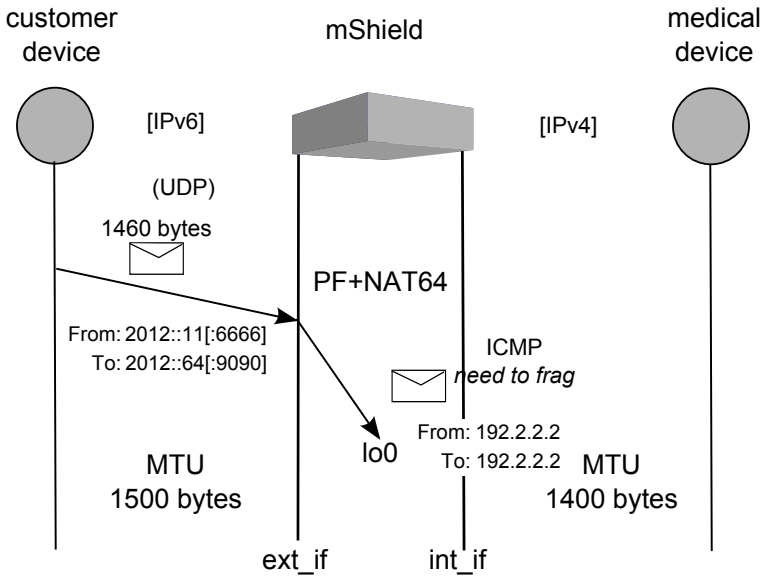

Figure 4.12: TODO:

```
$ tcpdump -Nnti vic0
2012::11.6666 > 2012::64.90: udp 1440

$ tcpdump -Nnti lo0
192.2.2.2 > 192.2.2.2: icmp: 192.2.2.2 unreachable - need to frag (mtu 1400)
```

Listing 4.15: PMTUD mShield outputs using NAT64 mode

4.5.4 Fragmentation Handling

In the NAT46 mode, the IPv4 fragments were successful translated and forwarded as shown in figure 4.16. Nevertheless, the IPv4 fragments are not directly translated to IPv6. They are reassembled by PF and then translated. Since the resulting packet is bigger than *next-hop* MTU, PF fragment it. For reassembled packets smaller than *next-hop* MTU, even if the original packet contains fragments, are not fragmented.

Voici du verbatim en couleur : `\verb en rouge`

Il y a aussi `du␣bleu␣étoilé␣^_\` puis par défaut `du noir ###` !

Maintenant, une ligne trop longue :

```
cos 1 + cos 1 + cos 1 + cos 1 + cos 1 + lg 2 + exp 3 + th 4 + ln
5 + ch 6 + cosh 7 + ln 5 + ch 6 + cosh 7
```

```
#mshield
$ tcpdump -Nnti vic0
192.1.1.1.6666 > 192.1.1.64.9090: udp 1450 (frag 1:800@0+)
192.1.1.1 > 192.1.1.64: (frag 1:658@800)
192.1.1.64 > 192.1.1.1: icmp: 192.1.1.64 udp port 9090 unreachable (DF)

$ tcpdump -Nnti vic2
2011::64 > 2011::11: frag (0x5b2f5b0e:1352@0+) 57197 > 9090:  udp 1450
2011::64 > 2011::11: frag (0x5b2f5b0e:106@1352)
2011::11 > 2011::64: icmp6: 2011::11 udp port 9090 unreachable
```

Listing 4.16: Fragmentation mShield outputs using NAT46 mode.

In the NAT64 mode, the IPv6 fragments are not properly handled, the

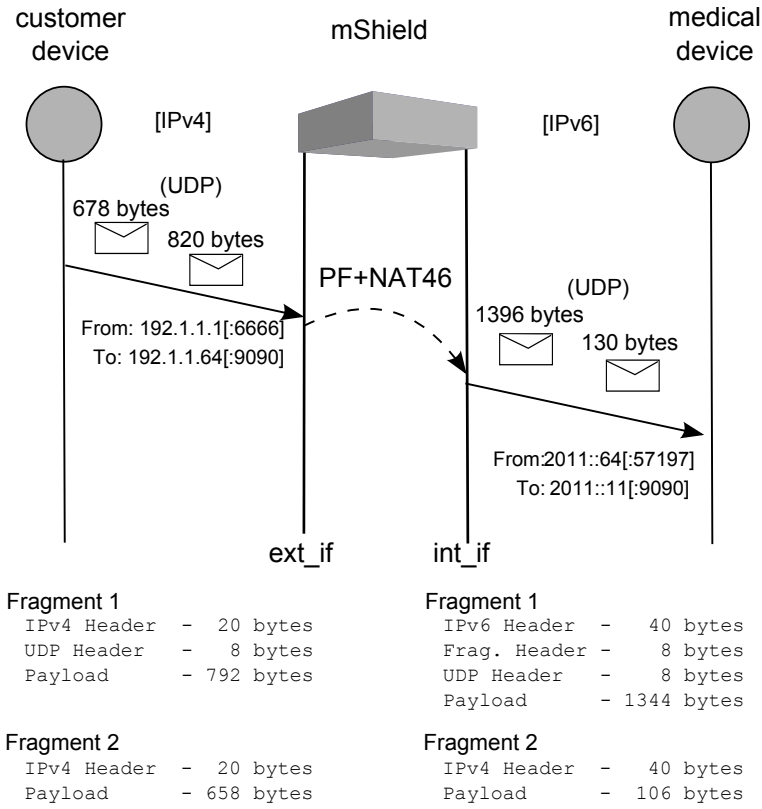

Figure 4.13: TODO:

packets were lost. Similar to NAT46 mode, the original fragments are first reassembled. During the translation, the information about the original fragments is discarded and the IPv4 DF flag is set. Such behavior will prevent a further fragmentation. As result of a reassembled packet bigger than *next-hop* MTU and DF flag active, PF tries unsuccessfully to send a ICMPv4 error message back. The captured packets are represented in figure 4.17.

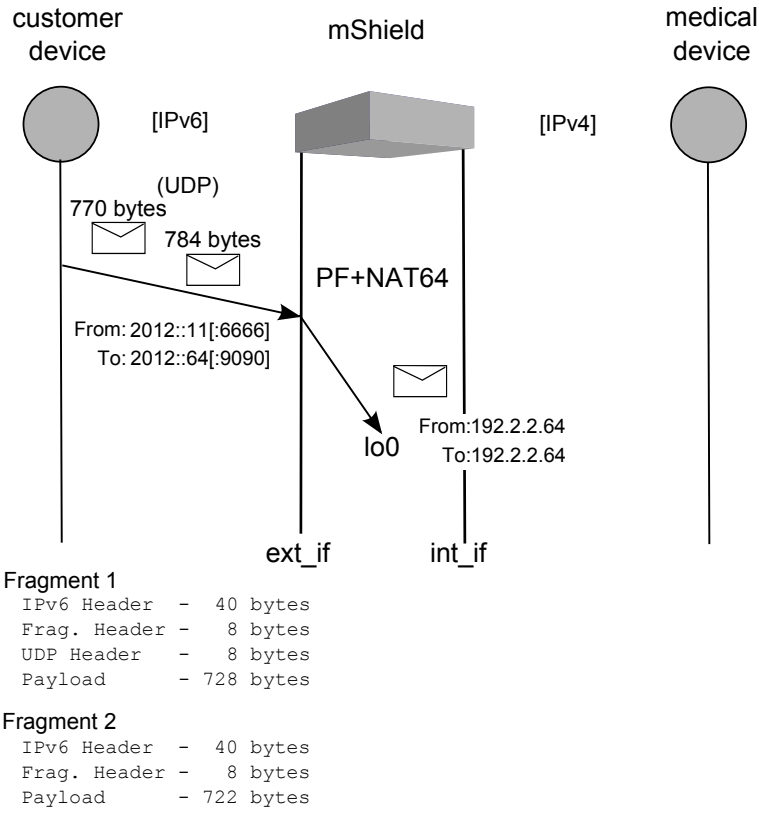

Figure 4.14: TODO:

```
#mshield
$ tcpdump -Nnti vic0
2012::11 > 2012::64: frag (0xf69cc880:736@0+) 6666 > 90:  udp 1450
2012::11 > 2012::64: frag (0xf69cc880:722@736)

$ tcpdump -Nnti lo0
192.2.2.64 > 192.2.2.64: icmp: 192.2.2.2 unreachable - need to frag (mtu 1400)
```

Listing 4.17: Fragmentation mShield outputs using NAT64 mode

A proposal patch, listing 4.18, instead of always assign the DF flag to the translated packet, check whether the original packet contained fragments and was reassembled. It tries to find a tag assigned during the reassembly process. Clear the flag means to allow a further fragmentation, for instance after the comparison with next-hop MTU.

```
$ diff -u pf.c.ori pf.c.new
--- pf.c.ori        Thu Nov  8 12:56:40 2012
+++ pf.c.new        Thu Nov  8 13:58:46 2012
@@ -2057,8 +2057,11 @@
         struct ip6_hdr                *ip6;
         struct icmp6_hdr        *icmp;
         int                            hlen;
+        struct m_tag            *mtag;
+        int                     reassembled;

         hlen = pd->naf == AF_INET ? sizeof(*ip4) : sizeof(*ip6);
+        reassembled = ((mtag = m_tag_find(pd->m, PACKET_TAG_PF_REASSEMBLED , NULL))

         /* trim the old header */
         m_adj(pd->m, pd->off);
@@ -2075,7 +2078,7 @@
                 ip4->ip_hl  = hlen >> 2;
                 ip4->ip_len = htons(hlen + (pd->tot_len - pd->off));
                 ip4->ip_id  = htons(ip_randomid());
-                ip4->ip_off = htons(IP_DF);
+                ip4->ip_off = htons(reassembled ? 0 : IP_DF);
                 ip4->ip_ttl = pd->ttl;
                 ip4->ip_p   = pd->proto;
                 ip4->ip_src = pd->nsaddr.v4;
```

Listing 4.18: pf.c patch to clear DF flag in the case of receive IPv6 fragments

Listing 4.19 shows the result after applying the patch. The original two IPv6 fragments containing 728 and 722bytes of data, were translated to a IPv4 and re-fragmented according to the MTU in two fragments of 1368 and 82bytes of data.

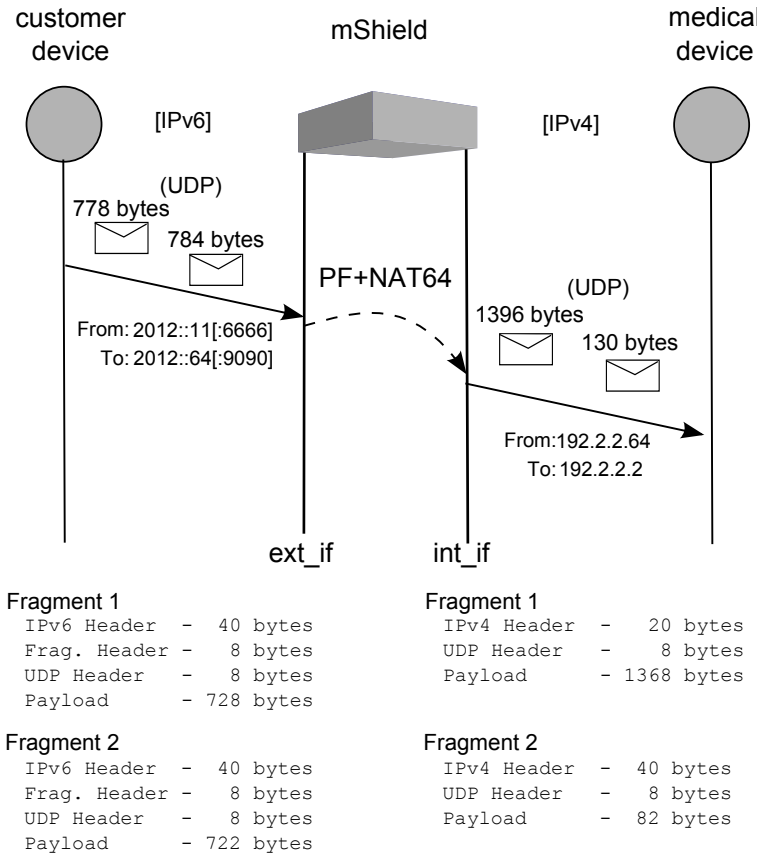

Figure 4.15: TODO:

```
#mshield
$ tcpdump -Nnti vic0
2012::11 > 2012::64: frag (0x8f91a9e2:736@0+) 6666 > 90:  udp 1450
2012::11 > 2012::64: frag (0x8f91a9e2:722@736)
2012::64 > 2012::11: icmp6: 2012::64 udp port 90 unreachable

$ tcpdump -Nnti vic2
192.2.2.64.51790 > 192.2.2.2.90: udp 1450 (frag 7707:1376@0+)
192.2.2.64 > 192.2.2.2: (frag 7707:82@1376)
192.2.2.2 > 192.2.2.64: icmp: 192.2.2.2 udp port 90 unreachable
```

Listing 4.19: Patched fragmentation mShield outputs using NAT64 mode

Not everything that can be counted counts, and not everything that counts can be counted.

Albert Einstein

5 Conclusions

5.1 Walkthrough

In chapter 2, an overview of the IPv6 theory and security concepts is provided. Particularly, the state of the art regarding IP translation mechanisms, namely the NAT64 framework.

Due to the IPv4 legacy devices, the transition approach is limited to IP translation. It can be done using the IETF proposed standard NAT64 or other non-standard method not covered by this thesis. In the theory review, it was observed a lack of studies about practical NAT64 deployments including practical security or performance issues.

The chapter 3, starts by describing the methodology used in the remainder of the document. It is composed of two main parts: a theoretical and a practical part. In the theoretical part, an analysis of the environment and its threats as well as a risk evaluation based on risk assessment guides [40] are presented. The practical part, design and implementaton, belongs to chapter 4.

The role of mShield is protect medical devices from network based attacks such as DoS attacks or malicious code propagation. Those threats surround the environment either in the current design or in the proposed design. Most of the security issues are related with IPv4 or IPv6 design and its implementation vulnerabilities. Nevertheless, in the security analysis is discussed an specific issue of current mShield design. Where it possible to trigger an DoS attack against medical devices by employing specific packet rates.

In chapter 4, a new mShield design is presented. The biggest change is related with moving mShield from bridge to routing mode and the necessary address translation. For the conversion role, the stateful NAT64 approach is proposed as the only available solution that really meets the outlined criteria either features or constraints. Each change is discussed and justified. Their implementation, part of this chapter, is presented in appendix B.

The performed tests revealed the expected behavior for TCP, UDP and ICMP. Nevertheless it is still necessary some patches for PMTUD and fragmentation support.

5.2 Problem Discussion

Using current and open source implementations it is possible to deploy a IPv4 to IPv6 converter box. The proposed design allows a full integration with current mShield design. It uses the same operating system and packet filtering mechanism instead of external tools or devices. That fact reduces the security impact resulting from changes in the design. Nevertheless, there are some security implications about the proposed design discussed in chapter 3.

Adding a new protocol increases the attack surface. It increases not only due to the use of a new protocol and its flaws but also due to their implementations and configurations.

Updating mShield to a non-transparent device, as proposed, exposes it to network scans and can be used to explore vulnerabilities. Nevertheless, it is unlikely that an OpenBSD vulnerability can be used to gain privileges and compromise the environment. There are no open OpenBSD vulnerabilities and in history its number is extremely reduced.

The network scans are possible either using IPv4 or IPv6 as well as in the current or the proposed design. Since there are no traffic authentication, mShield packet filtering can be bypassed using forged IP addresses. Therefore, it is possible scan the network discovering the allowed communications. On the other hand, vulnerabilities and malicious code exposing upper-layer protocols affect both IP versions.

At the end, the risk associated with the use of a new IP version will not significantly increase. A successful attack has the same impact on current or proposed design. In the most of the cases, the difference regards about the likelihood resulted from a bigger attack surface.

On the other hand, for the scenarios where the customer just use one IP, the attack surface can be reduced disabling one of the IP versions on external interface of mShield. In that case the attack surface are limited to one single IP version.

5.3 Main Contributions

The main contributions provided by this thesis are:

- State of the art discussion review from the IPv6 topics, such as the NAT64 approach, to the security concepts.
- Security analysis and risk assessment of the mShield environment.
- A design proposal and implementation of mShield as an IPv4 to IPv6 converter box.
- Practical tests on the implementation and its a discussion about its results. It includes an OpenBSD patch for PF and guidelines for the other failed tests.

5.4 Further Work

The following topics were identified as requiring further work:

- Additional research about IPv6 concepts and design vulnerabilities.
- Additional research for features such as DNS support, traffic authentication, anti-DoS mechanisms or traffic encryption at network layer.
- The proposal design and implementation represents a first step in the development cycle. It is necessary to develop and test additional components such as the configuration tool or dynamic PF configuration.
- More practical tests, including performance measurements in a real environment and reevaluation of the minimal hardware requirements.

5.5 Final Thoughts

TODO: I need to get inspired...

Acronyms

AH Authentication Header. 10, 81

ALG Application Layer Gateway. 17, 20, 32, 81

ARP Address Resolution Protocol. 11, 23, 59, 81

CERNET China Education and Research Network. 17, 81

CVE Common Vulnerabilities and Exposures. 39, 81

DAD Duplicate Address Detection. 13, 81

DDoS Distributed Denial of Service. 22, 24, 25, 81

DF Do not Fragment. 54, 62, 65, 66, 81

DHCP Dynamic Host Configuration Protocol. 13, 81

DHCPv6 Dynamic Host Configuration Protocol version 6. 13, 37, 81

DICOM Digital Imaging and Communications in Medicine. 32, 81

DNS Domain Name System. 13, 20, 72, 81

DoS Denial of Service. 14, 21, 23–25, 34, 36, 37, 39–43, 49, 70, 72, 81

DRDoS Distributed Reflective Denial of Service. 24, 81

ESP Encapsulated Security Payload. 10, 81

HTTP HyperText Transfer Protocol. 32, 47, 81

HTTPS HyperText Transfer Protocol Secure. 35, 81

IANA Internet Assigned Numbers Authority. 2, 81

ICMP Internet Control Message Protocol. 16, 17, 19, 20, 24, 32, 37, 52, 57, 70, 81

ICMPv4 Internet Control Message Protocol version 4. 11, 57, 62, 65, 80, 81

ICMPv6 Internet Control Message Protocol version 6. 10–12, 50, 57, 62, 80, 81

IDS Intrusion Detection System. 81

IETF Internet Engineering Task Force. 6, 15, 16, 46, 70, 81

IP Internet Protocol. 3, 9, 10, 12–17, 19, 20, 22–25, 29, 30, 32, 33, 35, 37, 38, 42, 43, 45–48, 50, 51, 53, 54, 70, 71, 74, 75, 77–81

IPS Intrusion Prevention System. 81

IPsec Internet Protocol security. 6, 32, 81

IPv4 Internet Protocol version 4. 2, 3, 6–10, 13–16, 18–21, 23, 27, 30, 32, 35, 37–39, 43, 45–47, 54, 57, 59, 62, 64, 65, 67, 70–72, 74, 80, 81

IPv6 Internet Protocol version 6. 2, 3, 6–23, 25, 27, 30, 32, 33, 35–40, 43, 45–48, 54, 59, 62, 64, 67, 70–72, 74, 80, 81

ISP Internet Service Provider. 25, 81

MAC Media Access Control. 14, 23, 37, 81

MSS Maximum Segment Size. 53, 81

MTU Maximum Transmission Unit. 53, 54, 64–67, 81

NA Neighbor Advertisement. 23, 81

NAT Network Address Translation. 14, 15, 38, 39, 48, 49, 81

nc Netcat. 52, 53, 81

ND Neighbor Discovery. 8, 12, 14, 39, 50, 59, 81

NRO Number Resource Organization. 2, 81

NS Neighbor Solicitation. 23, 81

OS Operating System. 33, 38, 39, 42, 45, 81

PCP Port Control Protocol. 50, 81

PF Packet Filter. 38, 45–48, 50, 51, 57, 64, 65, 72, 77, 79–81

PMTUD Path Maximum Transmission Unit Discovery. 52–54, 62, 70, 81

QoS Quality of Service. 10, 81

RA Router Advertisement. 13, 81

RDoS Reflective Denial of Service. 24, 81

RFC Request for Comments. 2, 15, 17, 81

RS Router Solicitation. 13, 81

RTT Round Trip Time. 81

SEND SEcure Neighbor Discovery. 14, 81

SIIT Stateless IP/ICMP Translation Algorithm. 16, 17, 81

SLAAC Stateless Address Autoconfiguration. 11, 13, 14, 33, 37, 50, 81

SSH Secure Shell. 33, 35, 47, 81

TCP	Transmission Control Protocol. 20, 24, 32, 50–53, 59, 70, 81
TLV	Type-length-Value. 81
UDP	User Datagram Protocol. 20, 24, 32, 50–52, 57, 59, 70, 81
UPnP	Universal Plug and Play. 49, 81

Bibliography

[1] R. Hinden and S. Deering, "IP Version 6 Addressing Architecture." RFC 4291 (Draft Standard), Feb. 2006. Updated by RFCs 5952, 6052.

[2] Number Resource Organization, "Free Pool of IPv4 Address Space Depleted." `http://www.nro.net/news/ipv4-free-pool-depleted`. [Online; accessed 30-August-2012].

[3] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification." RFC 2460 (Draft Standard), Dec. 1998. Updated by RFCs 5095, 5722, 5871, 6437, 6564.

[4] S. Deering, B. Haberman, T. Jinmei, E. Nordmark, and B. Zill, "IPv6 Scoped Address Architecture." RFC 4007 (Proposed Standard), Mar. 2005.

[5] C. Huitema and B. Carpenter, "Deprecating Site Local Addresses." RFC 3879 (Proposed Standard), Sept. 2004.

[6] X. Li, C. Bao, and F. Baker, "IP/ICMP Translation Algorithm." RFC 6145 (Proposed Standard), Apr. 2011.

[7] K. Nichols, S. Blake, F. Baker, and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers." RFC 2474 (Proposed Standard), Dec. 1998. Updated by RFCs 3168, 3260.

[8] S. Krishnan, J. Woodyatt, E. Kline, J. Hoagland, and M. Bhatia, "A Uniform Format for IPv6 Extension Headers." RFC 6564 (Proposed Standard), Apr. 2012.

[9] A. Conta, S. Deering, and M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification." RFC 4443 (Draft Standard), Mar. 2006. Updated by RFC 4884.

[10] T. Narten, E. Nordmark, W. Simpson, and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)." RFC 4861 (Draft Standard), Sept. 2007. Updated by RFC 5942.

[11] S. Thomson, T. Narten, and T. Jinmei, "IPv6 Stateless Address Autoconfiguration." RFC 4862 (Draft Standard), Sept. 2007.

[12] R. Droms, J. Bound, B. Volz, T. Lemon, C. Perkins, and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)." RFC 3315 (Proposed Standard), July 2003. Updated by RFCs 4361, 5494, 6221, 6422.

[13] J. Jeong, S. Park, L. Beloeil, and S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration." RFC 6106 (Proposed Standard), Nov. 2010.

[14] F. Gont, "Recent Advances in IPv6 Security." `http://www.si6networks.com/presentations/just4meeting2012/fgont-just4meeting2012-recent-advances-in-ipv6-security.pdf`, 2012. [Online; accessed 21-August-2012].

[15] T. Narten, R. Draves, and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6." RFC 4941 (Draft Standard), Sept. 2007.

[16] F. Gont, "A method for Generating Stable Privacy-Enhanced Addresses with IPv6 Stateless Address Autoconfiguration (SLAAC)." `http://tools.ietf.org/html/draft-ietf-6man-stable-privacy-addresses-00`, 2012. [Online; accessed 21-August-2012].

[17] J. Arkko, J. Kempf, B. Zill, and P. Nikander, "SEcure Neighbor Discovery (SEND)." RFC 3971 (Proposed Standard), Mar. 2005. Updated by RFCs 6494, 6495.

[18] P. Nikander, J. Kempf, and E. Nordmark, "IPv6 Neighbor Discovery (ND) Trust Models and Threats." RFC 3756 (Informational), May 2004.

[19] T. Aura, "Cryptographically Generated Addresses (CGA)." RFC 3972 (Proposed Standard), Mar. 2005. Updated by RFCs 4581, 4982.

[20] P. Srisuresh and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)." RFC 3022 (Informational), Jan. 2001.

[21] M. Wasserman and F. Baker, "IPv6-to-IPv6 Network Prefix Translation." RFC 6296 (Experimental), June 2011.

[22] F. Baker, X. Li, C. Bao, and K. Yin, "Framework for IPv4/IPv6 Translation." RFC 6144 (Informational), Apr. 2011.

[23] E. Davies, S. Krishnan, and P. Savola, "IPv6 Transition/Co-existence Security Considerations." RFC 4942 (Informational), Sept. 2007.

[24] C. Huitema, "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)." RFC 4380 (Proposed Standard), Feb. 2006. Updated by RFCs 5991, 6081.

[25] R. Bush, "The Address plus Port (A+P) Approach to the IPv4 Address Shortage." RFC 6346 (Experimental), Aug. 2011.

[26] I. Yamagata, Y. Shirasaki, A. Nakagawa, J. Yamaguchi, and H. Ashida, "NAT444 draft-shirasaki-nat444-06," 2012.

[27] A. Durand, R. Droms, J. Woodyatt, and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion." RFC 6333 (Proposed Standard), Aug. 2011.

[28] E. Nordmark, "Stateless IP/ICMP Translation Algorithm (SIIT)." RFC 2765 (Proposed Standard), Feb. 2000. Obsoleted by RFC 6145.

[29] "Network Address Translation - Protocol Translation (NAT-PT)." RFC 2766 (Historic), Feb. 2000. Obsoleted by RFC 4966, updated by RFC 3152.

[30] C. Aoun and E. Davies, "Reasons to Move the Network Address Translator - Protocol Translator (NAT-PT) to Historic Status." RFC 4966 (Informational), July 2007.

[31] X. Li, C. Bao, M. Chen, H. Zhang, and J. Wu, "The China Education and Research Network (CERNET) IVI Translation Design and Deployment for the IPv4/IPv6 Coexistence and Transition." RFC 6219 (Informational), May 2011.

[32] M. B. Simon Perreault, Jean-Philippe Dionne, "Ecdysis: Open-Source DNS64 and NAT64." http://www.viagenie.ca/publications/2010-03-13-asiabsdcon-nat64.pdf, 2012. [Online; accessed 21-August-2012].

[33] N. Skoberne and M. Ciglaric, "Practical Evaluation of Stateful NAT64/DNS64 Translation," *Advances in Electrical and Computer Engineering*, vol. 11, no. 3, 2011.

[34] C. Bao, C. Huitema, M. Bagnulo, M. Boucadair, and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators." RFC 6052 (Proposed Standard), Oct. 2010.

[35] M. Bagnulo, P. Matthews, and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers." RFC 6146 (Proposed Standard), Apr. 2011.

[36] M. Bagnulo, A. Sullivan, P. Matthews, and I. van Beijnum, "DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers." RFC 6147 (Proposed Standard), Apr. 2011.

[37] Microsoft, "The STRIDE Threat Model." http://msdn.microsoft.com/en-us/library/ee823878(v=cs.20).aspx. [Online; accessed 11-June-2012].

[38] P. Ferguson and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing." RFC 2827 (Best Current Practice), May 2000. Updated by RFC 3704.

[39] T. Cymru, "The Bogon Reference." http://www.team-cymru.org/Services/Bogons/, 2012. [Online; accessed 21-August-2012].

[40] G. Stoneburner, A. Y. Goguen, and A. Feringa, "Risk Management Guide for Information Technology Systems," tech. rep., 2002. [Online; accessed 16-July-2012].

[41] D. Malone, "Observations of ipv6 addresses," 2008.

[42] M. Heuse, "Recent advances in IPv6 insecurities reloaded - GOVCERT NL 2011." [Online; accessed 02-September-2012].

[43] G. F. Lyon, *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning.* Insecure, 2009.

[44] G. F. Lyon, "Nmap Script Engine Documentation." `http://nmap.org/nsedoc`. [Online; accessed 02-September-2012].

[45] X. Weilin, "6Guard: a honeypot-based IPv6 attack detector." `http://www.honeynet.org/node/944`. [Online; accessed 02-September-2012].

[46] P. Toan, "IPv6 attack detection tool." `http://www.honeynet.org/node/945`. [Online; accessed 02-September-2012].

[47] J. Abley, P. Savola, and G. Neville-Neil, "Deprecation of Type 0 Routing Headers in IPv6." RFC 5095 (Proposed Standard), Dec. 2007.

[48] The MITRE Corporation, "Common Vulnerabilities and Exposures." `http://cve.mitre.org`. [Online; accessed 30-August-2012].

[49] OpenBSD Team, "PF: The OpenBSD Packet Filter." `http://www.openbsd.org/faq/pf/`. [Online; accessed 03-September-2012].

[50] NLnet Foundation and Viagénie, "Ecdysis: open-source implementation of a NAT64 gateway." `http://ecdysis.viagenie.ca/`. [Online; accessed 11-June-2012].

[51] OpenBSD Team, "OpenBSD 5.1 Manual Page - pf.conf(5)." [Shell; accessed 03-September-2012].

[52] A. Jeffries, "IPv6 in Squid." `http://wiki.squid-cache.org/Features/IPv6`. [Online; accessed 09-September-2012].

[53] S. Cheshire and M. Krochmal, "NAT Port Mapping Protocol (NAT-PMP)," Internet-Draft draft-cheshire-nat-pmp-05.txt, IETF Secretariat, Sept. 2012.

[54] E. D. Wing, S. Cheshire, M. Boucadair, R. Penno, and P. Selkirk, "Port Control Protocol (PCP)," Internet-Draft draft-ietf-pcp-base-27.txt, IETF Secretariat, Sept. 2012.

[55] M. Wasserman, S. Hartman, and D. Zhang, "Port Control Protocol (PCP) Authentication Mechanism," Internet-Draft draft-ietf-pcp-authentication-00.txt, IETF Secretariat, June 2012.

[56] OpenBSD Team, "OpenBSD 5.1 Manual Page - dd(1)." [Shell; accessed 13-September-2012].

[57] Hobbit and E. Jackson, "OpenBSD 5.1 Manual Page - nc(1)." [Shell; accessed 13-September-2012].

[58] OpenBSD Team, "OpenBSD 5.1 Manual Page - sha1(1)." [Shell; accessed 13-September-2012].

[59] S. M. Van Jacobson, Craig Leres, "OpenBSD 5.1 Manual Page - tcpdump(8)." [Shell; accessed 13-September-2012].

[60] J. Mogul and S. Deering, "Path MTU discovery." RFC 1191 (Draft Standard), Nov. 1990.

[61] J. McCann, S. Deering, and J. Mogul, "Path MTU Discovery for IP version 6." RFC 1981 (Draft Standard), Aug. 1996.

[62] P. Biondi, "Scapy." `http://www.secdev.org/projects/scapy/`. [Online; accessed 09-October-2012].

[63] L. Rosa, "PMTUD issue using NAT64." `http://marc.info/?l=openbsd-tech&m=135109542921014`.

[64] R. F. Jason L. Wright, Andrew Thompson, "OpenBSD 5.1 Manual Page - bridge(4)." [Shell; accessed 15-September-2012].

[65] OpenBSD Team, "OpenBSD 5.1 Manual Page - vether(4)." [Shell; accessed 15-September-2012].

[66] OpenBSD Team, "OpenBSD 5.1 Manual Page - sysctl(8)." [Shell; accessed 15-September-2012].

[67] OpenBSD Team, "OpenBSD 5.1 Manual Page - sysctl.conf(8)." [Shell; accessed 15-September-2012].

[68] OpenBSD Team, "OpenBSD 5.1 Manual Page - pfctl(8)." [Shell; accessed 15-September-2012].