TALLINN UNIVERSITY OF TECHNOLOGY
Department of Computer Science
TUT Centre for Digital Forensics and Cyber Security

ITC70LT

Christian Ponti 144704

# WHAT APPROACH CAN BE USED TO GAIN NETWORK ACCESS FROM OUTSIDE BY USING ICMPV6?

Master Thesis

Supervisor: Bernhards Blumbergs
MsC

Tallinn 2016

# Autorideklaratsioon

Autorideklaratsioon on iga lõputöö kohustuslik osa, mis järgneb tiitellehele. Autorideklaratsioon esitatakse järgmise tekstina:

Olen koostanud antud töö iseseisvalt. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud. Käsolevat tööd ei ole varem esitatud kaitsmisele kusagil mujal.

Autor: [Ees− ja perenimi]

[November 26, 2016]

# Annotatsioon

Annotatsioon on lõputöö kohustuslik osa, mis annab lugejale ülevaate töö eesmärkidest, olulisematest käsitletud probleemidest ning tähtsamatest tulemustest ja järeldustest. Annotatsioon on töö lühitutvustus, mis ei selgita ega põhjenda midagi, küll aga kajastab piisavalt töö sisu. Inglisekeelset annotatsiooni nimetatakse Abstract, venekeelset aga

Sõltuvalt töö põhikeelest, esitatakse töös järgmised annotatsioonid:

- kui töö põhikeel on eesti keel, siis esitatakse annotatsioon eesti keeles mahuga $\frac{1}{2}$ A4 lehekülge ja annotatsioon *Abstract* inglise keeles mahuga vähemalt 1 A4 lehekülg;
- kui töö põhikeel on inglise keel, siis esitatakse annotatsioon (Abstract) inglise keeles mahuga $\frac{1}{2}$ A4 lehekülge ja annotatsioon eesti keeles mahuga vähemalt 1 A4 lehekülg;

Annotatsiooni viimane lõik on kohustuslik ja omab järgmist sõnastust:

Lõputöö on kirjutatud [mis keeles] keeles ning sisaldab teksti [lehekülgede arv] leheküljel, [peatükkide arv] peatükki, [jooniste arv] joonist, [tabelite arv] tabelit.

# Abstract

Võõrkeelse annotatsiooni koostamise ja vormistamise tingimused on esitatud eestikeelse annotatsiooni juures.

The thesis is in [language] and contains [pages] pages of text, [chapters] chapters, [figures] figures, [tables] tables.

# Glossary of Terms and Abbreviations

| | |
|---|---|
| *Node*: | a device that implements IPv6[1]. |
| *Link*: | a communication facility or medium over which nodes can communicate at the link layer, i.e., the layer immediately below IPv6[1]. |
| *Interface*: | a node's attachment to a link[1]. |
| *Neighbors*: | nodes attached to the same link[1]. |
| *Prefix*: | a bit string that consists of some number of initial bits of an address[1]. |
| *On-link*: | an address that is assigned to an interface on a specified link[1]. |
| *Off-link*: | an address that is not assigned to any interfaces on the specified link[1]. |
| *Longest prefix match*: | the process of determining which prefix in a set of prefixes covers a target address. A target address is covered by a prefix if all of the bits in the prefix match the left-most bits of the target address. When multiple prefixes cover an address, the longest prefix is the one that matches[1]. |
| *Reachability*: | whether or not the one-way "forward" path to a neighbor is functioning properly. In particular, whether packets sent to a neighbor are reaching the IP layer on the neighboring machine and are being processed properly by the receiving IP layer[1]. |
| *Packet*: | an IPv6 header plus payload[1]. |
| *Link MTU*: | the maximum transmission unit, i.e., maximum packet size in octets, that can be conveyed over a link[1]. |
| *Path MTU*: | the minimum link MTU of all the links in a path between a source node and a destination node[1]. |
| *Multicast capable*: | a link that supports a native mechanism at the link layer for sending packets to all (i.e., broadcast) or a subset of all neighbors[1]. |
| *Point-to-point*: | a link that connects exactly two interfaces[1]. |

| | |
|---|---|
| *Link-local address*: | a unicast address having link-only scope that can be used to reach neighbors[1]. |
| *All-nodes multicast address*: | the link-local scope address to reach all nodes, FF02::1[1]. |
| *All-routers multicast address*: | the link-local scope address to reach all routers, FF02::2[1]. |
| *Solicited-node multicast address*: | a link-local scope multicast address that is computed as a function of the solicited target's address. The function is chosen so that IP addresses that differ only in the most significant bits will map to the same solicited-node address thereby reducing the number of multicast addresses a node must join at the link layer[1]. |
| *Unspecified address*: | a reserved address value that indicates the lack of an address. It is never used as a destination address, but may be used as a source address if the sender does not know its own address. The unspecified address has a value of 0:0:0:0:0:0:0:0[1]. |
| *Covert Channel*: | is a communication paths that allow information transfer in violation of a system's security policies. In the context of network protocols, covert channel communication is generally achieved by manipulating an overt communication[2] [1]. |
| *Cover traffic*: | is the traffic that is being manipulated by covert channel participants[2]. |
| *Storage Covert Channel*: | manipulates a storage location in such a way that it conveys information to an observer. This definition was initially applied only to covert channels within a single machine or at least with a shared storage location. It was then extended to network covert channels and in this context, a storage channel is understood to be a channel that relies on modification of network traffic content[2]. |
| *Active Warden*: | is positioned so that it can observe and modify network traffic in its area of responsibility. The task of active wardens is to prevent and disrupt covert channel communication by modifying the content of network traffic[2] |

[1]open to view or knowledge; not concealed or secret, `http://www.dictionary.com/browse/overt`, accessed 1.4.2016

| | |
|---|---|
| *Policy*: | a formal, brief, and high-level statement or plan that embraces an organization's general beliefs, goals, objectives, and acceptable procedures for a specified subject area [2]. |
| *Standard*: | a mandatory action or rule designed to support and conform to a policy [2]. |
| *Procedure*: | procedures describe the process: who does what, when they do it, and under what criteria [2]. |
| *System Preservation*: | guarantees that the stegomessage is well formed within the rules of the protocol; the actual meaning of the stegomessage may be different than the original cover[3]. |
| *Semantic Preservation*: | means that, as observed at a point along the message's path through the network, the stegomessage has the same meaning as the original cover[3]. |
| *IDS*: | Intrusion Detection System. |
| *IPS*: | Intrusion Prevention System. |

[2] http://www.slu.edu/its/policies-and-processes, accessed 1.4.2016

# Contents

## List of Figures

# List of Tables

# 1. Introduction

IPv6 is the designated successor of IPv4, a protocol specified and implemented in a context with a limited number of users and hosts, most of them circumscribed to the scientific world. The need of a new protocol arose because of a changed context: the evolution of new powerful devices and their spread in many field of the society, which in turn modified the behavior and the requests of new entities, being them individuals or big organizations. The new IP protocol has been specified and redesigned in many aspects, taking in consideration the evolution of the requirements and the future needs of the involved actors.

IPv6[4], with respect to his predecessor, changed in many aspects. The headers have been modified to accommodate new functionality and improved capabilities, mainly it provides "expanded addressing capabilities", "header format simplification", "improved support for extensions and options", "flow labeling capability", and "authentication and privacy capabilities". One of the most relevant aspect is the increased address space from 32 bits to 128 bits, which deals with the demand of new communicating devices to fulfill organization's requirements.

Despite IPv6 specifications have been formalized in 1998, its spread and adoption by the world community is far from being accomplished. Among the many possible reasons that could explain this behavior, two of them deserve particular attention. The first one is related to the depletion of address space, which has been mitigated by the introduction of Network Address Translation (NAT)[5]: the use of NAT allows to use a private, not routeable, address space for the internal network of an organization, and the use of one, or limited number, public IPv4 address at the network boundary. This technique mitigated the need to request an IPv6 128 bits address by organizations, because their requirements to allocate new IPv4 address from IANA[3] have been reduced. The second reason is related to the applications and services offered by organizations. These applications have been written for, and tested against, IPv4. Many of them represents IT assets which are critical for the business assets and goals of enterprises: the introduction of new applications written for IPv6 represents a great effort in terms of financial investment, time for implementation and testing, and use of enterprises' resources.

The IPv6 world is composed by a number of protocols, which are used by nodes to fulfill

[3]http://www.iana.org/

their requirements: for some of them it is possible to find similar functionality in IPv4, while others are completely new protocols. In this set of protocols, one of them deserves particular attention: ICMPv6.

ICMPv6, despite it shares almost the same naming convention with respect with the predecessor, it is a new protocol, which make it a critical subject of research by the scientific community. The reasons behind its criticality arise because it holds some functions which are similar in ICMP (e.g. Echo Request), but at the same time it introduces a number of new functionalities, and responsibilities, for the correct behavior of an IPv6 node.

ICMP has been used by IPv4 to manage error and informational messages to allow for a better management and troubleshooting of the network. This protocol has been tested for many years, with the identification of vulnerabilities which can be potentially be exploited by malicious actors. Many best practices [4] [5] suggested to filter ICMPv4 messages at network boundaries to mitigate the risk of the exploitation of some vulnerabilities without compromising the network functionalities. Nowadays the network evolved in more sophisticated designs, and new concepts, like Bring Your Own Device (BYOD) and wireless networks, partially obsolete the very same concept of boundaries, bringing new threats to the internal network. The security controls could no more be applied only at the boundaries, but must be introduced in other segments of the network. This new security controls applies to the ICMPv4 protocol as well, and filtering must be introduced where such messages are not strictly required by network administrators.

"ICMPv6 is used by IPv6 nodes to report errors encountered in processing packets, and to perform other internet-layer functions, such as diagnostics (ICMPv6 Echo). ICMPv6 is an integral part of IPv6, and the base protocol (all the messages and behavior required by this specification) MUST be fully implemented by every IPv6 node"[6].

This paragraph of RFC 4443 describes broadly the purpose of ICMPv6, but the most important part is the last sentence, which states that the base protocol must be fully implemented by every IPv6 node. In the RFC terminology, the word "MUST, or the terms REQUIRED or SHALL, mean that the definition is an absolute requirement of the specification"[7]. The real distinction between ICMPv4 and ICMPv6 in this context is formal: RFC 792 states that

[4] http://www.cisco.com/c/en/us/support/docs/ip/access-lists/13608-21.html#anc29, accessed 20.03.2016

[5] http://www.cisco.com/c/en/us/about/security-center/firewall-best-practices.html#_Toc332805964, accessed 20.03.2016

“ICMP [...] must be implemented by every IP module”[8], which means that an implementation is required. The ICMPv6 specification is more precise because it refers to every IPv6 node, which covers the implementation inside the module, but also, with the concept of node, implies that the node must be able to use its functions.

This is a fundamental distinction, because the best practices in use with ICMPv4, with ICMPv6 are no more relevant, at least with some message types. As it is written in this paper[9], which cites RFC 4861[1], “ICMPv6 is used for basic functionalities and used by other IPv6 protocols [...] Neighbor Discovery Protocol is a protocol used with IPv6 to perform various tasks like router discovery, stateless automatic address configuration of a node, neighbor discovery, duplicate address detection, determining the Link Layer addresses of other nodes, address prefix discovery, and maintaining routing information about the paths to other active neighbor nodes”.

This is the first critical point to consider: despite version 4 and version 6 of the ICMP protocol have some similar functions, they are different protocols. ICMPv6 is not a protocol modified to adapt itself to IPv6, a new analysis must be performed in an exhaustive way taking into account new scenarios, and new best practices must be applied to mitigate the risk of exploitation of new vulnerabilities related to its functionalities.

Another important aspect to take into consideration when dealing with IPv6 in general, and with ICMPv6, is the transition from version 4 to version 6. For the aforementioned reasons, many organizations delayed as much as they could the deployment of an IPv6 native Internet connectivity. Nevertheless the scientific community continued to improve the new version and its related protocols, and Operating Systems (OS) started to include them in their network implementations. At the same time, techniques to allow a slow transition have been developed: examples of that are the dual stack, the presence and coexistence of both protocol versions in the same node, or the encapsulation of IPv6 inside IPv4, in those network segments where IPv6 has not been deployed yet. Additionally, in many OS, IPv6 is active by default and preferred over IPv4.

These aspects must not be underestimated and show again how critical and urgent is an in-depth analysis of ICMPv6: not only this protocol is fundamental for IPv6 and the best practices can be used only against the previous version, but it is already present and activated in OS that users employ in their everyday life. This last distinction is very important because it introduces another aspect to take into account: network administrator and security officer must not deal only with threats deriving from the exploitation of technical vulnerabilities, but also with behavioral procedures shaped around years of practice. IPv6 and ICMPv6 are not protocols that will be introduced in the future, they are already present in this transition period and we must deal with them now. Countermeasures must be in place, at the network boundaries and inside critical network segments above all, that take into consideration version 6 of the protocols. Such countermeasures must follow new best practices carefully shaped around the new version.

The underlined elements, the differences between the two ICMP version and the transition period with the coexistence of IPv4 and IPv6, lead to another topic: the impact of the full deployment of IPv6, and ICMPv6, and the implications for the involved actors, being them users, enterprises, states, or malicious actors.

The scientific community has been involved in the specification, in the improvement, and testing of the protocols since time. But, with respect with the preceding version of the protocols, version 6 of the suite must deal with a very different situation. The electronic communication is spread around the world in a pervasive way, and the near future, with the Internet of Things (IoT) [6], will further the spread. ICMPv6, and IPv6, will be fully deployed in a world strongly dependent on the electronic communication, where critical infrastructures (CI) must be managed and protected, where enterprises relies on its IT infrastructure to support their business assets, and where users are tight to their online experience for their everyday needs. This change in the very basic infrastructure of the network will have an impact which has no terms of comparison with respect to IPv4, where the initial small amount of users allowed to discover the concept of security and to learn by experience. The transition period, which introduced some security issues, in this case works as a mitigation technique to allow researcher to test the protocols in an exhaustive way, without waiting that the fully deployed IPv6 infrastructures reveals potential weaknesses.

This urgency can be better understood by taking into consideration the evolution of the threats, the sophistication of malware, and the malicious actors involved in the research of vulnerabilities to exploit, as well as their purposes and means to reach their goals.

The term Advanced Persistent Threat (APT) is used to define ”any sophisticated adversary engaged in information warfare in support of long-term strategic goals“[10]. One of the main

[6]IoT, `http://www.theinternetofthings.eu/what-is-the-internet-of-things`, accessed 21.03.2016

characteristic of APT detected in the wild is the sophistication of the attack. One example is the Uroburos rootkit[11], which ”modular structure allows extending it with new features easily, which makes it not only highly sophisticated but also highly flexible and dangerous“. The analysis of the rootkit suggests that ”the development of a framework like Uroburos is a huge investment“ and ”that it was designed to target government institutions, research institutions or companies dealing with sensitive information as well as similar high-profile targets“. There are other example of APT that suggested an evolution of the threat landscape, like Stuxnet[12]. Even if the aforementioned malware do not target directly IPv6, they share a common characteristic which is important to underline: the sophistication, the investment behind them, and the presence of an Advanced Persistent Adversary (APA)[13], term that ”depicts not only the threat, but the threat actors as well“.

The evolution of the threat landscape depicts a situation where APA, often associated with groups sponsored by state actors, have a remarkable budget to develop very sophisticated attacks to reach their goals. In the current situation, given the impact that the full deployment of IPv6 will have, those actors have the resources and the motivation to research and discover vulnerabilities in IPv6, ICMPv6, and other protocols involved. Moreover, if this research will succeed, APA can gain a considerable advantage over their opponents, because a possible vulnerability at network layer may allow to bypass more easily the security mechanisms in place and give more robust mechanisms to stay undetected for longer.

While research and tests on ICMPv6 are an ongoing process, the urgency and criticality of the subject require a more in-depth analysis which take into account the full deployment, but also the transition period. To pursue this goal it is important to start from the specifications of the protocol, the RFCs. Those documents represent a guideline, the result of an agreement between many stakeholders. As a guideline, there is no guarantee that the implementation will follow the suggestion, even in the sections specified with a ”must“.

It is worth to underline the contribution of this research. The first contribution is to produce the necessary awareness with regard to IPv6 and ICMPv6. This is the starting point to understand the need to analyze the protocols in detail. Furthermore, it is important to see the big picture, which include not only ICMPv6, but also its relationship with the devices involved in the communication, the target hosts, and the firewalls through which the packets must transit. Firewalls are devices configured by human beings, which may commit mistakes and use different best practices. Each configuration can produce a different scenario, which is worth to analyze because in the real world each system may differ from another, and the goal of an APA is to find these slight differences between them to take advantage. This work's contribution is an analysis of different scenarios where the use of the ICMPv6 protocol could arise different security issues, with respect to the configuration of network boundary devices. A scenario-based experiment needs also a test set, based on RFC specifications, and the implementation of a software to conduct the experiments, which also represent a contribution of this research.

The novelty of this research resides in the scenario-based experiment, which involves the identification of a test set based on the RFCs specifications, to find possible vulnerabilities that can be exploited. Furthermore, the tests must consider both directions of the network flow, from the outside to the internal network, and from the internal and trusted network to the outside. The traffic traveling in both directions must cross a network boundary device, which is a necessary condition to consider a potential vulnerability as such. The reason behind the novelty of this approach is that, as this research will underline in next chapter, other researches consider only one traffic flow direction, or the compliance with the protocol itself of most implementation, without considering network boundary devices, which, taking into account the transition period, may or may not be configured to cope with IPv6 and the related protocols. Furthermore, this research is developed from a penetration testing point of view. Penetration testing is a process with the aim to gain access to resources with the permission of the owner[14]. This point of view is important, because the process simulates what an attacker may do while attempting to gain access to a company's resources. As the research will point out, a vulnerability may exist, but its exploitation may depend on the configuration of a firewall, which might reduce the probability of the exploitation by a malicious actor.

The next chapters of this work will explore the background, the specification inside RFCs, and the related work(see 2), which includes existing tools and projects to test ICMPv6, and a review of the literature. Then the analysis proceeds with definition of the methodology(see 3) of this research, which is based on the scientific method of the experimentation; the chapter includes the motivation behind this choice and behind the choice of the technology to conduct the experiments. The implementation is the next chapter(see 4), which deal with the details of the technology, followed by the experiment(see 5), which describes the set of performed tests. The chapter related to the results(see 6) includes a discussion about the results of the experiment and its meaning. Finally, the conclusions(see 7) summarizes the research, discuss about suggestions and future work.

# 2. Background and Related Work

## 2.1. Background

"The Internet Standards process is an activity of the Internet Society[7] that is organized and managed on behalf of the Internet community by the Internet Architecture Board (IAB)[8] and the Internet Engineering Steering Group (IESG)[9]. [...] an Internet Standard is a specification that is stable and well-understood, is technically competent, has multiple, independent, and interoperable implementations with substantial operational experience, enjoys significant public support, and is recognizably useful in some or all parts of the Internet. [...] Each distinct version of an Internet standards-related specification is published as part of the Request for Comments (RFC) document series. This archival series is the official publication channel for Internet standards documents and other publications of the IESG, IAB, and Internet community."[15]

This work starts by analyzing the RFCs related to ICMPv6, which specify the protocol and the features to which each implementation must adhere. The main goal of an agreement on a protocol is interoperability [10], but, since this is not a binding process, and even though implementations follow the main specifications, it is always possible that some of them do not adhere completely. This could lead to some interoperability issues, which may introduce vulnerabilities in the protocol.

The background of this research is represented by two RFCs, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification"[6], and "Neighbor Discovery for IP version 6 (IPv6)"[1]. These two RFCs specify the general header for ICMPv6 messages, and each specific message header to which each implementation must adhere.

The IPv6 specification[4] is only partially in the scope of this research, but it must be mentioned, because of its tight relationship with ICMPv6, and because in each ICMPv6 message type specification some of its fields are cited, as it will be underlined later in the chapter.

[7] http://www.internetsociety.org/
[8] https://www.iab.org/
[9] https://www.ietf.org/iesg/
[10] http://www.merriam-webster.com/dictionary/interoperability

RFCs use a specific set of words to express the requirements of the implementations [7]. Words like “must”, or “should” have a particular meaning, as expressed in the cited document, that should be well understood by implementors. The reason is that a lack of compliance may lead to the introduction of some vulnerabilities, where a node is expecting a particular behavior from another peer, while the peer’s implementation follows slightly different specifications. This research, inside next sections, will try to underline where, under each message type, and along the general specifications (e.g. general processing rules), it is possible to insert vulnerabilities in the protocol if the interpretation of the specifications were different between the implementations.

Since the list of possibilities may be long, this research will use criteria in order to generate a subset of specification to test. The followed mechanism is to first select specifications and insert them into a set of tables, each related to a specific message, and then select from each table the test to be performed during the experiment. The criteria are presented at the end of this chapter, after the related works’ analysis.

### 2.1.1. RFC 2460

The table (see 28) shows the specification of the IPv6 header. From the point of view of this research, there are four interesting fields which are strictly involved in the ICMPv6 specification:

- **Next Header**: 8-bit, identifies the type of header immediately following the IPv6 header. Uses the same values as the IPv4 Protocol field. The ICMPv6 value is 58. The complete list is available on IANA's website[11]
- **Hop Limit**: 8-bit, decremented by 1 by each node that forwards the packet. The packet is discarded if Hop Limit is decremented to zero.
- **Source Address**: 128-bit address of the originator of the packet.
- **Destination Address**: 128-bit address of the intended recipient of the packet.

### 2.1.2. RFC 4443

The Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification categorizes two broad type of messages, error and informational messages.

As shown in the table (see 1), the general ICMPv6 header format is composed by an 8 bit **type** field, an 8 bit **code** field, and a 16 bit **checksum** field. The message body suggests specific header fields which are characteristic of each message type.

ICMPv6 messages are categorized by its type field, while the code field in this specification identifies a specific message under the category.

Error messages are characterized by a 0 in the high order bits of the type field, which gives a value range between 0 and 127. Informational message are instead characterized by a 1 in the high order bits, for a possible value range between 128 and 255.

The next two table (see 2 and 3) summarize the type of ICMPv6 messages described in this RFC.

[11] http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml, accessed on 27.03.2016

| Type | Code | Checksum |
|---|---|---|
| Message Body | | |

Table 1. ICMPv6 General Header Format

| Type field | Error Message |
|---|---|
| 1 | Destination Unreachable |
| 2 | Packet Too Big |
| 3 | Time Exceeded |
| 4 | Parameter Problem |
| 100 | Private experimentation |
| 101 | Private experimentation |
| 127 | Reserved for expansion of ICMPv6 error messages |

Table 2. ICMPv6 Error Messages

| Type field | Informational Message |
|---|---|
| 128 | Echo Request |
| 129 | Echo Reply |
| 200 | Private experimentation |
| 201 | Private experimentation |
| 255 | Reserved for expansion of ICMPv6 informational messages |

Table 3. ICMPv6 Informational Messages

The RFC, besides ICMPv6 fields, gives directives also for IPv6 fields of interest. For all the error messages and Echo Reply is the Destination Address field, which is "copied from the Source Address field of the invoking packet". For an Echo Request, the IPv6 field is "any legal IPv6 address".

Besides type and code fields, another field is common to all the ICMPv6 messages: the checksum. The checksum is "used to detect data corruption in the ICMPv6 message and parts of the IPv6 header" and "is the 16-bit one's complement of the one's complement sum of the entire ICMPv6 message, starting with the ICMPv6 message type field, and prepended with a pseudo-header of IPv6 header fields". The procedure implies to sum up the binary values of 16 bit at a time, and invert the bit values of the final result.

The RFC includes a "Message Processing Rules" section, which underlines the rules that a node must observe while processing an ICMPv6 message. The complete list is in table 44 in

Appendix 2.

During the preparation for the experiment it is important to identify tests that may lead to unexpected, from the RFC specification point of view, results. One example of that, taking the first rule in the aforementioned table, may be:
what would happened in the case that an ICMPv6 error message of unknown type is received at its destination, and it is passed to the upper-layer process with a forged malicious payload?

As this research will show, some test may be spread across different scenarios, while other are more suitable to be performed only in a specific scenario; the example above is a good candidate for an outside-to-inside direction scenario, in order to test the behavior of an internal node.

## Destination Unreachable

The first inspected message type is Destination Unreachable. Its purpose is to generate an error "in response to a packet that cannot be delivered to its destination address for reasons other than congestion".

For a detailed view of the header, table (see 29) in A.

The table (see 4) summarizes the type and code fields of the header, with the corresponding code name of the message.

| Type | Code | Description |
|---|---|---|
| 1 | 0 | No Route to Destination |
|  | 1 | Communication with destination administratively prohibited |
|  | 2 | Beyond scope of source address |
|  | 3 | Address unreachable |
|  | 4 | Port unreachable |
|  | 5 | Source address failed ingress/egress policy |
|  | 6 | Reject route to destination |

Table 4. Destination Unreachable Codes

The Unused field accordingly to the RFC must be initialized to zero by the originator and ignored by the receiver.

The following illustrates the meaning behind codes and description:

1. **No Route to Destination**: lack of a matching entry in the forwarding node's routing table
2. **Communication with destination administratively prohibited**: administrative prohibition (e.g., a "firewall filter")
3. **Beyond scope of source address**: the destination is beyond the scope of the source address (e.g., when a packet has a link-local source address and a global-scope destination address)
4. **Address unreachable**: the reason for the failure to deliver cannot be mapped to any of other codes
5. **Port unreachable**: generated in response to a packet for which the transport protocol (e.g., UDP) has no listener
6. **Source address failed ingress/egress policy**: the packet with this source address is not allowed due to ingress or egress filtering policies
7. **Reject route to destination**: the route to the destination is a reject route (may occur if the router has been configured to reject all the traffic for a specific prefix)

**Packet Too Big**

A Packet Too Big must be sent by a router in response to a packet that it cannot forward because the packet is larger than the MTU of the outgoing link.

The table (see 30) in Appendix 1 shows the header fields.

For this message the type field must always be 2 and the code set to 0 and ignored by the receiver. The MTU value represent "the Maximum Transmission Unit of the next-hop link".

**Time Exceeded Message**

The table (see 31) in Appendix 1 underlines the header fields of this type of message, table 5 shows the codes.

| Type | Code | Description |
|---|---|---|
| 3 | 0 | Hop limit exceeded in transit |
|  | 1 | Fragment reassembly time exceeded |

Table 5. Time Exceeded Codes

If a router receives a packet with a Hop Limit of zero, or if a router decrements a packet's Hop

Limit to zero, it must discard the packet and originate an ICMPv6 Time Exceeded message with Code 0 to the source of the packet. An ICMPv6 Time Exceeded message with Code 1 is used to report fragment reassembly timeout.

### Parameter Problem Message

“If an IPv6 node processing a packet finds a problem with a field in the IPv6 header or extension headers such that it cannot complete processing the packet, it must discard the packet and should originate an ICMPv6 Parameter Problem message to the packet’s source, indicating the type and location of the problem”.

The table (see 32) in Appendix 1 underlines the header fields of this type of message.

Table 6 shows a mapping between each code and a description of the error.

| Type | Code | Description |
|---|---|---|
| 4 | 0 | Erroneous header field encountered |
| | 1 | Unrecognized Next Header type encountered |
| | 2 | Unrecognized IPv6 option encountered |

Table 6. Parameter Problem Codes

Accordingly to the specification, the pointer field “identifies the octet offset within the invoking packet where the error was detected. The pointer will point beyond the end of the ICMPv6 packet if the field in error is beyond what can fit in the maximum size of an ICMPv6 error message”.

### Echo Request

The RFC states that each node must implements an Echo mechanism in order to produce Echo Requests and answer with Echo Replies. It also mentions the implementation of an application layer interface, for diagnostics purposes, to originate requests and receive replies.

The table (see 33) in Appendix 1 shows the header fields of an Echo Request message.

An Echo Request code is always 0. Identifier and Sequence Number are considered helping fields, which, in fact, could be expected to be zero, as depicted in table 7.

| Field | Value/Description |
|---|---|
| Type | 128 |
| Code | 0 |
| Identifier | An identifier to aid in matching Echo Replies to this Echo Request. May be zero. |
| Sequence Number | A sequence number to aid in matching Echo Replies to this Echo Request. May be zero. |
| Data | Zero or more octets of arbitrary data. |

Table 7. Echo Request Fields

**Echo Reply**

For the Echo Reply Header (see 34), while table 8 shows the fields with the value, where present, or the function.

| Field | Value/Description |
|---|---|
| Type | 129 |
| Code | 0 |
| Identifier | The identifier from the invoking Echo Request message. |
| Sequence Number | The sequence number from the invoking Echo Request message. |
| Data | The data from the invoking Echo Request message. |

Table 8. Echo Reply Fields

The RFC reiterate the implementation of the same mechanism as underlined in the Echo Request. In addition, it highlights that if the message is in response to a request to the unicast address of the node, the source address of the Echo Reply must be copied from the destination address field of the Echo Request.

An Echo Reply should be also originated in the case that the Request has been made to an IPv6 multicast or anycast address, in this case with the source address of the interface that received the message.

The data, which has no limitations in size, must be the same which has been received inside the Echo Request.

### 2.1.3. RFC 4861

The Neighbor Discovery Protocol has many functionalities and it is used by an IPv6 nodes in order to determine the link-layer address of neighbors, hosts and routers, and to discover which router is able to forward their packets to another off-link networks. In addition, it is used as a reachability mechanism to determine if a host is still alive.

The RFC defines five additional messages, characterized by the type field (for these messages the code is always zero), as summarized in table 9.

| Type field | Message |
|---|---|
| 133 | Router Solicitation |
| 134 | Router Advertisement |
| 135 | Neighbor Solicitation |
| 136 | Neighbor Advertisement |
| 137 | Redirect |

Table 9. RFC 4861 Messages

This RFC defines in addition five options, with the possibility to be extended in the future, that "provide a mechanism for encoding variable length fields, fields that may appear multiple times in the same packet, or information that may not appear in all packets". Options defined in the document are:

- Source Link-Layer Address
- Target Link-Layer Address
- Prefix Information
- Redirected Header
- MTU

#### Router Solicitation

The function of a Router Solicitation message, for a host, is to inform a router to generate a Router Advertisement. The format of a Router Solicitation is shown in Appendix 1 (see 35).

Router Solicitation specifies three IPv6 fields:

| | |
|---|---|
| Source Address | An IP address assigned to the sending interface, or the unspecified address if no address is assigned to the sending interface. |
| Destination Address | Typically the all-routers multicast address. |
| Hop Limit | 255 |

The next table (see 10) summarizes the requirements of the fields.

| Field | Value/Description |
|---|---|
| Type | 133 |
| Code | 0 |
| Reserved | This field is unused. It must be initialized to zero by the sender and must be ignored by the receiver. |
| Valid Options | |
| Source link-layer address | The link-layer address of the sender, if known. |

Table 10. Router Solicitation Fields and Options

**Router Advertisement**

Router Advertisement are sent by routers periodically, or in response to a Router Solicitation, to help hosts in their configuration processes. Examples of such processes are the mechanism to configure their addresses, how to obtain information about DNS servers, prefix information for the link.

The table 36 in Appendix 1 shows the header of a Router Advertisement.

Router Advertisement specifies three IPv6 fields:

| | |
|---|---|
| Source Address | must be the link-local address assigned to the interface from which this message is sent. |
| Destination Address | Typically the Source Address of an invoking Router Solicitation or the all-nodes multicast address. |
| Hop Limit | 255 |

The table (see 11) summarizes the fields with their specifications.

| Field | Value/Description |
|---|---|
| Type | 134 |
| Code | 0 |
| Cur Hop Limit | The default value that should be placed in the Hop Count field of the IP header for outgoing IP packets. A value of zero means unspecified |
| M | "Managed address configuration" flag. When set, it indicates that addresses are available via Dynamic Host Configuration Protocol |
| O | "Other configuration" flag. When set, it indicates that other configuration information is available via DHCPv6. |
| Reserved | It must be initialized to zero by the sender and must be ignored by the receiver. |
| Router Lifetime | The lifetime associated with the default router in units of seconds. |
| Reachable Time | The time, in milliseconds, that a node assumes a neighbor is reachable after having received a reachability confirmation. |
| Retrans Timer | The time, in milliseconds, between retransmitted Neighbor Solicitation messages. |
| Valid Options | |
| Source link-layer address | The link-layer address of the interface from which the Router Advertisement is sent. |
| MTU | should be sent on links that have a variable MTU. |
| Prefix Information | These options specify the prefixes that are on-link and/or are used for stateless address autoconfiguration. |

Table 11. Router Advertisement Fields and Options

**Neighbor Solicitation**

Neighbor Solicitation is used by nodes to request the Link-layer address of a neighbor, or to verify a neighbor's reachability.

See table 37 in Appendix 1 for the detailed header format.

Neighbor Solicitation specifies three IPv6 fields:

| | |
|---|---|
| Source Address | Either an address assigned to the interface from which this message is sent or (if Duplicate Address Detection is in progress) the unspecified address. |
| Destination Address | Either the solicited-node multicast address corresponding to the target address, or the target address. |
| Hop Limit | 255 |

Table 12 summarizes the fields with their specifications.

| Field | Value/Description |
|---|---|
| Type | 135 |
| Code | 0 |
| Reserved | It must be initialized to zero by the sender and must be ignored by the receiver. |
| Target Address | The IP address of the target of the solicitation. It must not be a multicast address. |
| Valid Options | |
| Source link-layer address | The link-layer address for the sender. |

Table 12. Neighbor Solicitation Fields and Options

**Neighbor Advertisement**

A Neighbor Advertisement is sent by a node in response to a Neighbor Solicitation, or to spread information quickly. The latter is called unsolicited Neighbor Advertisement.

Neighbor Advertisement specifies three IPv6 fields:

| | |
|---|---|
| Source Address | An address assigned to the interface from which the advertisement is sent. |
| Destination Address | For solicited advertisements, the Source Address of an invoking Neighbor Solicitation or, if the solicitation's Source Address is the unspecified address, the all-nodes multicast address. For unsolicited advertisements typically the all-nodes multicast address. |
| Hop Limit | 255 |

The table (see 13) shows the fields with the specifications.

| Field | Value/Description |
|---|---|
| Type | 136 |
| Code | 0 |
| R | Router flag. When set, the R-bit indicates that the sender is a router. |
| S | Solicited flag. When set, the S-bit indicates that the advertisement was sent in response to a Neighbor Solicitation from the Destination address. |
| O | Override flag. When set, the O-bit indicates that the advertisement should override an existing cache entry and update the cached link-layer address. |
| Reserved | It must be initialized to zero by the sender and must be ignored by the receiver. |
| Target Address | For solicited advertisements, the Target Address field in the Neighbor Solicitation message that prompted this advertisement. For an unsolicited advertisement, the address whose link-layer address has changed. The Target Address must not be a multicast address. |
| Valid Options | |
| Target link-layer address | The link-layer address for the target, i.e., the sender of the advertisement. |

Table 13. Neighbor Advertisement Fields and Options

**Redirect**

Redirect messages are used by routers to inform a node about a better first-hop node on the path to a destination.

Redirect specifies three IPv6 fields:

|  |  |
|---|---|
| Source Address | Must be the link-local address assigned to the interface from which this message is sent. |
| Destination Address | The Source Address of the packet that triggered the redirect. |
| Hop Limit | 255 |

Table 14 shows the fields with the specifications.

| Field | Value/Description |
|---|---|
| Type | 137 |
| Code | 0 |
| Reserved | It must be initialized to zero by the sender and must be ignored by the receiver. |
| Target Address | An IP address that is a better first hop to use for the ICMP Destination Address. When the target is the actual endpoint of communication, i.e., the destination is a neighbor, the Target Address field must contain the same value as the ICMP Destination Address field. Otherwise, the target is a better first-hop router and the Target Address must be the router's link-local address so that hosts can uniquely identify routers. |
| Destination Address | The IP address of the destination that is redirected to the target. |
| Valid Options |  |
| Target link-layer address | The link-layer address for the target. |
| Redirected Header | As much as possible of the IP packet that triggered the sending of the Redirect without making the redirect packet exceed the minimum MTU. |

Table 14. Redirect Fields and Options

**Options**

RFC 4861 specifies a set of options, which can be included in the messages (see 15). The general form includes an 8 bit identifier (Type) and an 8 bit length field, which represents "the length of the option in units of 8 octets. Options should be padded when necessary to ensure that they end on their natural 64-bit boundaries".

**Source/Target Link-layer Address**

"The Source Link-Layer Address option contains the link-layer address of the sender of the packet. It is used in the Neighbor Solicitation, Router Solicitation, and Router Advertisement packets. The Target Link-Layer Address option contains the link-layer address of the target.

| Type | Option Name |
|---|---|
| 1 | Source Link-Layer Address |
| 2 | Target Link-Layer Address |
| 3 | Prefix Information |
| 4 | Redirected Header |
| 5 | MTU |

Table 15. Options Type and Names

| Field | Value/Description |
|---|---|
| Type | 1 for Source Link-layer Address |
| | 2 for Target Link-layer Address |
| Length | The length of the option (including the type and length fields) in units of 8 octets. For example, the length for IEEE 802 addresses is 1 |
| Link-Layer Address | The variable length link-layer address. |

Table 16. Source/Target Link-layer Address Fields

It is used in Neighbor Advertisement and Redirect packets. These options must be silently ignored for other Neighbor Discovery messages."

**Prefix Information**

The Prefix Information is used by routers to inform hosts about on-link prefixes of the network segments, and allow them to perform address autoconfiguration (see 17).

**Redirect Header**

This options is used by Redirect messages and contains, in IP header and data, the original packet that triggered the redirect (see 18).

**MTU**

The MTU option is used to ensure that all the nodes on the link use the same MTU value, in such case where "heterogeneous technologies are bridged together" (see 19).

This ends the background analysis. Next, this research takes on the topic of covert channels, which is interesting because it covers the motivations and gives the possibility to exfiltrate data, after a malicious actor has gained a foothold in the internal network.

| Field | Value/Description |
|---|---|
| Type | 3 |
| Length | 4 |
| Prefix Length | The number of leading bits in the Prefix that are valid. |
| L | On-link flag. When set, indicates that this prefix can be used for on-link determination. When not set the advertisement makes no statement about on-link or off-link properties of the prefix. |
| A | Autonomous address-configuration flag. When set indicates that this prefix can be used for stateless address configuration. |
| Reserved1 | It must be initialized to zero by the sender and must be ignored by the receiver. |
| Valid Lifetime | The length of time in seconds (relative to the time the packet is sent) that the prefix is valid for the purpose of on-link determination. A value of all one bits (0xffffffff) represents infinity. |
| Preferred Lifetime | The length of time in seconds (relative to the time the packet is sent) that addresses generated from the prefix via stateless address autoconfiguration remain preferred. A value of all one bits (0xffffffff) represents infinity. |
| Reserved2 | It must be initialized to zero by the sender and must be ignored by the receiver. |
| Prefix | An IP address or a prefix of an IP address. The Prefix Length field contains the number of valid leading bits in the prefix. The bits in the prefix after the prefix length are reserved and must be initialized to zero by the sender and ignored by the receiver. |

Table 17. Prefix Information Fields

| Field | Value/Description |
|---|---|
| Type | 4 |
| Length | The length of the option in units of 8 octets. |
| Reserved | They must be initialized to zero by the sender and must be ignored by the receiver. |
| IP header + data | The original packet truncated to ensure that the size of the redirect message does not exceed the minimum MTU required to support IPv6. |

Table 18. Redirect Header Fields

| Field | Value/Description |
|---|---|
| Type | 5 |
| Length | 1 |
| Reserved | It must be initialized to zero by the sender and must be ignored by the receiver. |
| MTU | The recommended MTU for the link. |

Table 19. MTU Fields

## 2.2. Covert Channel

In the context of an attack, malicious actors can exploit a vulnerability to gain a foothold in the internal perimeter. This is a kind of scenario that concerns many actors, but it is not the only one. There is another scenario to take into consideration, especially if APTs and the amount of resources they have, are recognized. The case that the malicious actor has already gained a foothold and want to exfiltrate data without authorization. The absence of a proper authorization is an important concept, because it includes in the scenario also persons that are authorized to access the network, like employees, but have no authorization to send information outside the internal perimeter.

This scenario is characterized by the flow of the attack, from the internal perimeter to the outside, and by the need, from an attacker point of view, to stay undetected and, at the same time, to be able to pursue its goals.

The analysis of this scenario is presented next, and it is usually referred to with the presence of a covert channel.

Before attempting to analyze the scenario characterized by a covert channel in a networked communication, it is necessary to describe the context in which it came out first, that is, the Prisoners' Problem. This is important, because there are many similarities between the Prisoners' Problem and the context of a covert channel: two communicating actors, or processes, a warden, and a communication path that must be submitted to a policy in order to be allowed.

### 2.2.1. The Prisoners' Problem

The Prisoners' Problem involves two actors that "have been arrested and are about to be locked in widely separated cells", and a warden, who "is willing to allow the prisoners to exchange messages in the hope that he can deceive at least one of them into accepting as a genuine communication from the other either a fraudulent message created by the warden himself or else a modification by him of a genuine message"[16].

In the context of a networked communication, "Alice and Bob exploit an already existing communication path, corresponding to two arbitrary communicating processes: the sender and the receiver. Wendy is a warden, located somewhere along the communication path, monitoring all possible messages exchanged by Alice and Bob"[2].

In the analyzed scenario the attacker represents both the sender process located in the inside network, and the receiver process, located outside. The warden is a firewall, or a router, which allows specific communications from the inside to the outside, regulated by the organization's security policy, standards, and procedures. The challenge for the attacker is to find a communication path, which is allowed in the specific traffic direction flow, and use it as a covert channel, in order to deceive the countermeasures in place to consider it as a legitimate traffic.

### 2.2.2. Related Research

Covert channels are broadly divided into two categories: storage and timing. While the former implies a process that write into a shared resource, e.g. a protocol header field, the latter is based on the timing of events. Timing covert channels needs a synchronization mechanism, a clock in the receiving process which allows to measure the timing of events. For this reason, and because of its complexity, noise, and lower bandwidth, "[f]rom the attacker's point of view, network storage channels are preferable to timing channels"[17].

This research, because of the complexity of timing covert channels, considers only storage covert channels.

Lewandowski's research defines a communication model for network storage channels, which involves two parties who wish to communicate covertly. "As a cover, Alice and Bob might either select a suitable, already ongoing communication or generate an appropriate one if they can do so without arousing suspicion, and then they proceed to modify the cover communication's content to transmit their information. Meanwhile a third party, Wendy, positioned somewhere on the covert communication's path, attempts to disrupt Alice and Bob's efforts while preserving the integrity of the cover traffic"[2].

The author extracted six scenarios (see Figure 1) from the model, based on the work of Lucena[18].

The six scenarios depicts different positions of Bob and Alice in the communication model,

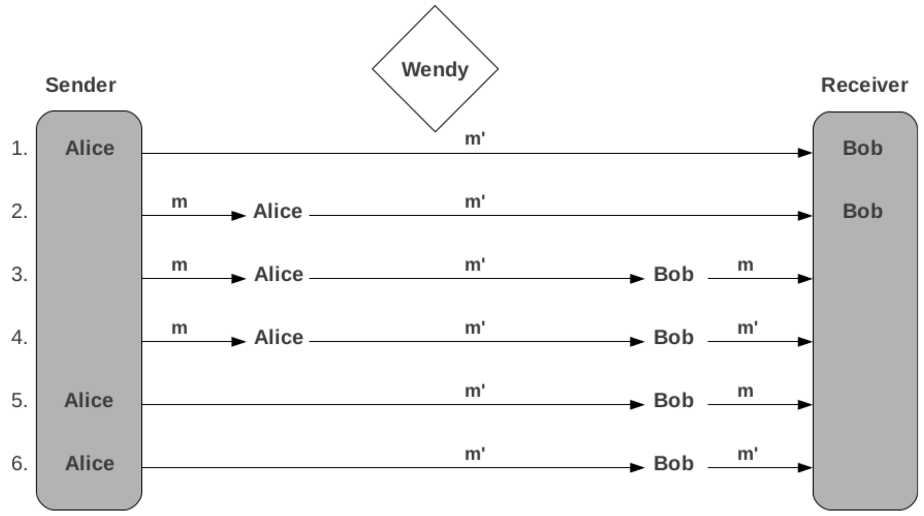


Figure 1. Communication Scenarios[2]

and the way the cover traffic (m) is manipulated by the covert channel (m′).

The first scenario is the simplest one, it involves a communication between Bob and Alice, where Alice use directly the covert channel and insert the manipulated message, which is directed to Bob, the receiver. The remaining scenarios explores different combinations, depending on the behavior and identity of the sender, i.e. if Alice is the sender and introduces a manipulated message in the channel or Alice modifies an existing and legitimate one, or on the behavior and identity of the receiver, i.e. whether Bob is the receiver or, where he is not, if he restores or not the original message after reception.

"In these scenarios, Wendy always should be positioned between Alice and Bob so that she can monitor m′ traffic. Were she positioned differently, and were unable to see m′, her presence would be irrelevant to the covert communication"[2].

For the scope of this research, it is assumed that the sender can control the covert channel and insert directly the manipulated message. This reduction in scope, with respect to Lewandowski's study, is justified by the limitation in time, and by the simplicity of the network configuration used for the experiment, which will be highlighted later in next chapters. However, it is worth saying that for an exhaustive testing of ICMPv6, related to covert channels, there are interesting implications to analyze inside the other scenarios, like the ability to preserve the covert channel in situations where Alice must modify the original message of the sender, or Bob must restore the original message before forwarding it to the legitimate receiver.

This assumption, however, has some implications:

“Alice can modify the traffic to a greater degree, since Bob does not necessarily expect the traffic to be meaningful and perhaps not even valid. On the other hand, if Alice and Bob use their own traffic to provide cover, they run a greater risk of exposure as they are openly communicating”[2]. Furthermore, in this research, it is assumed that the sender and the receiver are different processes, but controlled by the same subject, the attacker.

Another topic to take into consideration, which relates to the behavior of the active warden, is the system and semantics preservation. The concepts have been applied first to preserve steganography [12][3], but “the definitions can be applied to covert channels as well”[2].

In the context of covert channels, “the property of syntax preservation determines whether the modified traffic m′ adheres to the protocol syntax. On the other hand, the property of semantics preservation guarantees that the meaning of modified traffic m′ is the same as the original traffic m, or in other words that covert channel communication performed by Alice and Bob does not alter the meaning of cover traffic”[2].

The above concepts acquire more relevance in a complex topology scheme, where an IPv6 packet must eventually traverse multiple wardens. Lewandowski’s work defines the concept of “location-based syntax and semantics preservation”, which is useful to differentiate between nodes, along the path, “performing distinct functions” and with “multiple levels of protocol knowledge and understanding”. Each node, with a particular function in the network, may behave differently with respect to the packet in travel, and “as a result, a modified traffic’s syntax or semantics might be deemed correct by an IPv6 node with limited protocol knowledge while at the same time be rejected by a more knowledgeable node”[2].

This concepts will assume a particular relevance for this research when building the experiment. As it has been said before, it is important not only to understand the ICMPv6 protocol by itself, but also its relationship with devices like firewalls (the active warden) and the different configurations which produce different scenarios. Each configuration may lead to a

[12]“The art or practice of concealing a message, image, or file within another message, image, or file”, http://www.merriam-webster.com/dictionary/steganography, accessed 24.3.2016

different knowledge and understanding of the protocol by the node, which may affect the ability to preserve the covert channel, and must be taken into consideration.

For example, it is worth to mention the reserved and unused fields, which are present in some message's headers. The receiver must ignore them, they have no meaning and any modification does not change the semantics. If the syntax is not altered, these fields may be ideal candidates for covert messages[2]. Therefor, choosing a different configuration for the device, a firewall or router, may affect the level of knowledge of the protocol of that device, and the ability to preserve the covert channel. Or, in the other way, each scenario may produce different behaviors with the presence of a covert channel, and allow to assess a particular configuration and its ability to prevent it.

Indeed, "the objective of covert channel participants is to conduct their communication in such way that the necessary modifications of the cover traffic are always syntax and semantics preserving *with respect* to network nodes along the communication's path"[2].

**Analyzed RFCs**

The following table (see 20) shows the list of protocols and associated RFC from Lewandowski's study that are in scope with this research.

| Protocol | RFC |
|---|---|
| IPv6 [13] | RFC2460[4] |
| ICMPv6 | RFC4443[6] |
| Neighbor Discovery (ND) for IPv6 | RFC4861[1] |

Table 20. Investigated Protocols

**Properties of Covert Channels**

Lewandowski's study proposed six properties of covert channels which are important for the investigation. In this research, given its scope, two of them have been considered:

[13]RFC2460 is only partially in scope with this research, but it is worth to mention it because in ICMPv6 specification there are frequently references to IPv6 fields, like source and destination addresses

1. “degree of packet alteration – syntax- and semantics-preservation level of altered packets”. Since each configuration can have different syntax- and semantics-preservation level, it is important to test each covert channel against each configuration, to assess both the warden configuration and the covert channel in this particular scenario[2].
2. “channel bandwidth – amount of data that can be transfered in given covert channel per packet of cover traffic”. This property does not affect the existence of the covert channel, but its ability to be preserved for a longer period. Depending on the bandwidth, a covert channel may be useful, from attacker perspective, only in situations with a low amount of data to be exfiltrated, or when it is possible to insert a delay in the transmission to avoid suspicion on the traffic[2].

Lewandowski observed that some of the considered protocols, Neighbor Discovery is an example of them, “are designed for operation on a single network segment. In consequence, any covert channel using these protocols as a cover will be similarly limited in its range”, and that “the communication can be easily defeated by simple address-based filtering mechanism”[2].

This research, even though the observation may be correct, will not consider its assumption. The reason is given, as stated before, by the fact that it is important to consider also the actual transition period, where the two protocols, IPv4 and IPv6, coexist in many configurations. This scenario implies that both networks are in place, but the active warden is configured to inspect only IPv4-related protocols, thus with a low level of syntax and semantics preservation knowledge, and any IPv6 packet will transit without deep inspection. For the same reason, an address-based filtering mechanism for IPv6 would probably not be in place. Another aspect, still related to the active warden, is the direction of the communication, from the inside to the outside: since it is considered a communication from a trusted to an untrusted network, the default configuration may allow the traffic without restrictions.

**Analyzed Covert Channels**

In his research, Lewandowski defined a set of possible covert channels to test, which are spread among different protocols. For the scope of this research, only those that are mentioned in the aforementioned RFCs are taken into consideration.

Table 21 summarizes the protocol or message, the fields, and the bandwidth of the tested covert channels.

| Protocol/Message | Field | Covert Channel | Bandwidth |
|---|---|---|---|
| IPv6 | Source Address | False Source Address | 128 bits/packet |
| ICMPv6/All | Checksum | Arbitrary data | 16 bits/packet |
| ICMPv6/Destination Unreachable | Unused | Arbitrary data | 32 bits/packet |
| ICMPv6/Packet Too Big | Code | Undefined Code | 8 bits/packet |
| ICMPv6/Packet Too Big | MTU | Arbitrary data | 32 bits/packet |
| ICMPv6/Time Exceeded | Unused | Arbitrary data | 32 bits/packet |
| ICMPv6/Parameter Problem | Pointer | Arbitrary data | 32 bits/packet |
| ICMPv6/Echo Request | Code | Undefined Code | 8 bits/packet |
| ICMPv6/Echo Request | Data | Arbitrary data | Unlimited |
| ICMPv6/Echo Reply | Code | Undefined Code | 8 bits/packet |
| ICMPv6/Echo Reply | Data | Arbitrary data | Unlimited |

Table 21. Tested Covert Channels[2]

**Defense Against Covert Channels**

The first line of defense is the identification of a covert channel. Identification try to analyze the shared resource, i.e. header fields, during the design phase. Detection is the mechanism used to reveal an active covert channel. To detect and eliminate running covert channels it is possible to use protocol scrubbers, described in [19], traffic normalizers, and the aforementioned active wardens. Protocol scrubbers and traffic normalizers try to find and eliminate ambiguities in the traffic flow. Active wardens and traffic normalizers analyze incoming traffic and modify outgoing packets based on their knowledge of the protocol, e.g. setting to zero reserved or unused fields. Active wardens have different level of knowledge, and abilities to detect and eliminate attacks, depending on the type, which can be stateless, stateful, and network-aware. Stateless wardens have no knowledge of previous traffic. Stateful wardens can detect flows of traffic and base its decision accordingly to preceding knowledge of the flow. Network-aware wardens have also the knowledge of the network topology. In case of ICMPv6, it is possible to limit covert channels by disabling some message types, or limiting the protocol size and rate[17].

### 2.2.3. Existing Tools

**cctool**

Lewandowski built a tool called *cctool* in order to perform the tests. The tool has the ability to "verify the existence of the covert channels, as well as to test the functionality of active wardens". The results of his tests are strictly dependent on the quality of the wardens and on the scenarios used for the tests, which include, as mentioned before, also the possibility to modify the traffic in transit[2].

It is also worth mentioning some of the assumptions he made in order to better understand the tests. The syntax and semantics preservation are important factors in scenarios where the active wardens have the capabilities to understand, at different levels, the protocols. For example, Lewandowski built a scenario, called Cover Correctness Test, where a sender produces ICMPv6 traffic which is manipulated by Alice, but there is no receiver, namely Bob, at the end. The correctness of the covert channel, in this case, is demonstrated by the fact that an ICMPv6 Echo Request reaches the destination and an ICMPv6 Echo Reply is generated as usual. For such a test, "the conclusion is that the covert message embedding by Alice does not disrupt the cover traffic enough to impede its normal functionality"[2].

One limitation of this kind of tests, as the author specified, is that many covert channel cannot be tested, in particular where an answer is not expected back.

Another type of scenario, instead, include also Bob, who is able to receive the message and restore it back to its initial, syntactically and semantically correct, version. In this case it is possible to test also messages which do not include an answer in its processing rule, because the test environment expected that "both sender and receiver are located on a controlled network"[2].

The results performed by Lewandowski using cctool showed that the manipulation of the Source Address was defeated by the wardens. The tests on ICMPv6 implied an Echo Request, with correctness measured with the answer, resulted in a half of them defeated in the case of the manipulation of the Code field, while the covert channels were successful in case of the Data field manipulation.

One of the conclusions of Lewandowski study is "that specification-based countermeasures cannot defeat some of the more sophisticated attacks". In addition, with the tests performed with cctool, he demonstrated the existence of network storage covert channels in IPv6. Even though the test cases have not been exhaustive, his study has a great relevance for this research, in particular because of the accurate analysis and concepts underlined along the document[2].

### v00d00N3t

Another interesting tool that have been built with the purpose to test covert channel in ICMPv6 is *Voodonet*, or v00d00N3t. The purpose of the tool is to use ICMPv6 Echo messages as a covert channel to tunnel traffic. Covert Channel is embedded in the Data portion of Echo Reply message: the tool could send "Echo Reply messages with a payload of 1440 bytes in payload with no problem". The capabilities of the tool is to send and receive data from keyboard (chat) or as text files. Another interesting functionality, which increases the ability to maintain the covert channel, is a time- or bytes-based mechanism to throttle the dispatch of the data[20].

The two mentioned tools have been built mainly with testing purposes in mind. There is a suite of tools, instead, which has broadly functionalities and it is supposed to be used for penetration testing purposes: thc-ipv6.

### Thc-ipv6

The thc-ipv6 attack toolkit[14] is a "complete tool set to attack the inherent protocol weaknesses of IPV6 and ICMP6, and includes an easy to use packet factory library". The suite includes many features that could be used to test the IPv6 protocol in general, and the ICMPv6 protocol in particular[21].

*Alive6* is a tool used to check aliveness of IPv6 systems, both remotely using public addresses, and locally by taking advantage of an ICMPv6 Echo Request sent to the multicast addresses[21].

*Parasite6* can be used to perform a Man-in-the-middle attack to the internal LAN by exploiting the Neighbor Discovery process, and sending Neighbor Advertisements to nodes with the attacker MAC address and the IPv6 address of the requested node[21].

Another interesting tool is *redir6*, which use ICMPv6 redirect messages. The proposed technique takes advantage of the Redirect Header as an option for the Redirect message, which includes the IPv6 header and the data[21]:

[14] `https://www.thc.org/thc-ipv6/`, accessed 4.4.2016

1. Attacker sends an Echo Request with the target address as Source Address and the victim address as Destination Address.
2. The victim receive the Echo Request and sends an Echo Reply to the target (which can be intercepted by the attacker)
3. Attacker crafts a Redirect message, using the previous Echo Reply in the Redirect Header option, with local router as Source Address, victim as Destination Address, and attacker as Target Address

It is interesting to highlight that the cited paper, dated 2006, took into account the transition scenario and emphasized the possibility to attack the dual stack if proper filtering were not in place for both IPv4 and IPv6.

## 2.3. Other Projects

### 2.3.1. IPv6 Ready

The IPv6 Ready Logo [15] is "a conformance and interoperability testing program intended to increase user confidence by demonstrating that IPv6 is available now and is ready to be used". Its mission " is to define the test specifications for IPv6 conformance and interoperability testing, to provide access to self-test tools and to deliver the IPv6 Ready Logo" [22].

The project is composed by an Administrative Group, responsible for the administration, the definition of procedures and regulations, and by a Technical Group, which is responsible to define the tests specifications and submit them to the Administrative Group in order to be published for public review.

The procedure to obtain the certification and the possibility to exhibit the IPv6 Ready Logo on the products is to first download the test specifications. After that it is possible to use a self-testing tool or to submit the product to specialized and approved laboratories.

IPv6 Ready project entered the phase 2 and version 4.06 of the test since 2010. Phase 1 indicated "that the product includes IPv6 mandatory core protocols and can interoperate with other IPv6 equipments", while Phase 2 indicated "that a product has successfully satisfied

[15] https://www.ipv6ready.org, accessed 2.3.2016

strong requirements stated by the Logo Committee".

The project produced a detailed set of documents[23][24] which indicate the required network topology, as well as the detailed list of test to be performed against each protocol. Test are organized in sections based on RFCs, and inside each section in groups defined by their functionalities, e.g. Address Resolution and Neighbor Unreachability Detection for the section RFC 4861. Inside the section it is defined the scope, which describes broadly the content of the RFC, and the default packet of the message type (ICMPv6). The following table(see 22) is an example of a default Router Advertisement packet:

| Header | Field | Value |
|---|---|---|
| IPv6 Header | Source Address | TR[16] Link-Local address |
| | Destination Address | All-nodes multicast address |
| | Next Header | 58 |
| ICMPv6 Header | Type | 134 |
| | Code | 0 |
| | M Bit | 0 |
| | O Bit | 0 |
| | Router Lifetime | 20s |
| | Reachable Time | 10s |
| | Retrans Timer | 1s |
| Prefix Option | Type | 3 |
| | L Bit | 1 |
| | A Bit | 1 |
| | Valid Lifetime | 20s |
| | Preferred Lifetime | 20s |

Table 22. Router Advertisement Packet example for IPv6 Ready Project

In each group, e.g. "Address Resolution and Neighbor Unreachability Detection", the tests are described with a purpose, e.g. "Verify that a node correctly determines that a destination is on-link", the references inside the specifics RFCs, and the test setup with each packet definition, with the fields that differ from the mentioned default packet setup. Then the procedure to perform the test is shown, divided in parts. An example of that is the following:

*Part A: Link-local Address*:

1. TN1[17] transmits Packet A an Echo Request with TN1's link-local source address.
2. Observe the packets transmitted by the NUT[18].

[16]Testing Router
[17]Testing Node 1
[18]Node Under Test

which is followed by the “Observable Results”:

*Part A*:

2 The NUT should send a Neighbor Solicitation with Target Address equal to TN1’s link-local address, indicating that the NUT has successfully determined that TN1 was on-link.

The project covers all the RFCs in scope with this research with extensive tests, with great emphasis on RFC 4861, which is divided in three groups:

1. Address Resolution and Neighbor Unreachability Detection
2. Router and Prefix Discovery
3. Redirect Function

Tests cover the expected functionality, specified by the RFC, but also the behavior of nodes in case of manipulation of the messages, e.g. a wrong code field.

From the perspective of this research the test quality is very good, and it covers many possibilities to attempt to gain a foothold from the outside to the inside network. One missing part is the assessment of the opposite direction, from the inside to the outside, as is the case in presence of possible cover channels. The lack of references about the configuration of the wardens prevents the assessment of this kind of attacks, specifically with regards to the syntax and semantic preservation knowledge of the warden.

Another issue that it is worth to underline is that the project assumes an IPv6 world, without considering the transition scenario. This is not a critique towards the project, which aim is to test the readiness of devices and implementations with regard to IPv6, but only a consideration that limits the scope of the project towards this research.

## 2.4. Summary

Inside this section, this research started to analyze the background, that is RFCs document. RFCs are documents developed as an agreement to define standards upon which the Internet is built. The two main documents which are in scope with the research are RFC 4443 and RFC 4861, which define the ICMPv6 messages format and the processes for which they have been developed. Besides the header formats, RFCs include directives to the protocol implementation, using terms like MUST and SHOULD. This research examined the messages and presented them with a brief description. The actual header representations of each message are left to Appendix 1. Inside Appendix 2, it is possible to see the ICMPv6 specifications, with a selection of the specification and the term that the RFC uses to specify which feature is mandatory (MUST) or it is better to have. These specifications will also put together the tests selected for the experiment.

After the background analysis, this research started to examine the concept of Covert Channels, which is a "communication paths that allow information transfer in violation of a system's security policies". This is the first step on analyzing both directions of the network traffic flow, from the inside and trusted network to the outside. Even though a covert channel may exist in both directions, it is useful to understand, for example in case of APT, that an attacker may already have a foothold in the network and want to exfiltrate sensible data to an outside address. In this context it is important to consider the presence of wardens, firewall or routers, that may disrupt the covert channel. For such a reason, during the experiment, will be noteworthy to consider different scenarios based on different warden configurations.

The research continue by presenting the existing tools. Two of them, cctool and v00d00N3t, have been built to test the covert channels. The third is a suite of tools, or toolkit, which has been thought to perform security tests on IPv6 and ICMPv6.

After that, the IPv6 Ready project has been presented. The project's aim is to evaluate the readiness of the different products to IPv6 with an extensive set of tests based on the RFC specifications. The project offers, in many aspects, to analyze the opposite network flow direction with respect to covert channels, from the outside to the inside network.

In order to conclude this chapter, it is noteworthy to say that all the documents so far presented are, with different degrees, relevant for this research. The RFCs establish the base of the research, while the other documents offer an insight to understand the contest of some attacks and the reasons behind the choice to perform the tests. Besides those, other documents have been examined, but have been considered not relevant. Example are documents related to the common attacks on ICMPv6 and enumeration[25], RFCs related to the internal network only and not defining new messages[26], or related to internal Denial of Service attacks[27].

# 3. Methodology

The second chapter of this research starts by analyzing the background. The analyzed RFCs define the type of messages that compose the ICMPv6 protocol. For each message the RFCs specify the header and the fields, and the behavior of the protocol to fulfill its functions. The specifications include how to validate a message, the mandatory and optional fields, the data that each field must, should, or may deliver. The language use the terms must, should, may in order to underline what is mandatory to include in the implementation of the protocol, what it is better to include, and what may or not be implemented, leaving the developer more freedom in how to implement a task. The RFCs represents the knowledge about the protocol. The observation of the rules could lead to think if it is possible to break the rules in some way, and what could happen if different implementors interpret them in a different way.

Another observation regard the environment on which the protocol operate: a network based on a specific protocol suite. The Internet is a complex environment composed by devices interconnecting each other, using many protocols. This complexity make the Internet an environment not suitable to try to find an answer to the questions derived by the observation. This is because the amount of variables that it introduces do not allow to derive a unique consequence from an action, and thus reliable conclusions. In order to be able to derive reliable conclusions the environment must be simplified and be controllable. A controlled procedure needs to have exactly one variable, and to achieve that, as it has been mentioned before, a particular configuration must be defined. A specific configuration of the environment represents a constant, thus allowing to modify a specific rule. A set of tests is performed using manipulated rules against a well defined configuration. Then the same set of test can be performed against another configuration, leading to a number of tests equal to the number of configurations multiplied by the number of elements in the set.[28]

After the process of observing the background, and the environment in which the ICMPv6 protocol operate, this research found that the best methodology to follow is the scientific method. The reason behind the choice is that many element of the analysis of the second chapter, the observation of the data, led to the formulation of questions and hypothesis, that could be validated, or negated, by experimentation. As it will be better explained next, the process of the scientific method is ideal to test the hypothesis derived by the research questions, taking into account the different configurations of the environment which has been discussed in the second chapter.

## 3.1. Scientific Method

The scientific method has its roots in the ancient Greek philosophy, with Plato and Aristotle. The human reasoning has been the basis for the scientific evolution and studies, with the expression of an hypothesis, the argumentation about the truth or untruth, and eventually the formulation of a theory. Many centuries later, with the Enlightenment, two ways of looking at science coexisted in the developing scientific community: rationalism and empiricism. While the former accepted the human intuition, mainly in the field of natural science, and then applies testing hypothesis in order to formulate theories, the latter was more based on the observation of the data and then experimentation. Since then, both approaches have influenced the developing of the scientific method and reasoning, and applied to natural and human science.[29]

Another important distinction is between quantitative and qualitative research. Quantitative research is based on the use of statistics or on data that are quantifiable and measurable, in order to produce a result which is not biased on some context and, thus, could lead to a generalization of some theories. Qualitative research, instead, introduces the contextualization and relies on the context in which it is possible to observe a given phenomenon, which is more related to the human science, where observation and definition of a context in which human beings interact may allow to validate or not an hypothesis.[30]

This research uses both a quantitative and qualitative approach to perform the experiment. The qualitative approach is necessary to set up the experiment because of two factors: the choice of the tests and the considerations about the context. The tests are chosen starting from the description of the protocol inside the RFCs, which are for their nature not representable in a mathematical way. Among all the possible tests, some of them are considered for the experiment, as it will be presented later. The context is also very important and it involves the observation of human behavior in the configuration of the border network devices and the choices of the implementers of the end devices. The quantitative approach arises after the definition of the test set. Each test is performed against a well defined configuration. The set of tests and a specific configuration is an experiment where the only variable is the test itself, and the validation or not of the hypothesis can be done only for that specific configuration. In this way, this research is considering the quantitative approach, because inside the experiment, after its set up, no other variable other then the specific test will change, thus allowing it to be measurable in a binary way: either failed or succeed. The conclusions of the experiments, however, because of its context-dependent nature, could not lead to the formulation of a mathematical evaluation. In other words it wont be possible to state that the ICMPv6 protocol is vulnerable by itself, but only that, given this specific configuration, the protocol introduces some vulnerabilities.

Following the work of Peisert and Bishop in *How to Design Computer Security Experiments*, this research defines the following steps:

1. Form hypothesis
2. Perform experiment and collect data
3. Analyze data
4. Interpret data and draw conclusions

In addition to the aforementioned steps, Peisert and Bishop define three qualities, the most relevant in the domain of computer security, that the procedures requires. The procedures must be falsifiable, which means that the experiment must not be set up in order to justify a positive answer, but to test hypothesis that may ends up to be false. The procedures must be controlled, which involves the presence of exactly one variable. The procedures must be reproducible, which assumes that it is possible to repeat the experiment and obtain the same result.[28]

## 3.2. Research Questions and Hypothesis

The research questions are the starting point in order to define the hypothesis. Research questions express what the research want to know about the specific topic, and leads to definition of an hypothesis, which represents what the researcher want to validate (or invalidate) through the experiments. This research starts with a general research question, from which two additional and more specific research questions have been derived.

### 3.2.1. Research Question

*Is it possible to manipulate the ICMPv6 protocol and, by traversing border network devices, use in an unauthorized way a remote network?*

This question is a generalization which broadly introduces what this research want to know and test in relationship with the ICMPv6 protocol. Since the question introduces too many variables, it is useful and necessary to narrow down the scope, by defining two more specific research questions.

1. *Is it possible to exfiltrate data without authorization, from an internal network to the outside, traversing a border network device, by manipulating the ICMPv6 protocol?*
2. *Is it possible to gain unauthorized access to an internal network, from the outside and traversing a border network device, by manipulating the ICMPv6 protocol?*

### 3.2.2. Hypothesis

The formulation of the hypothesis follows the research questions. The hypothesis is the statement that the experiment aims to validate, which, following the aforementioned qualities of the process, may be negated by the experiment itself and by the following interpretation of data and conclusions.

The hypothesis of this research is:

*It is possible to manipulate the ICMPv6 protocol to use a remote network in an unauthorized way, after traversing a border network device*

Even thought this research will prove the truth of this hypothesis, some clarifications are necessary, and the final discussion inside the conclusions will take them into account. The first consideration is related to the general hypothesis itself, which, given its generality, may be less interesting from a scientific point of view. As stated before, each test and configuration is an experiment that may lead to prove the truth of the hypothesis for this exact experiment. The truth is related to a single experiment in the context of a general hypothesis, and the more experiments evaluate to true, the more relevant will be the results. It is very important to emphasize and understand that once a single experiment evaluate to true, the general hypothesis will be proved to be true. In other words, the truth of the hypothesis is the result of the logical *or* of the set of tests performed against each configuration. From this point of view the two more specific research questions, and the two sub-hypothesis, are more relevant because by narrowing down the problem, they define more specifically the context and give more indication about the results. The two sub-hypothesis are:

1. *It is possible to manipulate the ICMPv6 protocol to exfiltrate data without authorization from an internal network to the outside and traversing a border network device*
2. *It is possible to manipulate the ICMPv6 protocol in order to traverse a border network device and gain unauthorized access to an internal network*

The keywords of the first hypothesis are *exfiltrate data without authorization*. While the meaning of the first hypothesis is clear, the second one may be confusing. The keywords of the second hypothesis are *gain unauthorized access*. With gaining unauthorized access, this research do not refer only to the action of controlling a device remotely, but includes also the access to a process to stop it from working (i.e. denial of service) and the access to a process to change its behavior (i.e. man in the middle).

The assumption done before, that is, that the truth of the hypothesis is given by the logical *or* of the set of tests performed against each configuration, acquires more relevance in the case of one of the sub-hypothesis, because a more thorough specification of the problem has been made.

As it will be discussed in the conclusions, the relevance of the results is an important topic. This is because the aim of this research is not only to prove the truth or untruth of the hypothesis, but also to evaluate the impact that the exploitation of a vulnerability could have on an organization in relationship to the chosen wardens, the border network devices, and the deployed configuration.

## 3.3. Experiment Foundation

As shown before, the first step of the scientific method involves the definition of the research question, which in this case led to two sub-questions, and the formulation of the hypothesis and sub-hypothesis. The next step involves the execution of the experiment. This research will describe the experiment implementation in detail during next chapter and its execution afterwards. Before that, it is important to discuss the foundation of the experiment, that is, the choices and reasons behind it.

The selection of the test set is a critical task because it is the input of the experiment.

### Test set selection

"[T]he truth or falsity of every hypothesis is to some extent conditioned on the input data set [...] an additional step in testing a hypothesis is the validation of the input data set"[28]. The statement underlines the importance of the test set selection discussed in this section.

Inside chapter two (see 2), this research analyzed two RFCs which describe the ICMPv6 protocol and its messages. The analysis had the purpose to show the background of this research and to underline the basis to select the test set for the experiment. The background analysis produced a set of tables (see A and B).

In A it is possible to see the header of the messages with its fields. In the context of the set of tests, the fields acquire relevance to prove the truth of the first hypothesis, that is, the use of ICMPv6 to exfiltrate data from an internal network. Each field, with few exceptions, has been designed to carry data to fulfill a specification of the protocol. In the case of the data exfiltration described in this research, where the attacker controls both processes of sending and receiving data, only the border network device may perform some checks to validate the compliance with the protocol. The protocol itself may be used just as a transport vector for some data, by means of its fields, with the exception of the destination address specified in the IPv6 protocol, which is necessary to forward the packet to its destination. The aforementioned few exceptions are the unused or reserved fields, when defined by the message, which have no specific purposes in the context of the functionality of the protocol. In addition, each field shows an important indication for the purpose of exfiltrating data, the bandwidth. In fact, depending on the size of the field and of the data to be exfiltrated, it is possible to predict beforehand how many packets the process will use. This research, during the experiment, manipulates each ICMPv6 field and options headers against each border network device configuration, in order to test the hypothesis and assess the behavior of the warden.

In B, the tables show the specifications of each ICMPv6 message and the processing rules in terms of what must, should, or may be implemented. These specifications and processing rules define the behavior of the protocol in relationship with the functions for which it is used by other processes or protocols. Even though some of the specifications refers to an header field, as it has been mentioned above, the reasons behind the base used to choose the set of test in this appendix is different. While to test the first hypothesis both the sending and receiving processes are controlled by the attacker, the second hypothesis implies that the attacker controls only the sending process. In fact, if a packet successfully traverse the border network device, it is the behavior of the protocol and the implementation of the end device, i.e. the receiving process, to be under testing. In other words, it is the ending device that determines the truth of the hypothesis in case a packet has been able to reach it. In such a situation it is useful to consider the specifications that include the terms must, should, and may. This research assumes that at least the specifications including a must have been implemented. Nevertheless, each implementation may differ from another one by a very small degree, therefor such specifications must be included in the set of tests. The same applies to specifications characterized by the terms should and may, which, in addition, do not have the mandatory nature of the musts.

Another important aspect that has been taken into consideration in the choice of the set is the necessity to avoid the introduction of additional variables that it is not possible to control and verify in a reproducible way. An example of that is the last specification included in table 44, which specifies the need to limit the transmission rate of ICMPv6 error messages to limit the bandwidth and forwarding costs. Even though an high transmission rate may lead to a denial of service, there are other variables that affect the test of the specification, like the type of physical connection or the bandwidth usage of other processes. Therefor this type of specifications have not been selected.

The next step is the experiment environment and the reasons behind the choices.

**Environment**

The environment has been chosen taking into account the possibility to test the hypothesis and properties related to the methodology. To test the hypothesis, the experiment must include an attacking machine positioned in an outside network, which border is delimited by a border network device. The attacking machine is responsible for the receiving process in testing the first hypothesis (see Hypothesis 1), and it is used to send the manipulated ICMPv6 packets in testing the second hypothesis (see Hypothesis 2). The network configuration includes two wardens, a dedicated Cisco Adaptive Security Appliance (ASA)[19], and a Linux host with firewall (i.e. iptables) capabilities. Each experiment uses only a warden with a specific

[19]http://www.cisco.com/c/en/us/support/security/asa-5505-adaptive-security-appliance/model.html

configuration. The details of each configuration is discussed in the next chapter (see 4). The environment includes two end devices, one Linux host and one windows host. End devices purpose is to be responsible for the sending process in testing the first hypothesis (the Linux host), and, when testing the second hypothesis, to receive the ICMPv6 manipulated packets.

The choice of the devices for the experiment has been made taking into account the availability of the devices or virtualized environments. For virtualized environments, one limitation was the available memory (8GB) on the laptop used to develop the attacking PoC and to host the virtual machines. The first research about the environment aimed to have a complete virtualized environment. The difficulties in that research arose in finding a virtualized environment for one of the warden, because one of the objectives of this work is to test a security device which may be used in production systems, like a Cisco firewall. Since there are no freely available images, this possibility has been abandoned. After a bit of research, it has been possible to obtain a physical device (ASA firewall) on University's laboratory, which also has the advantage to decrease the overall need of memory for the virtualization. The virtualized environment is built on top of VirtualBox[20], chosen because it is available as open source software with GNU General Public License (GPL) version 2[21], and supports for many guest operating systems. The second warden is a Debian Jesse[22] Linux virtualized guest, which uses iptables[23] to provide firewall services. The virtualized environment has been chosen also for the two end devices, a Debian Jesse guest OS and a Windows 7 guest OS. In this case Debian Jesse has been chosen because it is a Linux open source OS, and because a Linux-based machine could be used as a server in production environments. The Windows 7 guest, instead, may represent common workstations in an internal network, and a virtual image could be downloaded[24] with a free trial license. The last device to consider is the attacking device, which had a constraint: because of the physical firewall, also that machine must be a physical device to form a network segment with the firewall. This constraint at the beginning posed some issues, because of a limited availability of devices. After some research the choice has fallen on a Raspberry Pi[25] with a Kali Linux OS. The reasons behind this choice is both on Kali Linux and Raspberry Pi as a attacking/pentesting device. Kali Linux has support for Raspberry Pi, and it is possible to download a pre-built image from

[20]https://www.virtualbox.org/
[21]http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
[22]https://www.debian.org/releases/jessie/
[23]https://wiki.debian.org/iptables
[24]https://developer.microsoft.com/en-us/microsoft-edge/tools/vms/windows/
[25]https://www.raspberrypi.org/

Offensive Security website[26], which are the creators of Kali. In addition, Raspberry Pi with Kali Linux may be a choice for attackers as a portable hacking station. Its dimension may allow to hide it or to use it without causing suspicion in places where the use of a laptop is not common.

**Border Network Device Configurations**

The firewall device, or service, is a key point in the security of the internal network of an organization. The market offers the state-of-the-art products, which include high performances in terms of forwarding capabilities, processing power, deep content inspection, services. Nevertheless, what make a firewall a security device that protect an internal network is the configuration. Most of the products include a default configuration, which must be perceived as a base, a starting point that should be tuned to the requirements of the organization. The default configuration may be enough to cope with the most common attacks, but from an attacker point of view it is sufficient to discover just one entry point in order to pass through. This is even more effective considering an APT.

The hypothesis of this research includes a firewall, as a border network device, because of the belief that the better way to test the ICMPv6 protocol is by correlating the implementation in end devices with the configuration of the firewall. Of course, each protocol may be vulnerable by itself, because of both its design or implementation, and dedicated research must be done as well. Nevertheless, there are scenarios in which, even if the exploitation of a vulnerability is possible, the surrounding environment may be a challenge for an attacker, reducing the likelihood to reach his goals.

Each firewall configuration represents a scenario, and an experiment, in which there is a set of tests and an end device. The set of configurations considered for this research is not exhaustive. The reason is that, even though there are best practices, each configuration must be fine-tuned for a particular set of requirements. The intent of this research is to show what may be the choice of different actors, based on different budgets.

A Linux-based machine, with firewall services, may be the choice of an organization with a low budget. Instead, a company with more resources may choose a dedicated device. This broad category includes organizations without security specialists, which use the default configuration, or companies with the needed expertise to follow best practices.

[26]https://www.offensive-security.com/kali-linux-arm-images/

In order to deploy meaningful configurations for the experiment, this research examined documents, best practices and RFCs.

The first document has been written by the National Institute of Standards and Technology (NIST)[27], which has a dedicated chapter on firewall policy and a sub-chapter on IPv6. Inside the section there is a statement that shows how much has to be done for IPv6 and ICMPv6, which says "[b]ecause IPv6 deployment is still in its early stages, there is not yet widespread agreement in the IPv6 operations community about what an IPv6 firewall should do that is different from IPv4 firewalls". The suggestions inside the document are more related to filtering packets with an invalid source or destination addresses, for example a link local address arriving at the boundary from the outside network. Another advice is to block all IPv6 traffic in the case that the organization do not use IPv6. For what is related to ICMP6, the document suggested to refer to RFC 4890, which will be examined next in this chapter.[31]

The second document have been draft by the National Aeronautics and Space Administration (NASA)[28] by means of the Communications Service Office (CSO), and used for the NASA Integrated Communication Services (NICS) contract[29]. The documents describes best practices related to IPv6 in detail. The relevance for this research is limited, because the document refers mostly to internal networks. Specifically, in the chapter related to firewall rules, the document warns about the need of neighbor solicitation and advertisement messages in order for the Neighbor Discovery protocol to work[32].

The most important suggestions for this research are described in RFC 4890. Chapter 4 divides firewall rules in two main categories, which, inside, define five broad sub-categories. The main categories are related to ICMPv6 transit traffic, and ICMPv6 local traffic. The following are the five sub-categories:

1. Messages that must not be dropped
2. Messages that should not be dropped
3. Messages that may be dropped in firewall/routers
4. Messages that administrators may or may not want to drop depending on local policy
5. Messages that administrators should consider dropping

Inside each category the RFC defines ICMPv6 messages that belongs to it, which in fact

[27]`http://www.nist.gov/`
[28]`http://www.nasa.gov/`
[29]`https://cso.nasa.gov/content/about-cso-and-nics`

allows to build firewall rules accordingly[33].

In addition, the RFC cites messages and technologies which are not in scope with this research: these messages are not taken into consideration.

The last source is related to the configuration of a Linux-based host with firewall services. The source also refers to RFC 4890, but as the author states, do not cover all the advices from there. Nevertheless this is a very good starting point to implement the Linux-based scenario of this research[34]. In addition, there is a basic script[30] that is downloadable from the main Computer Emergency Response Team (CERT) website, hosted by the Carnegie Mellon University[31].

Based on the above described information, this research decided to implement the following configuration categories:

1. Linux-based firewall services using ip6tables[32], forwarding without filtering
2. Linux-based firewall services using ip6tables, with RFC 4890 ICMPv6 best practices
3. Cisco ASA firewall with default configuration (without ICMP module)
4. Cisco ASA firewall with ICMP module for deep packet inspection

**Testing/Attacking application**

The experiment part of this research includes a Proof of Concept (PoC) which is able to manipulate and send ICMPv6 packets to test the hypothesis. The PoC is written in Python[33] as a programming language, which use an external library to manipulate ICMPv6 messages, called Scapy[34].

Python is a programming language which is possible to use in scripts, with functions, and with object oriented programming. It is installed by default in most Linux distributions and allows to use many third party libraries related to computer security and network testing. Example of that is a project supported by Open Web Application Security Project (OWASP)[35], called OWASP Python Security Project[36].

[30] http://www.cert.org/downloads/IPv6/ip6tables_rules.txt, accessed 22.4.2016
[31] http://www.cmu.edu/
[32] http://ipset.netfilter.org/ip6tables.man.html
[33] https://www.python.org/
[34] http://www.secdev.org/projects/scapy/
[35] https://www.owasp.org/index.php/Main_Page
[36] http://www.pythonsecurity.org/

Scapy is a very flexible tool that allows to build packets for many protocols. It allows to use the default protocol implementation, but it is also possible to manipulate packets or create new protocol from scratch. In addition, it allows to control the process of sending and receiving packets, and to visualize the content in different formats in order to build detailed reports. For these reasons, it the ideal tool to build a PoC for the experiment.

## 3.4. Summary

This research started this section by presenting the methodology, which is based on the scientific method. After a brief historical introduction, there is a description of the steps that the scientific method involves, starting from the research questions, which form the basis for the hypothesis, then going through the experiment, the analysis of the data, and finally the interpretation and conclusions.

After the general introduction of the methodology, this chapter went through the definition of the hypothesis that have to be tested during the experiment. The general hypothesis produced two more specific hypothesis, which are in relationship with the traffic flow direction of the ICMPv6 packets, from the internal network to the outside, and from the outside to the internal network, both of them traversing a firewall.

The next step in the chapter has been to describe the foundation of the experiment. This research first defined the reasons behind the choice of the test set for each hypothesis. The environment is another important aspect, with the choice of the involved devices which, sometimes, has been conditioned by the available resources.

The configuration of the firewall, as border network device, is another important topic, because each configuration represents a scenario. A scenario is a representation of what might be the choice of an organization for a firewall, based on aspects like the available budget, or the needed expertise to configure it.

Finally, this chapter introduced the components used to build the testing/attacking PoC, which are python as a programming language and scapy as packets manipulating library.

The next chapter will dig deeper in the implementation of the experiment, with the details of each involved component.

# 4. Implementation

The chapter is divided into four main parts. The first represents the technical details of each involved device. The second represents the network used to test both hypothesis, and the steps to go from the first to the second network configuration. The third describes in deep each firewall configuration. The fourth part is dedicated to the PoC and the Test Set.

## 4.1. Technical Details - Devices

**Virtualbox**

Virtualbox is the virtualized environment used to host the Linux Debian device, the Windows 7 device, and the Linux Debian Firewall.

**Virtualbox version:**

VirtualBox Graphical User Interface Version 5.0.24_Debian r108355

**Raspberry Pi with Kali Linux**

| Model | OS | OS and Software Update |
|---|---|---|
| Raspberry Pi 2 Model B | Kali Linux | 31.08.2016 |

Table 23. Raspberry Pi implementation

The OS has been downloaded from: https://www.offensive-security.com/kali-linux-arm-images/, and the updated on 31.08.2016 with the following commands (common for all Linux devices, root privileges required):

```
apt-get update
apt-get upgrade
apt-get dist-upgrade
```

**Kernel version (uname -a):**

Linux mazinga 4.1.19-v7 #1 SMP Tue Mar 15 15:10:00 CDT 2016 armv7l GNU/Linux

**/etc/network/interfaces:**

```
auto lo
iface lo inet loopback
```

```
allow-hotplug eth0
iface eth0 inet6 static
      address 2001:db8:acad:1::2
      netmask 64
      gateway fe80::1
```

**Python version:**

Python 2.7.12+

**<u>Linux Debian</u>**

**Device configuration**:

| Base Memory: | Processor: | Video Memory: |
|---|---|---|
| 2048MB | 2CPU | 24MB |

Table 24. Linux Debian

The OS has been downloaded from: `https://www.debian.org/releases/jessie/debian-installer/`, installed and the updated on 31.08.2016.

**Kernel version (uname -a):**

Linux debian 3.16.0-4.amd64 #1 SMP Debian 3.16.7-ckt25-2+deb8u3 (2016-07-02) x86_64 GNU/Linux

**First Hypothesis - Network Configuration (/etc/network/interfaces)**

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet6 static
      address 2001:abcd:acad:2::2
      netmask 64
      gateway fe80::1
```

**Second Hypothesis - Network Configuration (/etc/network/interfaces)**

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet6 auto
```

**Python version:**

Python 2.7.9

## Linux Debian Firewall

**Device configuration**:

| Base Memory: | Processor: | Video Memory: |
|---|---|---|
| 1024MB | 1CPU | 12MB |

Table 25. Linux Debian Firewall

The OS has been downloaded from: `https://www.debian.org/releases/jessie/debian-installer/`, installed and the updated on 31.08.2016.

**Kernel version (uname -a):**

Linux debian 3.16.0-4.amd64 #1 SMP Debian 3.16.7-ckt25-2+deb8u3 (2016-07-02) x86_64 GNU/Linux

**Network Configuration:**

**/etc/network/interfaces**

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet6 static
      address 2001:db8:acad:1::1
      address fe80::1
      netmask 64

auto eth1
iface eth1 inet6 static
      address 2001:db8:acad:2::1
      address fe80::1
      netmask 64
```

**/etc/sysctl.d/99-sysctl.conf**

```
 net.ipv6.conf.all.forwarding=1
```

**Python version:**

Python 2.7.9

## Windows 7

**Device configuration**:

| Base Memory: | Processor: | Video Memory: |
|---|---|---|
| 512MB | 1CPU | 27MB |

Table 26. Windows 7

The OS has been downloaded from: `https://developer.microsoft.com/en-us/microsoft-edge/tools/vms/#downloads`, installed on 24.10.2016.

**OS Version**:

Microsoft Windows 7 Enterprise, 32 bit, OS version 6.01.7601 Service Pack 1 build 7601

**Network Configuration:**

Network Auto configuration of IPv6.

**Linux Debian External Device**

**Device configuration**:

| Base Memory: | Processor: | Video Memory: |
|---|---|---|
| 1024 | 1CPU | 24MB |

Table 27. Linux Debian External

The device has been cloned from Linux Debian to test the ICMPv6 Redirect (Type 137) for the second Hypothesis.

**Cisco ASA Firewall**

**Model:**

Cisco Adaptive Security Appliance (ASA) 5505 Series

**Version:**

The version of the device and its component is visible in Appendix C.

## 4.2. Network Configurations

The network configurations (see 2 and 3) used to run the experiment are very similar and differ for a few details: the presence of the Windows box, and of the External device inside the second configuration. In addition the second configuration requires some changes in the addressing schema, by using an automatic configuration.

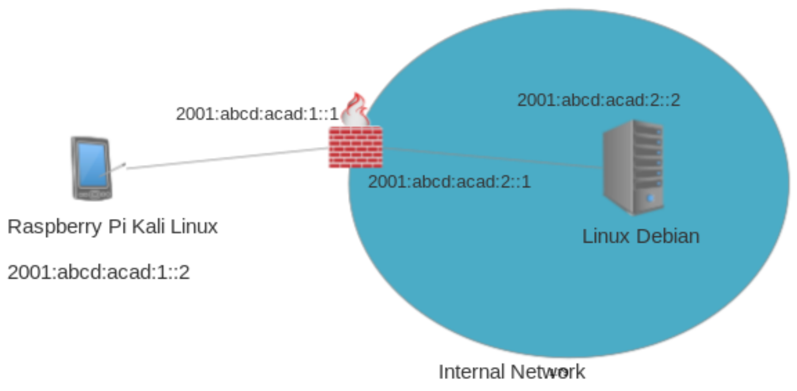


Figure 2. Network Configuration to test first hypothesis

In 2 is represented the configuration to test the first hypothesis (see Hypothesis 1). The context is given by an attacker who has gained a foothold on the internal network, but also by a disgruntled employee, whose aim is to exfiltrate data without authorization.

The network configurations are very simple and are not reflecting a production environment of a company. The reason behind this choice is to have a controlled environment, where it is possible to modify one variable at a time, which is a mandatory constraint to test the hypothesis.

Another choice in 2 is the use of a single internal device. There are two reasons behind that. The first is related to the PoC, which is written in python with a focus on a Linux environment (this topic will be developed later in this chapter). The second reason is related to the two processes, in the context of data exfiltration, which are both controlled by the attacker: the sending process is managed by the PoC in the Linux internal host, while the receiving process is running in the external host. Since both processes are not controlled only by the network stack implemented in the kernel of the devices, this research decided that a test using another device (the Windows host) would have been less interesting, while adding more complexity in writing the PoC.

The firewall, in both figures, is a general representation of a device with firewall services. During the execution of the experiment, the whole test set related to the hypothesis is run against each device and configuration. The difference between each configuration is discussed in detail later in this chapter, while the figure shows the common characteristics, the IPv6 addresses of the two interfaces, one linked to the internal network, and one to the ouside world.

In 3 is represented the configuration to test the second hypothesis (see Hypothesis 2).

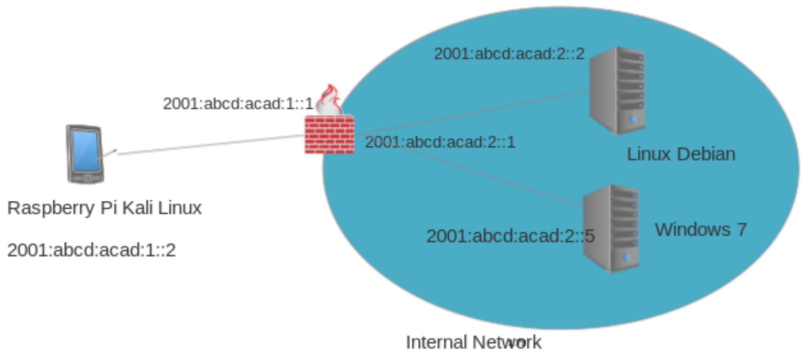


Figure 3. Network Configuration to test second hypothesis

In this scenario, the attacker controls the external device and want to test the possibility to use the internal network and hosts in an unauthorized way. The scenario, and hypothesis, is more oriented to test the implementation of ICMPv6 in different OSs, therefor the introduction of an additional device. The presence of only two OSs is another limitation in scope of this research. A Linux device may be the choice for a server offering services for an enterprise, and a Windows 7 device simulates an employee workstation. Further research should take into consideration a broader set of OSs.

**Network Configuration, from First to Second Hypothesis**

To test the second hypothesis it is necessary to accomplish some more steps. Since the Neighbor Discovery protocol is involved, the IPv6 address of the end devices, internal Debian Linux and Windows 7, is auto configured after receiving a Router Advertisement from the firewall.

While for the ASA Firewall this is the default behavior, the Debian Linux Firewall needs an additional package in order to send Router Advertisement, which could be installed as follows:

*apt-get install radvd*

After the package is installed, edit the configuration file in /etc/radvd.conf, as follows:

```
 interface eth0 {
      AdvSendAdvert on;
      MinRtrAdvInterval 5;
      MaxRtrAdvInterval 10;
      prefix 2001:abcd:acad:2::/64 {
            AdvOnLink on;
            AdvAutonomous on;
            AdvRouterAddr on;
      };
};
```

The process must be activated, after the device starts to act as a router (by forwarding packets), with the following command:

*systemctl start radvd*

## 4.3. Firewall Configuration

The configuration of the warden is a key point of this research. The firewall is the border network device that provide the line of defense of an enterprise network towards the external IPv6 world, where the IPv6 internal addresses are routeable and reachable.

The firewall configuration inside this research is also used as a way to simulate different requirements, budget, and expertise that an enterprise may have. These three characteristics must be carefully evaluated by an organization, and a balance between them must be found. The requirements are usually to have the maximum level of defense, and are evaluated against the budget of the enterprise. The configuration of the appliance is crucial, and requires the needed level of expertise, which, again, is strongly dependent on the available budget.

Finally, since this research's goal is to test the ICMPv6 protocol, the different configurations of the firewall must be chosen in a way that takes into consideration the boundaries of the test possibilities. This is accomplished by having four configurations, where two of them represents the boundaries of the test space. This concept will be developed next, inside each configuration description.

### Netfilter Open

This firewall configuration represents the lower limit in the test space, where packets are forwarded but not filtered.

As it is possible to see in G.1, the netfilter configuration enables IPv6 packets forwarding in the linux kernel, and explicitly accepts packets in the forwarding chain of the filter table.

This open configuration offers the possibility to reach two goals. It simulates a bad practice that may arise in the transition period, where the IPv4 protocol suite is the choice of an organization, but IPv6 is already active in the private network. The term simulation, here, is used on purpose, because the forwarding of IPv6 packets is not active by default. Nevertheless, it may stimulate the necessary awareness in the audience regarding the transition period and the need to deal with IPv6.

The second goal is to test the design and implementation of ICMPv6 by itself, without the additional filter of an organization's security practitioners. This configuration allows to test the protocol in a direction which is close to the aforementioned IPv6 Ready Project (see 2.3.1), in the contest of the second hypothesis. In addition, the configuration allows to evaluate the design of the ICMPv6 protocol and the fields in its messages in the contest of data exfiltration, which is related to the first hypothesis. The latter may be of secondary importance for the goals of this research, but, nevertheless, from the point of view of a network protocol's designer, could be interesting to think about how a protocol must rely on third party devices and configurations, like routers and firewalls, to avoid that some fields are used in a different way with respect to the semantic for which they have been created.

### ASA Default Configuration

The Cisco Adaptive Security Appliance is a device that may be the choice of small to medium organizations. When first fired on, it is possible to do some basic configurations to allow it to be operational by configuring and activating some basic features, like the IP/IPv6 address of the interfaces and their security level, the default being a security level of 100 for the inside interface, and a security level of 0 for the outside interface.

This kind of configuration may not be sufficient to protect the organization, and it is not in most cases, but it is a representation of a bad practice that leads to buy a professional, in some cases expensive, appliance, with the feeling that it is enough to protect the enterprise's business assets. This configuration simulate also the lack of the necessary expertise, both from a management and technical point of view. The management should evaluate the needs of both the appliance and the technical expertise to configure it.

The configuration file (see G.2) shows the configured parameters, with just few of them modified from the default: the IPv6 address of the two interfaces, the activation of ssh for management purposes, hostname, domain, and the encryption of the passwords. In order to activate IPv6 it is sufficient to configure the IPv6 address on the vlans associated to the network interfaces.

**ASA with ICMP module**

This configuration adds an important features with respect to ICMPv6: the activation of the ICMP module on the global policy, in order to allow deep packet inspection on the ICMPv6 packets. It is worth noting that the ASA configuration uses the term ICMP for both ICMP and ICMPv6.

In the case of this configuration, the management evaluated the needs of the appliance, but also the needed expertise to do the necessary configuration, which for this research considers an ICMPv6 only world. In other words, a production environment must take into consideration the services offered by the enterprise and add the required configuration, which involves the creation of access lists to determine the packets to allow in and out the internal perimeter.

The configuration file (see G.3) shows, under the policy-map global_policy section, the added ICMP module.

**Netfilter with Best Practices**

The last firewall configuration is represented by a Linux device with netfilter services. This choice represents the necessary expertise to configure a border network device to deal with ICMPv6, and, in the mean time, the use of an unexpensive device with an open source software to provide the filtering services. In terms of awareness, it means that the management may have chosen to invest in expertise rather then in the device itself, in a context where the budget has some limitations.

The configuration file (see G.4) has been developed starting from a script published by the CERT website [37], and modified to fullfill this research requirement using RFC 4890 (which also contains an example script)[33]. In addition, the original script has been thought for an host-based firewall, an end device, with rules applied to the input and output chains of the filter table. This research, instead, required a device for packets in transit, in addition to the packets with the firewall as destination. This involves the forwarding chain, where the best practices for the communication between the internal network and the rest of the world have been applied.

It is worth to mention this configuration as a contribution of this research. As it will be described next, the rules applied to the different chains allows to understand better some functionalities of ICMPv6, which may be usefull from a didactical point of view to introduce security practices in network classes.

**Ip6tables Rules (Netfilter Service)**

The script with best practices rules has comments that introduce each block of rules. An analysis of each block with the rules follows.

The first block contains variables related to the two network interfaces of the firewall, and the IPv6 addresses, which identify the internal and external network linked to them.

The second block allow to activate packet forwarding in the linux kernel. The command allows a temporary activation of the service, which last until a reboot of the OS.

The following block is the reset of the rules that may be present in previous configurations: it erases the rules and the user created chains, and clear the statistics of the packets. The statistics represents a very useful tool to understand, and verify, how the rules behave and which packets are blocked or enter a particular chain. The following command shows the statistics:

```
ip6tables -L -n -v
```

Then, ad hoc chains have been created. This step may not be necessary in terms of functionality, but it allows to better understand the organization of the code. This research point of

[37] `http://www.cert.org/downloads/IPv6/ip6tables_rules.txt`, accessed 22.4.2016

view is that readability and clear code is a security best practice, which should be applied not only in software engineering, but also in writing small script. The first two chains shows the network flow directions, which are mainly related to the two hypothesis. The two additional chains allows to establish an ssh connection to the firewall device, and to the attacking device, for monitoring purposes.

As the comment suggests, the next block applies the default policy in a whitelist way, which is dropping packets entering the chain by default, if there is not a rule matching the entry. The white list approach has advantages, but also drawbacks to take into considerations. In terms of security, this is the better approach, in contrast to the blacklist approach. The reason is that the blacklisting procedure allows to block, or filter, what is well known, which could be the signature of an attack, or an IPv6 address known to be the source of an attack. The problem arises for new attacks, or packets that the security practitioner is not aware of being part of an attack, which, with this approach, are forwarded. The whitelist approach, instead, denies the access for all but the known and authorized packets. It is the task of the administrator to provide the rules to allow the forwarding of the packets.

The whitelist approach requires a very good understanding of the network, and of the behavior of the protocols and of the services allowed on the network. Since all is denied by default, also authorized packets are blocked until not explicitly allowed. This may disrupt the network and break the availability of some services.

The following blocks define rules to apply to the input and output chains, the chains of the filter table to manage packets with the firewall network interface address as destination (input chain), or for packets that originate from the firewall device (output chain). This rules are needed for the correct operations of the local network segments, i.e. to allow protocol like the Neighbor Discover to work correctly.

The rules, applied against the forwarding chain that appear next, regulate packets with addresses which are valid in the local network only, like link-local and multicast addresses.

The script, then, defines the rules to enter in the aforementioned ad hoc chains: the source and destination addresses for the ICMPV6-TO-OUT chain, and the destination address for the ICMPV6-TO-IN chain. Both of them match the ICMPv6 protocol only.

The first analyzed rules regard packets originating from the internal network and forwarded to the outside. The ICMPv6 error message types are accepted and forwarded, since their function are needed for the correct operation of IPv6. The two informational messages are treated in a different way: Echo Requests are allowed but rate limited, while Echo Reply are droped. While the former rate limitation is not strictly necessary with today network bandwidth, the latter is a choice of this research, based on a possible organization's policy. Most of the examined documents, during this research, put the attention on the fact that, due to the huge address space in IPv6, it is no more necessary to filter Echo Request from the outside to deny network scan operations and the map of the internal network. This research is taking another approach, which follows the whitelist approach phylosophy. The functionality may be useful to test the reachability, from the outside, of some internal devices, e.g. a server in the DMZ. This functionality, however, may be needed during the initial configuration, or after some critical updates, but not during the everyday activity. It is the opinion of the researcher that it is a better practice to deny the forwarding of messages that are not needed for every day work, while it is possible to craft special configurations (in this case may be to allow one specific IPv6 source address to test reachability) to deal with special cases.

The remaining rules of this ICMPV6-TO-OUT chain are related to the Neighbor Discovery Protocol. Here the choice of this work is to comply explicitly with the RFC ([1]) and allow the forwarding of NDP messages only if the hop limit is equal to 255. This has the equivalent effect of denying the messages, since, in the forwarding process, the device decrement by one the hop limit, resulting in a value of 254, and therefor being droped by the policy. The last two rules allow to log the droped messages, and to explicitly drop the remaining, not matching, packets. The last rule is redundant because of the policy, and it is there just to show the last rule to apply in case the policy is to allow packets by default (blacklisting).

The last block of rules, to filter packets to be forwarded in the ICMPV6-TO-IN chain, reflects the choices of the preceding block. The ICMPv6 error messages are allowed to be forwarded. The informational messages are divided into the Echo Request, denied by the organization's policy, and Echo Reply, allowed as answer to an Echo Request from the internal network. The NDP messages, instead, are explicitly droped, which, as said, accomplishes the same effect previously described with the hop limit of 255, which is to deny the forwarding of the packet.

## 4.4. Proof of Concept (PoC) and Test Set

### PoC

The PoC for the experiment could be find at:

https://github.com/antekirtt/thesis/tree/master/python/code

It is possible to download a zip file with all the thesis material, or in alternative to use git:

```
git clone https://github.com/antekirtt/thesis.git
cd python/code/
```

**Requirements:**

The PoC has been written and tested for Linux Debian and Kali Linux.

The following libraries for python are required:

*scapy*:

Version 2.3.2_dev, retrieved on 31.08.16 and installed on all the devices that use the PoC.

*Note:* the version used in the Kali and Debian repositories is 2.3.2. This version has some bugs related to IPv6 that have been solved in the development version.

To download and install the library:

```
git clone https://github.com/secdev/scapy.git
cd scapy
python setup.py install
```

*bitstring*:

Installed on 31.08.16 in all the devices that use the PoC.

To install the library:

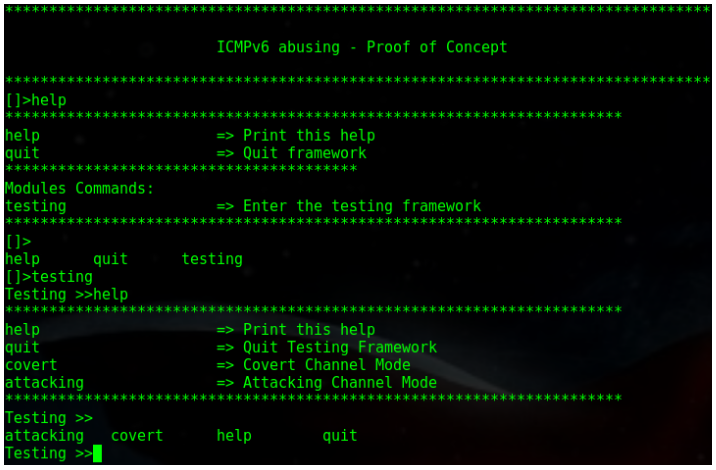
```
easy_install bitstring
```

The PoC is started at command prompt as follows (root privileges required):

./icmpv6 -i eth0

with the mandatory -i option defining the network interface to use.

The picture (see 4) shows the initial screen and some basic function, the help command with explanation of possible commands, and the command completion obtained by pressing the tab key twice.



```
********************************************************************************

                        ICMPv6 abusing - Proof of Concept

********************************************************************************
[]>help
**********************************************************************
help                    => Print this help
quit                    => Quit framework
****************************************
Modules Commands:
testing                 => Enter the testing framework
**********************************************************************
[]>
help      quit      testing
[]>testing
Testing >>help
**********************************************************************
help                    => Print this help
quit                    => Quit Testing Framework
covert                  => Covert Channel Mode
attacking               => Attacking Channel Mode
**********************************************************************
Testing >>
attacking   covert      help        quit
Testing >>
```

Figure 4. PoC initial screen

Help and command completion are available along all the sections of the PoC.

The PoC is divided into two main parts, each related to the test of each hypothesis.

## PoC - First Hypothesis

The PoC related to the test of the first hypothesis is contained in the file named allCovertTest.py. The main class inside the file, AllCovertTests, defines some variables and the main method “startSystem”. The variable “dataToExfilt” defines the second part of the data to be exfiltrated, which is the following: :

```
:this is the super secret data to exfiltrate to our attacking
    machine\n
```

The above data is concatenated with the name of each message type and field. For example, the complete data for Destination Unreachable type, and the use of the Code field to embed the data, is the following:

```
Destination Unreachable code :this is the super secret data to
    exfiltrate to our attacking machine\n
```

For each message type, a class is defined, and it is responsible to build the packets to send. In addition to the initialization, each class contains two kind of methods. The first is a set of methods, one for each field of the message, which follows the pattern "buildPacket" plus the name of the field. It is responsible to build each single packet using the functionalities of Scapy. The second is common to all the classes, it is called "execModule", and it is responsible to initialize the packets, and send them after the data has been divided to fit the field bandwidth. The last functionality is achieved using an helper class, which contains a family of methods to be used depending on the bandwidth, and on the way that Scapy manages internally some type of data (e.g. address fields).

One challenge of the functionality that allows to chunk the data to fit each field has been to find a data structure, and a methodology, that allows to send the data from one side, and rebuild the same data on the other side. Most of the variables representing the fields in Scapy are integers, which is a problem because integers are not represented with a fixed length. For example, the ascii decimal code for the character "c" is 99, while for character "d" is 100. The concatenation of the two, in case of two byte bandwidth, is 99100. Since the length is variable, on the other side would be more complicated to split it into two different bytes, and characters.

The algorithm used to achieve the functionality is the following (the example shows the methodology used with multiples of one byte bandwidth, for simplicity):

First, data has been divided into chunks accordingly to the field's bandwidth. Then, from each chunk, the single character is processed and transformed in its binary representation using the bitstring package. Each binary representation is added to the preceding one, with a resulting binary number which is a multiple of one byte. The binary representation is transformed back in an integer, which is sent.

On the receiving side it is possible to transform back the integer in a fixed length binary number, divide it into bytes, and obtain the ascii code wich is displayed as a character.

Another important file to test the hypothesis is named receiver.py. The Receiver class uses the functionality of the packet sniffer inside the Scapy library, filtering the packets by their IPv6 destination address. This could be considered the weekest part of the PoC: the packet sniffer itself, and the implementation to get the data, sometimes get some unwanted packages, leading to show some garbage on the screen. This is caused by the presence of network protocols data that the receiver is not able to filter appropriately, which may lead to the corruption of data. However, the result of the experiment is not affected by this issue, which is confirmed by intercepting the data using Wireshark[38], on the destination device, and by analyzing it. Nevertheless, an improvement of the receiver is needed in the evolution of the software.

### PoC - Second Hypothesis

The PoC to test the second hypothesis is different with respect to the first one, in terms of organization of the code and involved processes.

Since the simulation involves the control of just one process by the attacker, the sending process, the PoC reflects this condition. Another difference, as will be analyzed in the next section, is that each test must be performed individually, and the behavior examined in the end system target by other means. The configuration of the IPv6 address is no more required for this set of tests: each command is tied to a specific target. This choice is also related to the IPv6 auto configuration mechanism, because the generated address is more complicated to memorize and configure each time the target should be changed.

The commands to execute the experiment are located into two different file: *infoAndErrorAttacks.py* and *neighbDisc.py*.

Inside the latter there are also the commands to verify the internal attacks. They represents a step to define the actual tests, and are not part of the verification of the hypothesis.

Each message type has a unique source file and the functionality broadly reflects the following schema: one class per message, one execution test method per test, with a general method to build the required packet. For some messages, when the building of a packet changes in a significat way between executions, the execution method contains also the functionality to build the packet.

The involved IPv6 address are hard coded in the initialization method of each class, which is the place to modify the code to perform the experiment with different addresses.

[38] `https://www.wireshark.org/`, accessed 11.10.2016

**Test Set - First Hypothesis**

The complete test set for this hypothesis is visible in appendix (see 62).

The tables shows the fields used as test set. The rows are grouped by message type to show more clearly how many fields, for each type, are used for experimentation.

From the proposed test set there is a consideration that is important for this research: all the field of each message type have been considered for the set. It is worth to remember that RFCs defines the behavior of some processes that use the message types: for example, the unused field in Destination Unreachable and Time exceed messages, or the reserved field in the NDP messages, are often pointed out as fields to ignore, or zeroize by end devices or firewalls. Previously in this research, it has been specified the need to evaluate also the design of the network protocols, and not only the implementation. The unused and reserved field must be part of this evaluation, since, even though they may be there to reserve some space for future use, there may be other ways to accomplish future needs. The fields, in fact, are perfect candidates to be used for data exfiltration, because they allow to preserve the syntax and the semantics of the protocol [2].

**Test Set - Second Hypothesis**

The test set used to validate the second hypothesis is very different, in terms of reasoning and goals, from the one used to validate the first hypothesis. Instead of showing the set and the results in different tables, it is possible to see both of them in one table (see F.2).

The differences in reasoning and goals reside in the nature of the hypothesis and in the way that the results may be validated. First hypothesis is validated in a binary way, either the message representing the exfiltrated data arrives, and shows on the receiver process, or not. The second hypothesis, instead, implies a different kind of validation, because in the case that the tested ICMPv6 message arrives, the resulting behavior of the end system may be the one described by the RFC. In this case, before the validation of the hypothesis, it is necessary to understand the behavior of the end system when the packets arrive.

The table of the test set and results is organized in a different way for the two main type of messages: error and informational messages, and messages related to the ND protocol.

**Error and Informational messages**

The table is organized with four main columns. The description of the experiment shows some indications about the content of the test: wheather there is a manipulation of the source or destination IPv6 address, or if the value of a field has been modified to test extreme values. The table differentiates between different targets, and then between the four firewall configurations. The last column shows the behavior of the end system when the packets arrive, which also represents the result in the case that the behavior is not the expected one.

**Neighbor Discovery protocol messages**

The tested ICMPv6 messages defined by the ND protocol have different scope, with respect to error and informational messages: the scope is link local, the network segment which boundaries are defined by a router (phisical device or functionality). This difference in scope led to a different kind of thinking in order to build the test set: instead to try to directly test the hypothesis, it is worth to manipulate the messages and build tests to be performed in the internal network and then select the successful one to be performed from the outside network in order to validate, or negate, the hypothesis.

The tables (see F.2) reflects this kind of reasoning. Each table related to the ND protocol is divided into two parts. While the second part shows the same columns as for the other ICMPv6 messages, the first part shows a brief description of the type of attack performed internally, and the results for each target. Each test may then be selected for the actual test.

# 5. Experiment

The experiment of this research has been designed to test the research question and to validate, or negate, the hypothesis. The third chapter (see 3) defines a main research question and hypothesis, which could be divided into two more specific research questions and hypothesis, one for each network flow direction. That is, from the internal network to the external network (see Hypothesis 1), and from the external network to the internal network (see Hypothesis 2).

This chapter is organized to show how to run the experiment. A different network configuration has been designed to test each hypothesis. Each configuration includes an external host, a firewall device, and an internal Linux device. In order too test the second hypothesis, two devices have been introduced: a Window device for the whole test set, and an External (Debian Linux) device to test Redirect messages.

Since one of the goals of this research is to validate, or negate, the hypothesis against different firewall services and configurations, also the device changes accordingly: a Cisco ASA with two different configurations, and a Linux device also with two configurations.

It is worth to underline how this research uses the term “experiment” in different ways. Along this document, the term is often used in a general way, indicating the experimentation part of this research. But, as it has been mentioned before, an experiment is also the run of a single test against a firewall configuration and an internal host, in the context of the validation of an hypothesis. This is because the hypothesis may be validated just for that single test, while it is negated for another test, firewall configuration, or host.

## 5.1. First Hypothesis

In order to carry the experiment for the first hypothesis (see Hypothesis 1), it is required to build the appropriate network configuration (see 2).

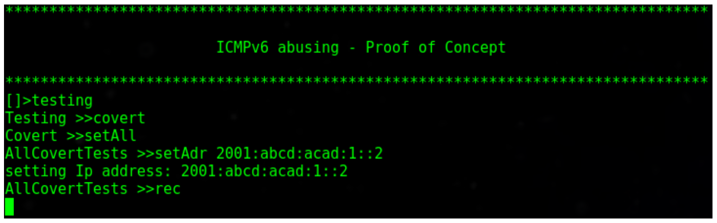
### Experiment Setup and Run

In order to setup the experiment it is necessary to prepare the devices as described in 4, and the network topology (see 2). In addition, the PoC must be downloaded [39] and installed on the devices with the correct dependencies, also described in 4.

The Linux Debian device, in the context of the experiment, is owned by the attacker with root privileges. The Raspberry Pi Kali Linux (RPKL) device is also controlled by the attacker.

The first step to run the experiment is to start the receiver on the RPKL device, with the command:

```
./icmpv6.py -i eth0
```



```
********************************************************************************
                    ICMPv6 abusing - Proof of Concept
********************************************************************************
[]>testing
Testing >>covert
Covert >>setAll
AllCovertTests >>setAdr 2001:abcd:acad:1::2
setting Ip address: 2001:abcd:acad:1::2
AllCovertTests >>rec
```

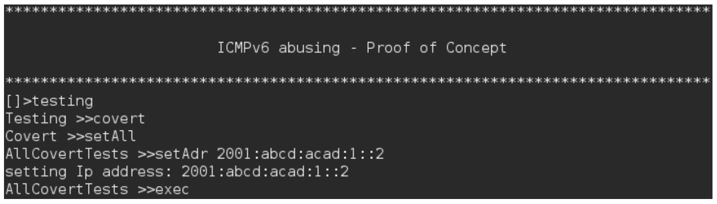
Figure 5. The screen of the receiver

As it is possible to see from the figure (see 5), the only piece of configuration is the IPv6 address of the receiver, the RPKL device.

The second step involves to start the PoC on the Linux Debian device. To start the PoC the same command is issued:

```
./icmpv6.py -i eth0
```

From the figure (see 6), it is possible to see the commands and the configured IPv6 address.



```
********************************************************************************
                    ICMPv6 abusing - Proof of Concept
********************************************************************************
[]>testing
Testing >>covert
Covert >>setAll
AllCovertTests >>setAdr 2001:abcd:acad:1::2
setting Ip address: 2001:abcd:acad:1::2
AllCovertTests >>exec
```

Figure 6. The screen of the sender

The results of the experiment, with all the details, will be discussed in the next chapter.

[39] https://github.com/antekirtt/thesis

## 5.2. Second Hypothesis

The second hypothesis (see Hypothesis 2) requires to build the appropriate network configuration (see 3).

### Experiment Setup and Run

In order to setup the experiment it is necessary to prepare the devices as described in 4. In addition, for the first and fourth firewall configurations (Debian Linux with Netfilter services), it is required to perform the modifications described in **??**. The PoC must be downloaded [40] and installed on the devices with the correct dependencies, also described in 4.

The Raspberry Pi Kali Linux (RPKL) is controlled by the attacker and it is used to perform the tests to validate, or negate the hypothesis. In addition, in order to perform the internal tests related to the ND protocol, it is possible to use the Debian Linux or the Linux Firewall with open configuration (see G.1), which requires to download and install the PoC in these devices as well.

The best procedure for the experiment is to first use the Linux Firewall with open configuration, since this kind of configuration grants the generated packets to reach the internal network. In addition, for many test it is useful to monitor the generated traffic, in both directions, from the firewall using an instance of Wireshark, which is installed on the device.

In order to start the experiment, execute the following commands:

```
./icmpv6.py -i eth0
testing
attacking
```

This leads to the part of the PoC used to test the second hypothesis. The PoC is then divided into the two groups of attacks, the first related to error and informational messages, the second related to the ND protocol.

```
setInfoAndError
   OR
setNeighborDiscovery
```

[40] https://github.com/antekirtt/thesis

### Error and Informational

After entering the informational and error section, it is possible to see the commands to perform this part of the experiment. The commands for this section are available in Tab. 63 from appendix E.

#### Destination Unreachable Tests

The first two tests send, in sequence, a packet with each code and type 1. A wireshark instance is used to monitor the network on the Linux Firewall. This general purpose kind of test may be useful to observe the behavior of end systems when a Destination Unreachable message arrives, in particular to assess if the end systems generate new packages as a response to the messages.

Tests from 3 to 16 are specific for each message code. The goal is to test each code using either the Attacker IPv6 address as source, or the spoofed firewall address. Wireshark is used to monitor the traffic in both the Linux Firewall and the end system target. From each end system, the test includes an http request to the attacker device, an Echo Request, or both, in order to verify if changes occur in the conversation after the reception of the ICMPv6 packets.

Tests 17 and 18 are used to assess the behavior of the end systems against unknown codes.

Tests 19 and 20, instead, are related to the new Length field. The purpose here is to evaluate the end system against a field which is implemented in the used version of Scapy, but not present in RFC 4443 [6]. Since the Scapy version of the experiment is the development one, it is possible to find some fields which are not yet approved by the IETF. This should not be the case of the implementation of ICMPv6 in the end systems, nevertheless it could be worth to evaluate to which point implementations in OSs (or applications using ICMPv6) are following the directives.

#### Packet Too Big

Tests from 21 to 26 manipulates the MTU field (minimum for IPv6 is 1280), by sending values which are greater (21-22), smaller (23-24), or in average (25-26) with respect to the minimum required.

Tests 27 and 28 have the aim to evaluate the behavior of end systems with unknown codes.

**Time Exceeded**

Tests 29 to 32 use the spoofed IPv6 address of the firewall to generate code 0 and code 1 messages. Effects on end systems and network traffic are observed using Wireshark.

Tests 33 and 34 are performed against unknown codes, while 35 and 36 are using the RFC draft Length field with values varying from 0 to 255.

**Parameter Problem**

Tests 37 to 42 use respectively code 0 to 2 with the spoofed IPv6 address of the firewall, with Wireshark to monitor traffic, while tests 43 and 44 are performed against unknown codes.

Tests 45 to 48 manipulate the pointer field of the message. The first two tests use, in sequence, low values (0-1024), while the last two use high values from the 32 bit field. The aim is to evaluate the end systems against CPU and memory consumption using Task Manager (windows) and the top command (linux).

**Echo Request**

The aim of the tests for this message type is to evaluate the behavior of the target with respect to Reachability and Neighbor Cache. With tests 49 and 50, the source address is generated to be an on-link address, while the destination is the target end system. Tests 51 and 52 perform the same tests, but using the generated address as destination, and the target end device as source. In order to evaluate the results is possible to use the following commands:

```
(Windows) netsh interface ipv6 show neigh
(Linux) ip -6 neigh
```

**Echo Reply**

Tests 53 and 54 send packets to the target systems with the source spoofed IPv6 address of the firewall. The goal is to observe the behavior of the target when it receives an Echo Reply with a fake Echo Request in the payload.

Test 55 sends the same packet as test 53, but in this case the host firewall of the target has been disabled to observe the behavior of the end system.

## Neighbor Discovery

The commands for this section are available in Tab. 64 from appendix E. Evend though are not part of the experiment to test the hypothesis, also the command to perform internal test are available in the PoC, as in Tab. 65. The internal tests have been used as a first step to then define and select the actual tests, and will be briefly discussed later with the results of the experiment.

### Router Solicitation

No tests have been selected for this message type. The reason is that internal tests did not suggest a way to use this type as an attacking vector.

### Router Advertisement

The tests for this message type are divided into two main categories: the first tests the injection of network prefixes, while the second tests the modification of the MTU value on target devices.

Tests 56 to 61 have Windows 7 as target, while tests from 62 to 67 have Debian Linux as target. This series of tests are executed sequencially using one command for each end system, because each of them tries to inject a different prefix. Tests variation includes different source addresses (Firewall, Debian Linux, and Windows 7), and different values for L and R flags. To evaluate the behavior it is possible to use *ifconfig* (linux), or *ipconfig* (windows).

Tests 68 and 69 inject an arbitrary MTU value on the target. The verification of the result can be done as follows: target end device as source. In order to evaluate the results is possible to use the following commands:

```
(Windows) netsh interface ipv6 show subinterfaces
(Linux) cat /proc/sys/net/ipv6/conf/eth0/mtu
```

### Neighbor Solicitation

Test 70, with Debian Linux as target, has been selected after the internal test showed that the target started to send Neighbor Advertisement to itself. Source and destination IPv6 addresses are the same.

### Neighbor Advertisement

Test 71, performed against Debian Linux, uses an Echo Request with a spoofed internal IPv6 address to induce the target to start the Reachability process, which may be completed by sending a Neighbor Advertisement after the target sends a Neighbor Solicitation. Also this test has been selected after a successful internal attack. The behavior of the target could be observed using Wireshark on the target and the command *ip -6 neigh*.

### Redirect

For this tests is necessary to add the External Device, as in Fig. 3. Tests 72 and 73 aim to verify the possibility to perform an half Mitm attack, by redirecting the target request to the attacker. Use Wireshark on the attacking device to verify the tests.

# 6. Results

This chapter exhibits the results of the experiment of this research. It is organized to introduce first how the results are presented. Then, the chapter is divided into the two sub-hypothesis (see Hypothesis 1, and Hypothesis 2) to show for each of them what are the outcomes of the experiment.

The results related to the first hypothesis are presented by introducing some general considerations about the experiment, and by describing how to read the table in appendix that shows the outcomes. After that, for each firewall configuration, this research will explain the results and discuss the differences between them and the expectations before performing the experiment. Finally, the result of each experiment against a particular firewall configuration is compared to the others, and the hypothesis is eventually validated or negated.

## 6.1. First Hypothesis

In order to test the first hypothesis, this research defined a test set (see D). The PoC has being written to validate or negate the hypothesis using each test of the set against each warden configuration, as previously described.

It is worth to remember that, for the experiment to be valid, only one variable at a time must be modified. This variable is represented by the specific firewall configuration, while the other components of the experiment remain constant. For this reason, also the results are presented, along this chapter, in sections named after the firewall configuration that generated them (see 4.3).

The results of the experiment are visible in appendix F.1.

The table with the results is organized by enumerating the tests in the first column to allow a direct reference to each test. Tests from number 1 to 10 included represents the ICMPv6 error messages. Tests from 11 to 18 represents ICMPv6 informational messages. Tests from 19 to 44 use ICMPv6 messages defined by the Neighbor Discovery protocol. Grouping the tests is usefull because each group of messages depicts a different broad functionality of the ICMPv6 protocol, and it reflects some expectations about the results of the experiment, which, as it is possible to see, and it will be described next, it is not always the case.

The second and third columns represents respectively the ICMPv6 message type and the field of the message used to exfiltrate data in the covert channel.

The fourth column shows the bandwidth obtained by using the particular covert channel offered by the field, in terms of bytes for each sent packet. The value of the bandwidth is static and it has been measured taking the amount of bits that a field can carry. This value is only theoretically correct, because there is no mechanism in the PoC to manage the retransmission of a lost packet. Nevertheless, this data is interesting in the evaluation of the covert channel and to analyze the possibility that a covert channel will stay undetected for longer: channels with low bandwidth requires more packets, given the same amount of data, for a successful exfiltration, which means a higher chance to be detected.

The fifth column depicts the firewall configuration applied for the test set. It is divided in four sub-columns which represents each configuration. Each sub-column, numbered after the configuration categories defined in 3.3, shows if that specific test validates or negates the hypothesis for that specific configuration.

**Netfilter Open**

The tests against this configuration produced the expected results and validate the first hypothesis for this specific configuration (see F.1).

The Netfilter Open could be seen as the trivial case. The covert channel is dependent from its ability to preserve the syntax and the semantics of the message [2], and from the ability of the wardens, along the path to the destination, to catch each modification of those properties. The triviality of this part of the experiment is given by the fact that the warden has no knowledge of the protocol, thus it is unable to detect such modifications. Nevertheless, in the context of the experiment, the tests allow to assess the following:

- Exploit the possibility for further investigations. In the case that the tests shows the presence of covert channels, it is possible to proceed by investigating more restrictive configurations.
- Analyze the presence, or absence, of covert channels in the ICMPv6 protocol by itself, and eventually broaden the discussion to network protocols in general.

The table with the results shows that this configuration allows to use a covert channel to exfiltrate data with all the fields of each message, without exceptions.

The experiment to test the first hypothesis suggests that it is useful to proceed by investigating more restictive configurations. Furthermore, since the configuration represents the trivial case, the conclusion of this research is that the ICMPv6 protocol allows the presence of covert channels. In other words, without an active warden which is capable to detect them, the ICMPv6 protocol is not able to prevent covert channels that use its fields to transport information. This conclusion will be further analyzed later inside the comparison of the results.

### ASA Default Configuration

The results of the experiment with the ASA with the default configuration was quite different from the expectations of this research. The table (see F.1) shows that the ASA has been able to block the covert channels embedded in the ICMPv6 error messages. Instead, covert channels using ICMPv6 informational messages, and messages defined in the context of the Neighbor Discovery protocol, validate the hypothesis for this kind of tests and this configuration.

The expectation of this research was quite the opposite, and it was based on the best practices that came out from the research, and applied to the fourth configuration. The ASA firewall is a dedicated device, which must be configured accordingly to the needs of an organization and its network, and services. In fact, a best practice regarding devices, or applications, is to never leave them in the default configuration, which, in the best case scenario, is something that every malicious actor can guess and try to bypass with public information. Nevertheless, this research's opinion is that the expectation after buying a device like the ASA firewall, is to have a device with minimal operation capabilities with the default configuration, which requires then to be conformed to the needs of the company for what is related to specific services. In other words, in terms of a common protocol like ICMPv6, the expectation is to have something as close as possible to the best practices.

From the table of the results:

1. tests from 1 to 10 included negate the hypothesis
2. tests from 11 to 44 validate the hypothesis

As mentioned before, tests from 1 to 10 refer to the error messages of ICMPv6. The message types are: Destination Unreachable, Packet Too Big, Time Exceeded, and parameter Problem. These messages are used by the network to communicate that there are issues of different kind in the delivering of a packet. For example, the Packet Too Big type uses the MTU field to communicate to the source of the packet that some link, along the path to the destination, has a Maximum Transmission Unit that is lower than the one that connects the origin to its first hop. This kind of information is important for the communication to work properly, and should not be blocked by the device.

On the other side, the ICMPv6 messages defined by the Neighbor Discovery protocol, Router Solicitation, Router Advertisement, Neighbor Solicitation, Neighbor Advertisement, and Redirect, are messages employed inside a network segment, inside the LAN, and are not supposed to travel beyond a firewall, or a router. In fact, the best practices state to filter such packets with a Hop Limit lower than 255. This kind of messages must not leave the internal network and must be blocked at the boundaries of the network segment.

**ASA with ICMP module**

This configuration adds the deep inspection of ICMPv6 messages by activating the ICMP module.

The table (see F.1) depicts the results of this set of experiments. The hypothesis is negated by tests from 1 to 11 included, and from 15 to 18 included. For test 12,13,14, and from 19 to 44, instead, the experiment validate the hypothesis and allow the creation of covert channels to exfiltrate data.

The expectations of this research, for this set of tests and configuration, was sightly different compared to the previous one. Since the covert channels have been created by modifying the semantics of the fields, with the activation of the deep packet inspection for ICMPv6 messages (the module is called icmp for both ICMP and ICMPv6) this research expected the negation of the hypothesis for all the ICMPv6 message types. Instead, the experiment validates the hypothesis for the Echo Request type, using Identifier, Sequence Number, and Data fields, and for all the messages related to the Neighbor Discovery protocol.

The expectations about the experiment with this configuration were built upon a wrong interpretation of what the deep packet inspection for ICMP and ICMPv6 is able to do. In fact, the module enables the firewall to be stateful with respect to the ICMPv6 protocol, which is by nature stateless. By adding a session to the traffic, the ASA allows the returning traffic, generated by ICMPv6 messages, to pass through the ASA and return back to the host that generated the initial messages. This is the case for an Echo Reply, which return traffic is generated by an Echo Request. In fact, the module allows to open a session only for an Echo Request and not for other type of messages, which are not supposed to expect a return traffic.

The session allows the device to block the covert channels related to the fields of the Echo Reply message, as shown by the results. This is because there are no valid Echo Requests that opened a session and allowed for the trafffic to come back. The ASA searches for an open session, which is not found, and the packet is dropped. From the results, instead, only the covert channel which uses the Echo Request code field has been blocked. The opinion of this research is that this is due to the fact that the ICMP module is looking for a valid code in order to open a session, but it could not find one, since the only semantically valid code for an Echo Request is 0. This opinion however needs further investigations, and a new iteration of the experiment.

The following command, issued on the ASA cli, can be used to debug and better understand the applied rules:

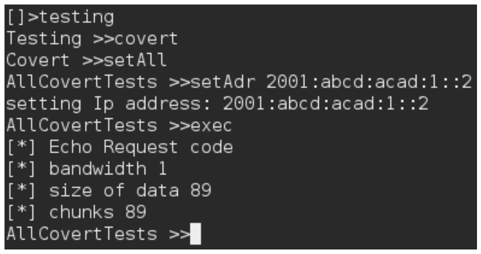
```
packet-tracer input inside icmp 2001:abcd:acad:2::2 128 0 2001:
    abcd:acad:1::2 detailed
```

the command shows in detail when and at which phase a rule may allow or drop a packet. It takes into consideration packets in the input of the inside interface, related to the icmp protocol (ICMP or ICMPv6), which arrived from the given source address. The ICMPv6 message has type 128 (Echo Request) and code 0. The command comfirms that a packet with the given characteristics is allowed to pass the firewall. Test number 11, instead, changed the semantics of the field, which results in a drop of the packet. Another command useful for the investigation is the following:

```
show asp drop
```

the command shows the statistics of the dropped packets (to reset the statistics use the *clear asp drop* command).

To confirm the aforementioned opinion, this research decided to slightly modify to PoC, by commenting all the tests, but the one that uses the Echo Request code field. After clearing the statistics, the experiment is performed.



```
[]>testing
Testing >>covert
Covert >>setAll
AllCovertTests >>setAdr 2001:abcd:acad:1::2
setting Ip address: 2001:abcd:acad:1::2
AllCovertTests >>exec
[*] Echo Request code
[*] bandwidth 1
[*] size of data 89
[*] chunks 89
AllCovertTests >>
```

Figure 7. Experiment iteration

The proposed command confirmed the assumption:

```
ASA-EXP(config)# show asp drop

 Frame drop:
  ICMP Inspect bad icmp code (inspect-icmp-bad-code)       89
```

The test sent 89 packets in order to exfiltrate data, which are inspected by the ICMP module and dropped because of a bad code. In this case, the ICMP module is able to negate the first hypothesis and prevent the covert channel.

Tests from 19 to 44 follows the same reasoning depicted in the previous configuration, that is, the messages defined by Neighbor Discovery must be blocked inside the broadcast domain.

### Netfilter with Best Practices

The table (see F.1) shows the results of the experiment with the Netfilter configuration using best practices. The results fulfills the expectations of this research and validate the first hypothesis for tests from 1 to 14 included. Test from 15 to 44, on the other hand, negate the hypothesis.

The result is fully expected because the best practices, which led to this configuration, clearly define which are the functionalities that a network needs in order to work properly. In addition, the applied best practices divide the functionalities between those that are required by the internal network, and those that are required for an end-to-end communication. For both of them, as described in 4.3, the rules to allow or drop an ICMPv6 message are applied.

Tests from 1 to 10 included use ICMPv6 error messages. Those messages are required to communicate that the network traffic encountered some kind of error, which may disrupt or slow down the communication. The results were expected because, even though the configuration follows the best practices, the device has not enough knowledge of the syntax and semantics of the protocol, which are the properties to which a covert channel must adhere in order to stay undetected. With the test, the syntax is respected, but the semantics change completely: the values of the fields are meaningless for the message. For example, a Time Exceeded code must always have values between 0 and 1 included, while the covert channel that uses that field modifies the value accordingly to the data it needs to transport.

Tests from 11 to 18 included represent the ICMPv6 informational messages. Here the hypothesis has been validated by the Echo Request tests, and negated by the Echo Reply tests. In this case the difference is caused by the application of best practices, but by an hypothetical internal policy that a company may have. The policy states that the organization want to be able to test external connectivity and allow return traffic under the form of an Echo Reply. On the other side, the policy grants that an external entitiy would not be able to perform aliveness test, and for this reason the return traffic is dropped.

Finally, since NDP is required on the internal network only, ICMPv6 messages defined by this protocol are dropped at the boundaries. This negates the hypothesis for tests from 19 to 44.

The Netfilter configuration with best practices is part of the contribution of this research, but it is not enough to detect all the covert channel, as shown by the results. This configuration is able to filter the messages accordingly to the best practices, but in order to disrupt also the covert channels that use the required ICMPv6 messages and fields, another kind of mechanism is needed. This mechanism is the deep packet inspection, which is based on a deeper knowledge of the syntax and semantics of the protocol. The mechanism may be offered by IDS and IPS services like Snort [41], or Suricata [42]. These services are not in scope with this research and will not be further analyzed.

## Considerations about the Results

The experiment related to the first hypothesis led to some considerations which are specific to each particular firewall configuration. Inside this section, instead, this research explores some

[41] `https://www.snort.org/`, accessed 19.10.2016
[42] `https://suricata-ids.org/`, accessed 19.10.2016

general considerations regarding the ICMPv6 protocol in relationship with covert channels.

It is useful to present again the first hypothesis:

*It is possible to manipulate the ICMPv6 protocol to exfiltrate data without authorization from an internal network to the outside and traversing a border network device*

The experiment shows that the hypothesis is validated by some tests and configurations. It is useful at this point to step back and try to evaluate the protocol itself, by keeping away for a moment the last part of the hypothesis, the one that includes the border network device. The trivial case, the one with the Netfilter Open configuration, may represent this case, where the only device's task is to forward packets.

As shown before, this special case validates the hypothesis for all the messages and fields. This result leads to the following question:

*the presence of covert channels in the ICMPv6 protocol is specific to the protocol itself, to the way it has being designed and implemented, or this presence could be extended to other networked protocols?*

It is the opinion of this research that this presence could be extended to other protocols, as far as the protocol has a mechanism that allows it to be transported to the destination. For ICMPv6, this mechanism is granted by the IPv6 protocol. On the other side, a trivial example is the destination address field of the IPv6 protocol: since the field is used to route the packet, a hypothetical covert channel that use this field will disrupt the communication, and the covert channel itself. Other protocols must be tested to confirm that they could be used for a covert channel, but if they rely on other protocols to be delivered, there is a great possibility that they are vulnerable as well, in all or some of their fields.

As the experiment shows, each field could be used for a covert channel. The nature of this particular configuration depicts the fact that the ICMPv6 protocol is vulnerable for this kind of misuse of a network protocol. The defense, in fact, is delegated to other device, like active warden, or protocols.

From an implementation point of view, the specification in RFCs do not explicitly mention the behavior of intermediate devices with respect to ICMPv6 packets. The specifications regard only end systems and the behavior of the sender and receiver of the packet. In the validation of the Neighbor Discovery protocol, the RFC specifies that a packet whose IPv6 Hop Limit field is less than 255 must be silently discarded [1]. The directive refers to end systems, which could also be routers or firewalls, if the packet is originated from, or directed to it. Nevertheless, this directive may be an implicit suggestion for a defense mechanism on intermediate devices.

From a design point of view, there are some consideration that are worth to underline. First, the presence of a covert channel is in the design of the protocol itself. The opinion of this research is that this is common and acceptable, because a network protocol must adhere to its requirements only, and any addition, in terms of functionality, could introduce complexity and new vulnerabilities. Other protocols must be designed and implemented to deal with other kind of functionalities. A simple example is the IPv6 protocol, which supply the transport mechanism. In the case of ICMPv6, a security mechanism to filter manipulated packets is demanded to the active wardens with knowledge of the syntax and semantics of the protocol.

The second consideration is related to the unused and reserved fields. Even though their presence could be justified by the needs to maintain a consistency in the lenght of the header, or for future use, these kind of fields may be considered overdesign. As mentioned before, a covert channel must be consistent to the syntax and semantics of the field to stay undetected. By introducing such fields in the design, the ICMPv6 protocol defines fields semantically meaningless, which gives less weapons to detect the covert channel to those intermediate devices to which the security of ICMPv6 is delegated.

Another aspect that is worth to analyze is the behavior of the active wardens in relationship with covert channels. The Netfilter configuration with Best Practices represents a service that must be completely configured accordingly to the needs of an organization, which involves an analysis of the requirements, and an understanding of the security implications that a choice may produce. In this case, the persons involved in the process needs to be aware of what a specific configuration is able to do. The experiment shows that following the best practices is not enough to block all the covert channel, and this research suggests the use of other services. This is a contribution in terms of security awareness.

The configuration of the ASA firewall, instead, may introduce some issues, depending on the approach taken by the management and security personnel. The experiment shows some differences between the expectations of this research and the actual results. The configuration applied to the device by an organization may follow their expectations as well, with the consequence to be vulnerable to data exfiltration using a covert channel. The default policy of the ASA firewall allows traffic to travel from an interface with high security level to an interface with lower secuirty level, but not the opposite [43]. This policy may be seen as an asymmetric evaluation between the risk that a malicious actor try to break the company's defenses from the outside network, and the risk that the same malicious actor has already gained access and try to exfiltrate data. Since one of the goal of this research is to produce the necessary awareness, this asymmetry must be taken into consideration.

## 6.2. Second Hypothesis

In order to test the second hypothesis, this research used different approaches. First at all, it was not possible to apply a systematic methodology like during the test of the first hypothesis. The reason is that during the previous tests, the fields of the protocol have been manipulated and used in a complete different way with respect to their semantics. This has been possible because the sending and receiving processes were both controlled by the attacker. The fields under test acquired importance with respect to the ability of the active warden to detect changes in the syntax and semantics of the field. For the second hypothesis, instead, the attacker is able only to send the message, while the manipulated packet may have an effect on the implementation of ICMPv6 at destination, which is not under control of the attacker. In this context, in addition to the active warden, the validation of the hypothesis depends on the enforcement of the rules defined by the RFCs on the target OSs. These rules vary in a sensible way, depending on the purpose and scope of the message. The purpose is different for example between informational and error messages, which have different scope with respect to messages defined by the ND protocol. Because of this difference in scope, messages defined by the ND protocol have been tested inside the internal network first. Then, in case of successful attack, a specific test have been selected to test the hypothesis from the external network.

The results of the experiment are visible in appendix F.2, and will be discussed next. Since the approach has been different, also the presentation of the results will follow this difference. First, this research will present the table with the results. After that, the results will be

[43]http://www.cisco.com/c/en/us/td/docs/security/asa/asa91/configuration/general/asa_91_general_config/interface_complete_routed.html, accessed 22.10.2016

disclosed from the point of view of each message type. Inside each message's section are also discussed the effects of a test on the target, and briefly the behavior of the active warden. The presentation of the effects will take into account mostly the configuration of the firewall with Netfilter Open. This configuration allows the packets to reach the target, evaluate potential changes, and assess the implementation of ICMPv6 in the target OSs, which may lead to a validation of the hypothesis for this configuration. Finally, a broader discussion will analyze the firewall configurations and the impact that the firewall has with the ICMPv6 protocol used to attack an internal network from outside.

The results are organized into tables, one for each message type. The tables for ICMPv6 with types 1 to 4, 128, and 129, have the same columns. The first column shows the test number. The second includes a brief description of the test, or its specification, e.g. the code of the message, indications about source and destination, or the nature of the payload. Then, one column shows the target OS, and one, for each firewall configuration, points out if the packet successfuly traveled through the warden. Finally, the target's behavior when the packet arrives, and the validation method used, when it is different from an observation with Wireshark.

The tables related to the ND protocol, instead, include the test selection in their first part. First column shows if the test has been selected, followed by the description of the attack. Then, after the target indication, the last columns exhibits the results of the attack and a brief explanation. The second part of the table follows the same structure as the one described for informational end error messages.

**Destination Unreachable**

The set of test related to the Destination Unreachable message negates the second hypothesis.

The first tests sent, in sequence, a packet with each known code. The goal was to verify if the target reacted after reception of the message in violation of the processing rules defined by the RFC [6].

Tests from 3 to 16, instead, tested each message code with different source addresses and payloads. An instance of Wireshark inside the target showed no new packets generated as a reaction to the tests. The verification included an http, an echo request, or both, to verify potential effects on the traffic new network traffic.

The last tests flooded the target using unknown codes, and different values for the length field. It is interesting to note that the latter is not even considered by the instance of Wireshark as a valid field. In fact, this new field use part of the official unused field, and Wireshark sees the modification of the value as part of this field.

The results show also a symmetry in the evaluation of the firewall: both Netfilter Open and Netfilter with Best Practices allowed the packets, while the two configurations related to the ASA firewall blocked them. While the first two are expected results, the configuration of the ASA deserve another consideration. The default behavior of the appliance is to block the traffic from a network with a lower security level to a network with an higher security level. This behavior may break the communication between hosts and is against the best practices. From a security point of view, this research thinks that this approach is reasonable and follows a white list methodology, where the administrator must configure access lists for the known services. Nevertheless, this is a confirmation that the default configuration is not enough for this device. A company must be aware that the ASA, without an investment in the necessary expertise for the configuration, will introduce new issues.

**Packet Too Big**

The set of test related to the Packet Too Big message negates the second hypothesis.

Since the MTU field represents the Maximum Transmission Unit of the next hop, the tests aimed to verify the possiblity to induce the target to fragment packets, or to send packets too big for the link. After the packets have been received, the validation of the test implied the creation of Echo Requests with different sizes. This is possible using the following commands, respectively for Windows 7 and Debian Linux:

```
ping 2001:abcd:acad:1::2 -l [size]
ping6 2001:abcd:acad:1::2 -s [size]
```

Wireshark showed no evidence of changes in the network traffic started from the targets. One possible reason for the failure is a lack on matching the process that would cause the Packet Too Big message, which the target should infer from the payload of the message.

The last tests flood the targets with messages with unknown codes. Also in this case there is no evidence on effects on the targets.

In terms of firewall configurations, the results underlined the same symmetry seen with the previous message: configurations with Netfilter allowed the packets to enter the network, while configurations with the ASA blocked the packets.

**Time Exceeded**

The second hypothesis has been negated also for the tests related to the Time Exceeded message.

Packets sent first with code 0, and then with code 1, produced no evidence of effects on targets, which does not generate additional network traffic after reception of the packets.

The last tests, flooding the target with unknown codes and different lenght values, do not produce evidence of changes on the target. The length field, also in this case, has been interpreted by Wireshark as part of the unused field.

As for the other message type, firewall configuration with Netfilter Open and Netfilter with Best Practices allowed the packets to travel across the device, while the two ASA configurations blocked the packets.

**Parameter Problem**

The test set related to the Parameter Problem message negates the second hypothesis.

Tests from 37 to 42 sent messages with valid codes, and the firewall spoofed IPv6 address as source, to the two targets, which showed no evidence of changes. Also, no new ICMPv6 messages have been generated in response to the tests. The same results occured by flooding the target with unknown codes.

Tests from 45 to 48 also produced no evidence. The idea behind these last tests was to verify the behavior of the target with different values of the pointer field, which represents a reference, inside the payload, of the octet of the invoking packet where an error caused the generation of the message [6]. By sending values between 0 and 1024 first, and then high values closed to the 32 bit upper bound, this research wanted to verify possible anomalies in the CPU and memory utilization on the target. In order to verify this behavior, on Windows system has been used the Task Manager and Resource Monitor, while for Linux the command Top.

The experiment depicted the same symmetry previously seen, with Netfilter configurations allowing the packets on the internal network, and ASA configurations blocking them.

**Echo Request**

The test set related to the Echo Request message validates the second hypothesis for the Netfilter Open configuration.

The tests used a 16 bit value range to generate as much spoofed IPv6 addresses with the internal network's prefix, and flood the target. The generated addresses have been used as destination addresses, for tests 49 and 50, and as source addresses for tests 51 and 52. The goal of the tests was to induce the target to start the Reachability process and to exhaust the Neighbor Cache.

From the targets it was possible to verify the initial steps of the Reachability process: target inserted the spoofed addresses in the Neighbor Cache and started to send Neighbor Solicitations, and waited for a Neighbor Advertisement to complete the process, which never arrived. The entry in the Neighbor Cache is then put in a failed state.

In addition, with the exception of test 52, both the target device and the Debian Linux Firewall are affected. For test 52, where the source address has been spoofed, the target Debian Linux device droped the packet, while the firewall began the Reachability process.

The impact of the tests on the target devices is low: both targets' Neighbor Cache have a limitation on the stored entries and used a circular data structure, or a queue, to manage them. The first entry in the data structure is also the first erased when the limit has been reached.

From a network traffic point of view, the tests showed a low impact. Nevertheless, this research want to underline the fact that the network used for the experiment included three devices out of an address space, considering an internal network with 64 bits lenght prefix, of 64 bits. For the future, it is possible to imagine the use of big subnets, for example in the case of small devices used to monitor critical environments (storms, ocean wawes and tsunami): in those cases the impact on the network traffic would be higher, and this result should be taken into consideration, either by finding means to mitigate the risk, or by designing networks with a smaller amount of devices.

The results for the firewall configurations do not depict the same symmetry as for the other message types. It is worth to remember that this research included, in the Netfilter with Best Practices, a rule to block Echo Request from an external network: this measure has been defined as a policy choice, not a best practice. The reason is that, from a best practices point of view, an Echo Request is often used to enumerate a network; this practice has been obsoleted by the huge address space of IPv6, which makes infeasible a complete enumeration. Nevertheless, there is an impact on the target network. An organization must take it into consideration and carefully evaluate the possibility to use best practices with additional rules, accordingly to the nature of their network and requirements.

**Echo Reply**

Test number 55 validates the second hypothesis for Netfilter Open and Netfilter with Best Practices configurations.

Tests 53 and 54 do not show effects on the target systems, which ignored the packets.

Test 55 has been selected after some experiment on a Mitm attack, on the internal network, using Neighbor Advertisements. In order to understand the failure of some early attempts, the Windows 7 firewall has been disabled, to verify if the host-based firewall was the issue. In that situation, the analysis of the network traffic using Wireshark on the Debian Linux devices showed an unusual Parameter Problem message, with code 1 (unrecognized next header type). The discovery led to an in-depth analysis of the reasons and additional verifications.

The first verification implied a situation where the target is expected to send back a Parameter Problem message with code 1. The packet has been crafted with an incorrect next header IPv6 field sent to the Debian Linux target. The value could be a number between 143 and 252, which are unassigned [44]. The target behaved as expected by sending back a Parameter Problem with code 1 message. The same test, but performed against the Windows 7 target, with the firewall activated, failed. The deactivation of the host-based firewall, instead, led to a successful test.

The results of this analysis shows the following:

[44] `http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml`, accessed on 15.11.2016

1. The Debian Linux device is not vulnerable to this enumeration method.
2. The Debian Linux device behaves correctly after the reception of a packet with an unrecognized next header.
3. The Windows 7 device, with the Service Pack 1 and default installation (without further updates) is vulnerable.
4. With this methodology, it is possible to state that the target may correspond to the aformentioned device without an active host-based firewall.

The level of experimentation on this issue does not allow to be more precise, on point 4, on the nature of the target. The experiment should be repeated in an environment, with phisical devices, which include also other OS versions and types. Nevertheless, as a further iteration of this experiment, this research downloaded a Windows 8.1 [45] device: the experiment, without host-based firewall, has been successful. This last result shows that at least two versions of the Windows OS are vulnerable.

This research mentioned IPv6 Ready (see 2.3.1), a project with the aim to test conformance and interoperability. The website [46] offers the possibility to search inside the approved list of the program for Microsoft products. The entry for the version of Windows used inside the experiment [47] shows that the project approved the product, which passed test specification 4.06 related to the Core Protocols. The project publishes the specifications of the tests [48], which includes the generation of a Parameter Problem message with code 1 in case the next Header is not recognized.

This research thinks that the host-based firewall, from a technical point of view, should not be considered as a component of an OS. Therefor the results of the test specified by the project may be considered correct. Nevertheless, the host-based firewall is active by default, and, in fact, modifies the behavior of the device. Which may be seen as a way to have the approval first, but then use other means to break the specification of the IETF and the interoperability with other devices in terms of network traffic.

[45] `https://developer.microsoft.com/en-us/microsoft-edge/tools/vms/#downloads`, downloaded on 15.11.2016

[46] `https://www.ipv6ready.org/db/index.php/public/search/?vn=Microsoft&p=1&o=5&do=1&lim=50`, accessed on 24.11.2016

[47] `https://www.ipv6ready.org/db/index.php/public/logo/02-C-000524/`, accessed on 24.11.2016

[48] `https://www.ipv6ready.org/docs/Core_Conformance_Latest.pdf`, p. 290, accessed on 24.11.2016

Another aspect that is worth to underline is that the same tool used by system administrators to defend, administer, or monitor a network, may also be used by an attacker. This mindset, for example, leads Penetration Tester to emulate attackers and the thechniques they use in order to find the weeknesses of a system. Because of this, as a side effect of the experiment, this research tried to invert the process, that is, by imagine the methodology just found to enumerate the target in a defensive way. The script in appendix H shows the concept: while a thread receives packets and wait for a Parameter Problem, the PoC sends the same packet of the experiment, with a two second interval. The context is given by an attacker that gained a foothold on the target device and needs to deactivate the firewall to perform some tasks. The administrator will start to receive Parameter Problem messages and be adviced that, at least, there is a problem with a device.

This monitor solution may not be ideal because of the additional traffic generated. In spite of that, this research consider it a good example, in terms of awareness, about the interchangeability of the tools between attack and defense.

**Router Solicitation**

The second hypothesis has been negated for Router Solicitation messages. No tests have been selected.

Accordingly to the testing selection for ND protocol, three internal test have been performed, but none of them fulfilled the requirements. Two of them were not successful, while the last used the All Router Multicast Address, which cannot be used from a different network segment.

**Router Advertisement**

The test set related to the Router Advertisement message negates the second hypothesis.

From the internal tests, two attacks have been selected: the injection of an arbitrary prefix, and the change of the MTU for the link on the targets. This generated tests from 56 to 61 with Windows 7 as target, and tests from 62 to 67 directed to Debian Linux. The tests tried to inject a prefix on the targets, using different address as source and by manipulating the flags of the Prefix Information option, without success. The effects on the target can be measured using commands *ipconfig* and *ifconfig*, for Windows and Linux respectively. The probable reasons of the failure are the enforcement of the rule regarding the Next Hop IPv6 field, which must be 255, and of the rule which defines that the source address must have link-local scope [1].

Tests 68 and 69 tried to inject a new MTU value on the targets, without success. The validation is possible with the following commands, respectively for Windows and Linux:

```
netsh interface ipv6 show subinterfaces
cat /proc/sys/net/ipv6/conf/eth0/mtu
```

From a firewall configuration point of view, the packets have been forwarded only by the Netfilter Open configuration.

### Neighbor Solicitation

The test set related to the Neighbor Solicitation message negates the second hypothesis.

The selected internal test showed a particular behavior of the Debian Linux device. The packets have been sent with a Solicited Multicast source address of the target, which started to send Neighbor Advertisement to itself. Test 70 followed the same concept, but with source and destination Global Unicast addresses of the target, but without having the same effect on the target. The reason is again the Next Hop field, which after traversing the firewall is decremented by one.

The firewall with Netfilter Open configuration allowed the packets to be forwarded internally, while the other configurations blocked them.

### Neighbor Advertisement

The test set related to the Neighbor Advertisement message negates the second hypothesis.

From the internal test, the attack against the neighbor cache have been selected. This attack had two goals: try to exhaust the neighbor cache, and override legal entries in the table to produce a DOS condition against the target with respect to other devices in the subnet. Test 71, failed: the first Echo Request started the Reachability process in the target device, but the following Neighbor Advertisement is droped because of a Hop Limit less than 255.

The Netfilter Open has been the only configuration that forwarded the packet.

It is worth to mention the result and the reasons why the internal Mitm against Windows 7 has not been selected. The Windows device uses by default a temporary global unicast address to initiate the conversations, while it uses the permanent one when provides server functionalities or services. In order to perform the Mitm, the attacker pretend to be the Windows 7 device from a firewall point of view, and the firewall from a Windows 7 point of view. This is accomplished by sending Neighbor Advertisements. The issue resides on the fact that the Windows device uses the temporary address to initiate a conversation (e.g. an http request). Without the knowledge of this address, the attacker is able to intercept only half of the conversation, from Windows to the firewall, but not the response. In order to complete the Mitm the attacker needs to announce also the temporary address to the firewall. This address may be disclosed internally after a period (e.g. Reachability process). From an outside network this may happen only if the internal device initiate a request to the attacker machine. For this reason, the test has not been selected.

**Redirect**

The test set related to the Redirect message negates the second hypothesis.

The goal of tests 72 and 73 was an half Mitm attack, with the Attacker as Redirect Target, and the External device as Redirect Destination. The reasons of the failure may be the Hop Limit and the source address which is not of link-local scope.

The Netfilter Open has been the only configuration that forwarded the packet.

**Firewall Configurations**

The tables with the results (see F.2) shows which configurations allows the ICMPv6 traffic to enter the perimeter. Netfilter Open represents the trivial case, but acquires more relevance with respect to the experiment related to the first hypothesis. In order to validate the hypothesis, an experiment must send packets that are at the same time forwarded by the warden, and have an effect on the end system. Nevertheless, the Netfilter Open configuration allows the packets to reach the target and assess the behavior of the end system first. Then, the other configurations allows to evaluate the impact of a particular device, or configuration, on the mitigation of the effects in the case that the first test is successful.

The two configurations related to the ASA firewall blocked the packets. The experiments that validated the hypothesis using Netfilter Open failed for these configuration and mitigated the effects on the end systems. An example of this behavior is test 55, which try to understand if the internal device is a Windows 7 without host-based firewall activated. In this case the packet is not able to reach the destination. On the other side, these configurations violate the best practices by blocking Error and Informational messages, which may break a legitimate network request. From a network security point of view, they follow a white-list approach, which requires a complete knowledge of the services offered and required by the organization and the configuration of access lists to allow the required traffic. From an awareness point of view, instead, the organization must understand that the device itself, in the default configuration, or with the activated icmp module, is not enough, and other resources must be allocated for the necessary expertise to configure the appliance.

The Netfilter with Best Practice configuration, instead, balance the requirements of the network communication with the needs in terms of security. Error and Informational messages are allowed to enter, with the exception of the Echo Request message type, while messages defined by the ND protocol, which are not supposed to be forwarded, are blocked. Nevertheless, test 55 produced packets that were able to traverse the warden, to produce an effect on the end system, and to travel back to the source.

The last statement is very important for this research. This is because, from one side, it shows that, even following best practices, there is not the certainty to have a secure network. On the other side it shows that pursuing security through obscurity is a dangerous practice. The fact that an attacker is not aware that Windows 7 uses the host-based firewall to bypass the standards, does not grant that he or she will discover this information. But the lack of this knowledge may prevent a security administrator to perform the necessary tests and try to find solutions to mitigate possible risks.

# 7. Conclusions

# References

[1] T. Narten, E. Nordmark, W. Simpson, and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)," Internet Requests for Comments, RFC Editor, RFC 4861, September 2007. [Online]. Available: http://www.rfc-editor.org/info/rfc4861

[2] G. Lewandowski, "Network-aware Active Wardens in IPv6," *Electrical Engineering and Computer Science - Dissertations*, 2011.

[3] N. Lucena, J. Pease, P. Yadollahpour, and S. J. Chapin, "Syntax and Semantics-Preserving Application-Layer Protocol Steganography," *Syracuse University*, 2004.

[4] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," Internet Requests for Comments, RFC Editor, RFC 2460, December 1998. [Online]. Available: http://www.rfc-editor.org/info/rfc2460

[5] P. Srisuresh and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)," Internet Requests for Comments, RFC Editor, RFC 3022, January 2001. [Online]. Available: http://www.rfc-editor.org/info/rfc3022

[6] A. Conta, S. Deering, and M. E. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification," Internet Requests for Comments, RFC Editor, RFC 4443, March 2006. [Online]. Available: http://www.rfc-editor.org/info/rfc4443

[7] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," Internet Requests for Comments, RFC Editor, RFC 2119, March 1997. [Online]. Available: http://www.rfc-editor.org/info/rfc2119

[8] J. Postel, "Internet Control Message Protocol," Internet Requests for Comments, RFC Editor, RFC 792, September 1981. [Online]. Available: http://www.rfc-editor.org/info/rfc792

[9] M. Chakraborty, N. Chaki, and A. Cortesi, "A New Intrusion Prevention System for Protecting Smart Grids from ICMPv6 Vulnerabilities," *Conference: 2014 Federated Conference on Computer Science and Information Systems*, vol. 2, 2014.

[10] Fortinet, "Threats on the Horizon: The Rise of the Advanced Persistent Threat," *solution report, Fortinet Inc.*, 2013.

[11] G. D. SecurityLabs, “Uroburos: Highly complex espionage software with Russian roots,” *tech. rep., G Data Software AG*, 2014.

[12] N. Falliere, L. O. Murchu, and E. Chien. (2011) W32.Stuxnet Dossier. Accessed: 21-03-2016. [Online]. Available: http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf

[13] F. Rajpari, “Finding the Advanced Persistent Adversary,” *SANS Institute*, 2014.

[14] S. Northcutt, J. Shenk, D. Shackleford, T. Rosenberg, R. Siles, and S. Mancini, “Penetration Testing: Assessing Your Overall Security Before Attackers Do,” *SANS Institute*, 2006.

[15] S. Bradner, “The Internet Standards Process – Revision 3,” Internet Requests for Comments, RFC Editor, RFC 2026, October 1996. [Online]. Available: http://www.rfc-editor.org/info/rfc2026

[16] J. G. Simmons, “The prisoners’ problem and the subliminal channel,” *Advances in Cryptology, Proceedings of CRYPTO ’83, pages 51–67*, 1984.

[17] A. Mileva and B. Panajotov, “Covert Channels in TCP/IP Protocol Stack,” *Central European Journal of Computer Science*, 2014.

[18] N. Lucena, “Application-level protocol steganography,” *PhD thesis, Syracuse University*, 2008.

[19] D. Watson, M. Smart, and R. Malan, “Protocol Scrubbing: Network Security Through Transparent Flow Modification,” *IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 12, NO. 2*, 2004.

[20] R. P. Murphy. (2006) IPv6/ICMPv6 Covert Channels. Accessed: 2-04-2016. [Online]. Available: https://www.defcon.org/images/defcon-14/dc-14-presentations/DC-14-Murphy.pdf

[21] The Hacker Choice. (2006) Attacking the IPv6 Protocol Suite. Accessed: 4-04-2016. [Online]. Available: https://www.thc.org/papers/vh_thc-ipv6_attack.pdf

[22] IPv6 Ready Committee. (2013) IPv6 Ready Logo White Paper. Accessed: 2-04-2016. [Online]. Available: https://www.ipv6ready.org/docs/IPv6_Ready_Logo_White_Paper_Final.pdf

[23] IPv6 Ready Committee. (2010) Core Conformance. Accessed: 2-04-2016. [Online]. Available: https://www.ipv6ready.org/docs/Core_Conformance_Latest.pdf

[24] IPv6 Ready Committee. (2010) Core Interoperability. Accessed: 2-04-2016. [Online]. Available: https://www.ipv6ready.org/docs/Core_Interoperability_Latest.pdf

[25] A. Pilihanto, "A complete guide on IPv6 attack and defense," *SANS Institute*, 2011.

[26] S. Thomson, T. Narten, and T. Jinmei, "IPv6 Stateless Address Autoconfiguration," Internet Requests for Comments, RFC Editor, RFC 4862, September 2007. [Online]. Available: http://www.rfc-editor.org/info/rfc4862

[27] F. Gont and W. Liu, "Security Implications of IPv6 Options of Type 10xxxxxx draft-gont-6man-ipv6-smurf-amplifier-03," Internet Draft, Draft, March 2013. [Online]. Available: https://tools.ietf.org/html/draft-gont-6man-ipv6-smurf-amplifier-03

[28] S. Peisert and M. Bishop, "How to Design Computer Security Experiments," *Fifth World Conference on Information Security Education: Proceedings of the IFIP TC11 WG 11.8, WISE 5, 19 to 21 June 2007, United States Military Academy, West Point, New York, USA*, 2007.

[29] T. E. Carroll, D. Manz, T. Edgar, and F. L. Greitzer, "Realizing Scientific Methods for Cyber Security," *Proceedings of the 2012 Workshop on Learning from Authoritative Security Experiment Results*, 2012.

[30] B. Kaplan and D. Duchon, "Combining Qualitative and Quantitative Methods in Information Systems Research: A Case Study," *Management Information Systems Research Center, University of Minnesota*, 1988.

[31] National Institute of Standards and Technology. (2009) Guidelines on Firewalls and Firewall Policy. Accessed: 22-04-2016. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-41-Rev1/sp800-41-rev1.pdf

[32] Nasa Integrated Communication Services. (2014) NICS IPv6 Best Practices Guide. Accessed: 22-04-2016. [Online]. Available: https://www.hpc.mil/images/hpcdocs/ipv6/nics_ipv6_best_practices_guide.pdf

[33] E. Davies and J. Mohacsi, "Recommendations for Filtering ICMPv6 Messages in Firewalls," Internet Requests for Comments, RFC Editor, RFC 4890, May 2007. [Online]. Available: http://www.rfc-editor.org/info/rfc4890

[34] Giobbi R. (2008) Filtering ICMPv6 Using Host-Based Firewalls. Accessed: 22-04-2016. [Online]. Available: https://insights.sei.cmu.edu/cert/2008/11/filtering-icmpv6-using-host-based-firewalls.html

# A. Appendix 1 - Protocol Headers

## IPv6

| | | |
|---|---|---|
| Version | Traffic Class | Flow Label |
| Payload Length | Next Header | Hop Limit |
| Source Address | | |
| Destination Address | | |

Table 28. IPv6 Header

## ICMPv6

### RFC 4443

The following are the Protocol Headers described in RFC 4443.

#### Destination Unreachable

| | | |
|---|---|---|
| Type | Code | Checksum |
| Unused | | |
| As much of invocking packet as possible without the ICMPv6 packet exceeding the minimum IPv6 MTU | | |

Table 29. Destination Unreachable

#### Packet Too Big

#### Time Exceeded

| | | |
|---|---|---|
| Type | Code | Checksum |
| MTU | | |
| As much of invoking packet as possible without the ICMPv6 packet exceeding the minimum IPv6 MTU | | |

Table 30. Packet Too Big

| | | |
|---|---|---|
| Type | Code | Checksum |
| Unused | | |
| As much of invoking packet as possible without the ICMPv6 packet exceeding the minimum IPv6 MTU | | |

Table 31. Time Exceeded

**Parameter Problem**

| | | |
|---|---|---|
| Type | Code | Checksum |
| Pointer | | |
| As much of invoking packet as possible without the ICMPv6 packet exceeding the minimum IPv6 MTU | | |

Table 32. Parameter Problem

**Echo Request**

| | | |
|---|---|---|
| Type | Code | Checksum |
| Identifier | | Sequence Number |
| Data | | |

Table 33. Echo Request

**Echo Reply**

| | | |
|---|---|---|
| Type | Code | Checksum |
| Identifier | | Sequence Number |
| Data | | |

Table 34. Echo Reply

## RFC 4861

### Router Solicitation

| | | |
|---|---|---|
| Type | Code | Checksum |
| Identifier | | Reserved |
| Options | | |

Table 35. Router Solicitation

### Router Advertisement

| | | | | |
|---|---|---|---|---|
| Type | Code | | | Checksum |
| Cur Hop Limit | M | O | Reserved | Router Lifetime |
| Reachable Time | | | | |
| Retrans Timer | | | | |
| Options | | | | |

Table 36. Router Advertisement

### Neighbor Solicitation

| | | |
|---|---|---|
| Type | Code | Checksum |
| Reserved | | |
| Target Address | | |
| Options | | |

Table 37. Neighbor Solicitation

### Neighbor Advertisement

### Redirect

### Source/Target link-layer Address

| | | | | | |
|---|---|---|---|---|---|
| Type | | | | Code | Checksum |
| R | S | O | Reserved | | |
| Target Address | | | | | |
| Options | | | | | |

Table 38. Neighbor Advertisement

| | | |
|---|---|---|
| Type | Code | Checksum |
| Reserved | | |
| Target Address | | |
| Destination Address | | |
| Options | | |

Table 39. Redirect

| | | |
|---|---|---|
| Type | Length | Link-layer Address |

Table 40. Source/Target link-layer Address

**Prefix Information**

| Type | Length | Prefix length | L | A | Reserved1 |
|---|---|---|---|---|---|
| Valid Lifetime | | | | | |
| Preferred Lifetime | | | | | |
| Reserved2 | | | | | |
| Prefix | | | | | |

Table 41. Prefix Information

**Redirect Header**

| Type | Length | Reserved |
|---|---|---|
| Reserved | | |
| IP Header + Data | | |

Table 42. Redirect Header

**MTU**

| Type | Length | Reserved |
|---|---|---|
| MTU | | |

Table 43. MTU

## B. Appendix 2 - ICMPv6 specifications

| Processing Rules | |
|---|---|
| Selected | Specification |
| | If an ICMPv6 error message of unknown type is received at its destination, it MUST be passed to the upper-layer process that originated the packet that caused the error, where this can be identified |
| | If an ICMPv6 informational message of unknown type is received, it MUST be silently discarded. |
| | Every ICMPv6 error message MUST include as much of the IPv6 offending (invoking) packet |
| | In cases where the internet-layer protocol is required to pass an ICMPv6 error message to the upper-layer process, the upper-layer protocol type is extracted from the original packet and used to select the appropriate upper-layer process to handle the error. |
| | An ICMPv6 error message MUST NOT be originated as a result of receiving the following:<br>• An ICMPv6 error message.<br>• An ICMPv6 redirect message.<br>• A packet destined to an IPv6 multicast address. (There are two exceptions to this rule: (1) the Packet Too Big Message to allow Path MTU discovery to work for IPv6 multicast, and (2) the Parameter Problem Message, Code 2 reporting an unrecognized IPv6 option that has the Option Type highest- order two bits set to 10).<br>• A packet sent as a link-layer multicast.<br>• A packet sent as a link-layer broadcast.<br>• A packet whose source address does not uniquely identify a single node. |
| | Finally, in order to limit the bandwidth and forwarding costs incurred by originating ICMPv6 error messages, an IPv6 node MUST limit the rate of ICMPv6 error messages it originates. |

Table 44. RFC 4443 - Message Processing Rules

| Destination Unreachable | |
|---|---|
| Selected | Specification |
| | Unused field: This field is unused for all code values. It MUST be initialized to zero by the originator and ignored by the receiver. |
| | A Destination Unreachable message SHOULD be generated by a router, or by the IPv6 layer in the originating node, in response to a packet that cannot be delivered to its destination address for reasons other than congestion. |
| | One specific case in which a Destination Unreachable message is sent with a code 3 is in response to a packet received by a router from a point-to-point link, destined to an address within a subnet assigned to that same link (other than one of the receiving router's own addresses). In such a case, the packet MUST NOT be forwarded back onto the arrival link. |
| | A destination node SHOULD originate a Destination Unreachable message with Code 4 in response to a packet for which the transport protocol (e.g., UDP) has no listener, if that transport protocol has no alternative means to inform the sender. |
| | For security reasons, it is recommended that implementations SHOULD allow sending of ICMP destination unreachable messages to be disabled, preferably on a per-interface basis. |
| | A node receiving the ICMPv6 Destination Unreachable message MUST notify the upper-layer process if the relevant process can be identified |

Table 45. RFC 4443 - Destination Unreachable

| Packet Too Big | |
|---|---|
| Selected | Specification |
| | A Packet Too Big MUST be sent by a router in response to a packet that it cannot forward because the packet is larger than the MTU of the outgoing link. |
| | An incoming Packet Too Big message MUST be passed to the upper-layer process if the relevant process can be identified. |

Table 46. RFC 4443 - Packet Too Big

| Time Exceeded | |
|---|---|
| Selected | Specification |
| | If a router receives a packet with a Hop Limit of zero, or if a router decrements a packet's Hop Limit to zero, it MUST discard the packet and originate an ICMPv6 Time Exceeded message with Code 0 to the source of the packet. |
| | An incoming Time Exceeded message MUST be passed to the upper-layer process if the relevant process can be identified. |

Table 47. RFC 4443 - Time Exceeded

| Parameter Problem | |
|---|---|
| Selected | Specification |
| | If an IPv6 node processing a packet finds a problem with a field in the IPv6 header or extension headers such that it cannot complete processing the packet, it MUST discard the packet and SHOULD originate an ICMPv6 Parameter Problem message to the packet's source, indicating the type and location of the problem. |
| | A node receiving this ICMPv6 message MUST notify the upper-layer process if the relevant process can be identified. |

Table 48. RFC 4443 - Parameter Problem

| Echo Request | |
|---|---|
| Selected | Specification |
| | Every node MUST implement an ICMPv6 Echo responder function that receives Echo Requests and originates corresponding Echo Replies. A node SHOULD also implement an application-layer interface for originating Echo Requests and receiving Echo Replies, for diagnostic purposes. |
| | Echo Request messages MAY be passed to processes receiving ICMP messages. |

Table 49. RFC 4443 - Echo Request

| Echo Reply | |
|---|---|
| Selected | Specification |
| | The source address of an Echo Reply sent in response to a unicast Echo Request message MUST be the same as the destination address of that Echo Request message. |
| | An Echo Reply SHOULD be sent in response to an Echo Request message sent to an IPv6 multicast or anycast address. In this case, the source address of the reply MUST be a unicast address belonging to the interface on which the Echo Request message was received. |
| | The data received in the ICMPv6 Echo Request message MUST be returned entirely and unmodified in the ICMPv6 Echo Reply message. |
| | Echo Reply messages MUST be passed to the process that originated an Echo Request message. |
| | An Echo Reply message MAY be passed to processes that did not originate the Echo Request message. |
| | Note that there is no limitation on the amount of data that can be put in Echo Request and Echo Reply Messages. |

Table 50. RFC 4443 - Echo Reply

| Router Solicitation | |
|---|---|
| Selected | Specification |
| | Hosts MUST silently discard any received Router Solicitation Messages. |
| | A router MUST silently discard any received Router Solicitation messages that do not satisfy all of the following validity checks:<br>• The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.<br>• ICMP Checksum is valid.<br>• ICMP Code is 0.<br>• ICMP length (derived from the IP length) is 8 or more octets.<br>• All included options have a length that is greater than zero.<br>• If the IP source address is the unspecified address, there is no source link-layer address option in the message. |
| | The contents of the Reserved field, and of any unrecognized options, MUST be ignored. |
| | The contents of any defined options that are not specified to be used with Router Solicitation messages MUST be ignored and the packet processed as normal. The only defined option that may appear is the Source Link-Layer Address option. |

Table 51. RFC 4861 - Validation of Router Solicitation

| Router Advertisement | |
|---|---|
| Selected | Specification |
| | A router MUST silently discard any received Router Advertisement messages that do not satisfy all of the following validity checks:<br>• IP Source Address is a link-local address. Routers must use their link-local address as the source for Router Advertisement and Redirect messages so that hosts can uniquely identify routers.<br>• The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.<br>• ICMP Checksum is valid.<br>• ICMP Code is 0.<br>• ICMP length (derived from the IP length) is 16 or more octets.<br>• All included options have a length that is greater than zero. |
| | The contents of the Reserved field, and of any unrecognized options, MUST be ignored. |
| | The contents of any defined options that are not specified to be used with Router Advertisement messages MUST be ignored and the packet processed as normal. The only defined options that may appear are the Source Link-Layer Address, Prefix Information and MTU options. |
| | A host MUST NOT send Router Advertisement messages at any time. |

Table 52. RFC 4861 - Validation of Router Advertisement

| Neighbor Solicitation | |
|---|---|
| Selected | Specification |
| | A node MUST silently discard any received Neighbor Solicitation messages that do not satisfy all of the following validity checks:<br>• The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.<br>• ICMP Checksum is valid.<br>• ICMP Code is 0.<br>• ICMP length (derived from the IP length) is 24 or more octets.<br>• Target Address is not a multicast address.<br>• All included options have a length that is greater than zero.<br>• If the IP source address is the unspecified address, the IP destination address is a solicited-node multicast address.<br>• If the IP source address is the unspecified address, there is no source link-layer address option in the message. |
| | The contents of the Reserved field, and of any unrecognized options, MUST be ignored. |
| | The contents of any defined options that are not specified to be used with Neighbor Solicitation messages MUST be ignored and the packet processed as normal. The only defined option that may appear is the Source Link-Layer Address option. |

Table 53. RFC 4861 - Validation of Neighbor Solicitation

| Neighbor Solicitation - Sending | |
|---|---|
| Selected | Specification |
| | If the source address of the packet prompting the solicitation is the same as one of the addresses assigned to the outgoing interface, that address SHOULD be placed in the IP Source Address of the outgoing solicitation. |
| | If the solicitation is being sent to a solicited-node multicast address, the sender MUST include its link-layer address (if it has one) as a Source Link-Layer Address option. |
| | Otherwise, the sender SHOULD include its link-layer address (if it has one) as a Source Link-Layer Address option. |
| | On unicast solicitations, an implementation MAY omit the Source Link-Layer Address option. |

Table 54. RFC 4861 - Sending Neighbor Solicitation

| Neighbor Solicitation - Reception | |
|---|---|
| Selected | Specification |
| | A valid Neighbor Solicitation that does not meet any of the following requirements MUST be silently discarded:<br>• The Target Address is a "valid" unicast or anycast address assigned to the receiving interface.<br>• The Target Address is a unicast or anycast address for which the node is offering proxy service.<br>• The Target Address is a "tentative" address on which Duplicate Address Detection is being performed. |
| | If the Source Address is not the unspecified address and, on link layers that have addresses, the solicitation includes a Source Link-Layer Address option, then the recipient SHOULD create or update the Neighbor Cache entry for the IP Source Address of the solicitation. |
| | If the Source Address is the unspecified address, the node MUST NOT create or update the Neighbor Cache entry. |

Table 55. RFC 4861 - Reception of Neighbor Solicitation

| Neighbor Advertisement | |
|---|---|
| Selected | Specification |
| | A node MUST silently discard any received Neighbor Advertisement messages that do not satisfy all of the following validity checks:<br>• The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.<br>• ICMP Checksum is valid.<br>• ICMP Code is 0.<br>• ICMP length (derived from the IP length) is 24 or more octets.<br>• Target Address is not a multicast address.<br>• If the IP Destination Address is a multicast address the Solicited flag is zero.<br>• All included options have a length that is greater than zero. |
| | The contents of the Reserved field, and of any unrecognized options, MUST be ignored. |
| | The contents of any defined options that are not specified to be used with Neighbor Advertisement messages MUST be ignored and the packet processed as normal. The only defined option that may appear is the Target Link-Layer Address option. |

Table 56. RFC 4861 - Validation of Neighbor Advertisement

| Sending Solicited Neighbor Advertisement | |
|---|---|
| Selected | Specification |
| | If the solicitation's IP Destination Address is not a multicast address, the Target Link-Layer Address option MAY be omitted. |
| | If the solicitation's IP Destination Address is a multicast address, the Target Link-Layer option MUST be included in the advertisement. |
| | if the node is a router, it MUST set the Router flag to one; otherwise, it MUST set the flag to zero. |
| | If the Target Address is either an anycast address or a unicast address for which the node is providing proxy service, or the Target Link-Layer Address option is not included, the Override flag SHOULD be set to zero. Otherwise, the Override flag SHOULD be set to one. |
| | If the source of the solicitation is the unspecified address, the node MUST set the Solicited flag to zero and multicast the advertisement to the all-nodes address. Otherwise, the node MUST set the Solicited flag to one and unicast the advertisement to the Source Address of the solicitation. |

Table 57. RFC 4861 - Solicited Neighbor Advertisement

| Receipt of Neighbor Advertisement | |
|---|---|
| Selected | Specification |
| | When a valid Neighbor Advertisement is received (either solicited or unsolicited), the Neighbor Cache is searched for the target's entry. If no entry exists, the advertisement SHOULD be silently discarded. |

Table 58. RFC 4861 - Receipt of Neighbor Advertisement

| Redirect | |
|---|---|
| Selected | Specification |
| | A host MUST silently discard any received Redirect message that does not satisfy all of the following validity checks:<br>• IP Source Address is a link-local address. Routers must use their link-local address as the source for Router Advertisement and Redirect messages so that hosts can uniquely identify routers.<br>• The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.<br>• ICMP Checksum is valid.<br>• ICMP Code is 0.<br>• ICMP length (derived from the IP length) is 40 or more octets.<br>• The IP source address of the Redirect is the same as the current first-hop router for the specified ICMP Destination Address.<br>• The ICMP Destination Address field in the redirect message does not contain a multicast address.<br>• The ICMP Target Address is either a link-local address (when redirected to a router) or the same as the ICMP Destination Address (when redirected to the on-link destination).<br>• All included options have a length that is greater than zero. |
| | The contents of the Reserved field, and of any unrecognized options, MUST be ignored. |
| | The contents of any defined options that are not specified to be used with Redirect messages MUST be ignored and the packet processed as normal. The only defined options that may appear are the Target Link-Layer Address option and the Redirected Header option. |
| | A host MUST NOT consider a redirect invalid just because the Target Address of the redirect is not covered under one of the link's prefixes. |

Table 59. RFC 4861 - Validation of Redirect

| Redirect - Router and Host Specifications | |
|---|---|
| Selected | Specification |
| | A router MUST NOT update its routing tables upon receipt of a Redirect. |
| | A host receiving a valid redirect SHOULD update its Destination Cache accordingly so that subsequent traffic goes to the specified target. |
| | If no Destination Cache entry exists for the destination, an implementation SHOULD create such an entry. |
| | If the Target and Destination Addresses are the same, the host MUST treat the Target as on-link. |
| | If the Target Address is not the same as the Destination Address, the host MUST set IsRouter to TRUE for the target. |
| | A host MUST NOT send Redirect messages. |

Table 60. RFC 4861 - Redirect, specifications

| Options Processing | |
|---|---|
| Selected | Specification |
| | In order to ensure that future extensions properly coexist with current implementations, all nodes MUST silently ignore any options they do not recognize in received ND packets and continue processing the packet. |
| | All options specified in this document MUST be recognized. |
| | A node MUST NOT ignore valid options just because the ND message contains unrecognized ones. |
| | The option MUST NOT depend on the presence or absence of any other options. |
| | Options in Neighbor Discovery packets can appear in any order; receivers MUST be prepared to process them independently of their order. |
| | The amount of data to include in the Redirected Header option MUST be limited so that the entire redirect packet does not exceed the minimum MTU required to support IPv6. |
| | The size of an ND packet including the IP header is limited to the link MTU. When adding options to an ND packet, a node MUST NOT exceed the link MTU. |

Table 61. RFC 4861 - Options

## C. Cisco ASA Version

```
 Cisco Adaptive Security Appliance Software Version 9.1(5)
Device Manager Version 7.5(2)

Compiled on Thu 27-Mar-14 09:36 by builders
System image file is "disk0:/asa915-k8.bin"
Config file at boot was "startup-config"

ASA-EXP up 5 hours 57 mins

Hardware: ASA5505, 256 MB RAM, CPU Geode 500 MHz,
Internal ATA Compact Flash, 128MB
BIOS Flash M50FW080 @ 0xfff00000, 1024KB

Encryption hardware device : Cisco ASA-5505 on-board accelerator (revision 0x0)
                      Boot microcode   : CN1000-MC-BOOT-2.00
                      SSL/IKE microcode : CNLite-MC-SSLm-PLUS-2_05
                      IPSec microcode  : CNlite-MC-IPSECm-MAIN-2.09
                      Number of accelerators: 1

 0: Int: Internal-Data0/0 : address is 0021.a02e.8564, irq 11
 1: Ext: Ethernet0/0  : address is 0021.a02e.855c, irq 255
 2: Ext: Ethernet0/1  : address is 0021.a02e.855d, irq 255
 3: Ext: Ethernet0/2  : address is 0021.a02e.855e, irq 255
 4: Ext: Ethernet0/3  : address is 0021.a02e.855f, irq 255
 5: Ext: Ethernet0/4  : address is 0021.a02e.8560, irq 255
 6: Ext: Ethernet0/5  : address is 0021.a02e.8561, irq 255
 7: Ext: Ethernet0/6  : address is 0021.a02e.8562, irq 255
 8: Ext: Ethernet0/7  : address is 0021.a02e.8563, irq 255
 9: Int: Internal-Data0/1 : address is 0000.0003.0002, irq 255
10: Int: Not used     : irq 255
11: Int: Not used     : irq 255

Licensed features for this platform:
Maximum Physical Interfaces : 8       perpetual
VLANs                    : 3          DMZ Restricted
Dual ISPs                : Disabled   perpetual
VLAN Trunk Ports         : 0          perpetual
Inside Hosts             : 50         perpetual
Failover                 : Disabled   perpetual
Encryption-DES           : Enabled    perpetual
Encryption-3DES-AES      : Enabled    perpetual
AnyConnect Premium Peers  : 25         perpetual
AnyConnect Essentials    : Disabled   perpetual
Other VPN Peers          : 10         perpetual
Total VPN Peers          : 25         perpetual
Shared License           : Disabled   perpetual
AnyConnect for Mobile    : Disabled   perpetual
AnyConnect for Cisco VPN Phone : Disabled perpetual
Advanced Endpoint Assessment : Disabled perpetual
UC Phone Proxy Sessions   : 2          perpetual
Total UC Proxy Sessions   : 2          perpetual
Botnet Traffic Filter    : Disabled   perpetual
```

```
Intercompany Media Engine : Disabled   perpetual
Cluster                   : Disabled   perpetual

This platform has a Base license.

Serial Number: JMX1247Z330
Running Permanent Activation Key: 0x4e25d64e 0xf8628255 0xb822197c 0x8714f8d8 0x0b31aba6
Configuration register is 0x1
Configuration last modified by enable_15 at 11:55:20.129 UTC Sat Oct 22 2016
```

## D. Test Set - First Hypothesis

| Message | Field | Message | Field |
|---|---|---|---|
| Destination Unreachable | Code | Router Solicitation | Code |
| Destination Unreachable | Length | Router Solicitation | Reserved |
| Destination Unreachable | Unused | Router Advertisement | Code |
| Packet Too Big | Code | Router Advertisement | Cur Hop Limit |
| Packet Too Big | MTU | Router Advertisement | M |
| Time Exceeded | Code | Router Advertisement | O |
| Time Exceeded | Length | Router Advertisement | H |
| Time Exceeded | Unused | Router Advertisement | Prf |
| Parameter Problem | Code | Router Advertisement | P |
| Parameter Problem | Pointer | Router Advertisement | Reserved |
| Echo Request | Code | Router Advertisement | Router Life Time |
| Echo Request | Identifier | Router Advertisement | Reachable Time |
| Echo Request | Sequence Number | Router Advertisement | Retrans Timer |
| Echo Request | Data | Neighbor Solicitation | Code |
| Echo Reply | Code | Neighbor Solicitation | Reserved |
| Echo Reply | Identifier | Neighbor Solicitation | Target Address |
| Echo Reply | Sequence Number | Neighbor Advertisement | Code |
| Echo Reply | Data | Neighbor Advertisement | R |
| | | Neighbor Advertisement | S |
| | | Neighbor Advertisement | O |
| | | Neighbor Advertisement | Reserved |
| | | Neighbor Advertisement | Target Address |
| | | Redirect | Code |
| | | Redirect | Reserved |
| | | Redirect | Target Address |
| | | Redirect | Destination Address |

Table 62. Data Exfiltration Test Set

## E. Commands - Second Hypothesis

| Test | Command | Test | Command |
|---|---|---|---|
| 1 | execDestUnreachAllWin | 29 | execTimeExceededHopLimitWin |
| 2 | execDestUnreachAllLinux | 30 | execTimeExceededHopLimitLinux |
| 3 | execDestUnreachNoRouteWin | 31 | execTimeExceededFragmentReassemblyWin |
| 4 | execDestUnreachNoRouteLinux | 32 | execTimeExceededFragmentReassemblyLinux |
| 5 | execDestUnreachComAdminProhibWin | 33 | execTimeExceededBadCodeWin |
| 6 | execDestUnreachComAdminProhibLinux | 34 | execTimeExceededBadCodeLinux |
| 7 | execDestUnreachBeyondScopeWin | 35 | execTimeExceededLengthWin |
| 8 | execDestUnreachBeyondScopeLinux | 36 | execTimeExceededLengthLinux |
| 9 | execDestUnreachAdrUnreachWin | 37 | execParameterProblemErrHeaderWin |
| 10 | execDestUnreachAdrUnreachLinux | 38 | execParameterProblemErrHeaderLinux |
| 11 | execDestUnreachPortUnreachWin | 39 | execParameterProblemUnrecHeaderWin |
| 12 | execDestUnreachPortUnreachLinux | 40 | execParameterProblemUnrecHeaderLinux |
| 13 | execDestUnreachSrcFailedPolicyWin | 41 | execParameterProblemUnrecIPOptionWin |
| 14 | execDestUnreachSrcFailedPolicyLinux | 42 | execParameterProblemUnrecIPOptionLinux |
| 15 | execDestUnreachRejectRouteWin | 43 | execParameterProblemBadCodeWin |
| 16 | execDestUnreachRejectRouteLinux | 44 | execParameterProblemBadCodeLinux |
| 17 | execDestUnreachBadCodeWin | 45 | execParameterProblemFloodPointerWin |
| 18 | execDestUnreachBadCodeLinux | 46 | execParameterProblemFloodPointerLinux |
| 19 | execDestUnreachDifferentLengthWin | 47 | execParameterProblemFloodHighPointerWin |
| 20 | execDestUnreachDifferentLengthLinux | 48 | execParameterProblemFloodHighPointerLinux |
| 21 | execPacketTooBigMTUBigWin | 49 | execEchoRequestNeighCacheExhaustionDstVictimWin |
| 22 | execPacketTooBigMTUBigLinux | 50 | execEchoRequestNeighCacheExhaustionDstVictimLinux |
| 23 | execPacketTooBigMTUSmallWin | 51 | execEchoRequestNeighCacheExhaustionSrcVictimWin |
| 24 | execPacketTooBigMTUSmallLinux | 52 | execEchoRequestNeighCacheExhaustionSrcVictimLinux |
| 25 | execPacketTooBigMTUWin | 53 | execEchoReplyRemoteWin |
| 26 | execPacketTooBigMTULinux | 54 | execEchoReplyRemoteLinux |
| 27 | execPacketTooBigBadCodeWin | 55 | execEchoReplyRemoteWin |
| 28 | execPacketTooBigBadCodeLinux | | |

Table 63. PoC Commands - Second Hypothesis

| Test | Command | Test | Command |
|---|---|---|---|
| 56-61 | execRAPrefixRemoteWin | 70 | execNSRemoteSelfSolLinux |
| 62-67 | execRAPrefixRemoteLinux | 71 | execNACacheFloodingRemoteLinux |
| 68 | execRAMTURemoteWin | 72 | execRedirectRemoteWin |
| 69 | execRAMTURemoteLinux | 73 | execRedirectRemoteLinux |

Table 64. PoC Commands - Second Hypothesis

| Command | Command |
|---|---|
| execRSInternalWin | execNSInternalFloodingWin |
| execRSInternalLinux | execNSInternalFloodingLinux |
| execRSInternalFirewall | execNSInternalSelfSolWin |
| execRAPrefixInternalWin | execNSInternalSelfSolLinux |
| execRAPrefixInternalLinux | execNACacheFloodingInternalWin |
| execRAMTUInternalWin | execNACacheFloodingInternalLinux |
| execRAMTUInternalLinux | execNAWinMitmInternal |
|  | execRedirectInternalWin |

Table 65. PoC Commands - Internal Tests

## F. Results

### F.1. First Hypothesis

| Test Number | Message | Field | Bandwidth (Bytes/-Packet) | Firewall Configuration | | | |
|---|---|---|---|---|---|---|---|
| | | | | 1 | 2 | 3 | 4 |
| 1 | Destination Unreachable | Code | 1 | ✓ | ☒ | ☒ | ✓ |
| 2 | Destination Unreachable | Length | 1 | ✓ | ☒ | ☒ | ✓ |
| 3 | Destination Unreachable | Unused | 3 | ✓ | ☒ | ☒ | ✓ |
| 4 | Packet Too Big | Code | 1 | ✓ | ☒ | ☒ | ✓ |
| 5 | Packet Too Big | MTU | 4 | ✓ | ☒ | ☒ | ✓ |
| 6 | Time Exceeded | Code | 1 | ✓ | ☒ | ☒ | ✓ |
| 7 | Time Exceeded | Length | 1 | ✓ | ☒ | ☒ | ✓ |
| 8 | Time Exceeded | Unused | 3 | ✓ | ☒ | ☒ | ✓ |
| 9 | Parameter Problem | Code | 1 | ✓ | ☒ | ☒ | ✓ |
| 10 | Parameter Problem | Pointer | 4 | ✓ | ☒ | ☒ | ✓ |
| 11 | Echo Request | Code | 1 | ✓ | ✓ | ☒ | ✓ |
| 12 | Echo Request | Identifier | 2 | ✓ | ✓ | ✓ | ✓ |
| 13 | Echo Request | Sequence Number | 2 | ✓ | ✓ | ✓ | ✓ |
| 14 | Echo Request | Data | 8 | ✓ | ✓ | ✓ | ✓ |
| 15 | Echo Reply | Code | 1 | ✓ | ✓ | ☒ | ☒ |
| 16 | Echo Reply | Identifier | 2 | ✓ | ✓ | ☒ | ☒ |
| 17 | Echo Reply | Sequence Number | 2 | ✓ | ✓ | ☒ | ☒ |
| 18 | Echo Reply | Data | 8 | ✓ | ✓ | ☒ | ☒ |
| 19 | Router Solicitation | Code | 1 | ✓ | ✓ | ✓ | ☒ |
| 20 | Router Solicitation | Reserved | 4 | ✓ | ✓ | ✓ | ☒ |
| 21 | Router Advertisement | Code | 1 | ✓ | ✓ | ✓ | ☒ |
| 22 | Router Advertisement | Cur Hop Limit | 1 | ✓ | ✓ | ✓ | ☒ |
| 23 | Router Advertisement | M | 0.12 | ✓ | ✓ | ✓ | ☒ |
| 24 | Router Advertisement | O | 0.12 | ✓ | ✓ | ✓ | ☒ |
| 25 | Router Advertisement | H | 0.12 | ✓ | ✓ | ✓ | ☒ |
| 26 | Router Advertisement | Prf | 0.25 | ✓ | ✓ | ✓ | ☒ |
| 27 | Router Advertisement | P | 0.12 | ✓ | ✓ | ✓ | ☒ |
| 28 | Router Advertisement | Reserved | 0.25 | ✓ | ✓ | ✓ | ☒ |
| 29 | Router Advertisement | Router Life Time | 2 | ✓ | ✓ | ✓ | ☒ |
| 30 | Router Advertisement | Reachable Time | 4 | ✓ | ✓ | ✓ | ☒ |
| 31 | Router Advertisement | Retrans Timer | 4 | ✓ | ✓ | ✓ | ☒ |

Table 66. Data Exfiltration Results

| Test Number | Message | Field | Bandwidth (Bytes/-Packet) | Firewall Configuration | | | |
|---|---|---|---|---|---|---|---|
| | | | | 1 | 2 | 3 | 4 |
| 32 | Neighbor Solicitation | Code | 1 | ✓ | ✓ | ✓ | ☒ |
| 33 | Neighbor Solicitation | Reserved | 4 | ✓ | ✓ | ✓ | ☒ |
| 34 | Neighbor Solicitation | Target Address | 16 | ✓ | ✓ | ✓ | ☒ |
| 35 | Neighbor Advertisement | Code | 1 | ✓ | ✓ | ✓ | ☒ |
| 36 | Neighbor Advertisement | R | 0.12 | ✓ | ✓ | ✓ | ☒ |
| 37 | Neighbor Advertisement | S | 0.12 | ✓ | ✓ | ✓ | ☒ |
| 38 | Neighbor Advertisement | O | 0.12 | ✓ | ✓ | ✓ | ☒ |
| 39 | Neighbor Advertisement | Reserved | 3 | ✓ | ✓ | ✓ | ☒ |
| 40 | Neighbor Advertisement | Target Address | 16 | ✓ | ✓ | ✓ | ☒ |
| 41 | Redirect | Code | 1 | ✓ | ✓ | ✓ | ☒ |
| 42 | Redirect | Reserved | 4 | ✓ | ✓ | ✓ | ☒ |
| 43 | Redirect | Target Address | 16 | ✓ | ✓ | ✓ | ☒ |
| 44 | Redirect | Destination Address | 16 | ✓ | ✓ | ✓ | ☒ |

Table 67. Data Exfiltration Results

## F.2. Second Hypothesis

| Test | Description | Tgt. OS | Firewall Configuration | | | | Target behavior when packets reach destination (observed using Wireshark) |
|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | |
| 1 | Testing general behavior by sending all codes in sequence | Win | ✓ | ☒ | ☒ | ✓ | No reactions from target, using Wireshark on firewall's eth0 |
| 2 | Testing general behavior by sending all codes in sequence | Linux | ✓ | ☒ | ☒ | ✓ | No reactions from target, using Wireshark on firewall's eth0 |
| 3 | Code 0, Src is Attacker, Echo Request payload | Win | ✓ | ☒ | ☒ | ✓ | No effects on conversation (http and echo request) |
| 4 | Code 0, Src is Attacker, Echo Request payload | Linux | ✓ | ☒ | ☒ | ✓ | No effects on conversation (http and echo request) |
| 5 | Code 1, Src is Firewall, TCP payload | Win | ✓ | ☒ | ☒ | ✓ | No effects on conversation (http and echo request) |
| 6 | Code 1, Src is Firewall, TCP payload | Linux | ✓ | ☒ | ☒ | ✓ | No effects on conversation (http and echo request) |
| 7 | Code 2, Src is Firewall, TCP payload | Win | ✓ | ☒ | ☒ | ✓ | No effects on conversation (http and echo request) |
| 8 | Code 2, Src is Firewall, TCP payload | Linux | ✓ | ☒ | ☒ | ✓ | No effects on conversation (http and echo request) |
| 9 | Code 3, Src is Attacker, TCP payload | Win | ✓ | ☒ | ☒ | ✓ | No effects on conversation (http) |
| 10 | Code 3, Src is Attacker, TCP payload | Linux | ✓ | ☒ | ☒ | ✓ | No effects on conversation (http) |
| 11 | Code 4, Src is Attacker, TCP payload | Win | ✓ | ☒ | ☒ | ✓ | No effects on conversation (http) |
| 12 | Code 4, Src is Attacker, TCP payload | Linux | ✓ | ☒ | ☒ | ✓ | No effects on conversation (http) |
| 13 | Code 5, Src is Firewall, TCP payload | Win | ✓ | ☒ | ☒ | ✓ | No effects on conversation (http) |
| 14 | Code 5, Src is Firewall, TCP payload | Linux | ✓ | ☒ | ☒ | ✓ | No effects on conversation (http) |
| 15 | Code 6, Src if Firewall, Echo Request payload | Win | ✓ | ☒ | ☒ | ✓ | No effects on conversation (http and echo request) |
| 16 | Code 6, Src if Firewall, Echo Request payload | Linux | ✓ | ☒ | ☒ | ✓ | No effects on conversation (http and echo request) |
| 17 | Testing unknown codes (7 to 255) | Win | ✓ | ☒ | ☒ | ✓ | Packets dropped at destination |
| 18 | Testing unknown codes (7 to 255) | Linux | ✓ | ☒ | ☒ | ✓ | Packets dropped at destination |
| 19 | Testing Length field (0 to 255) | Win | ✓ | ☒ | ☒ | ✓ | |
| 20 | Testing Length field (0 to 255) | Linux | ✓ | ☒ | ☒ | ✓ | |

Table 68. Attacking internal network, Destination Unreachable (Type 1)

| Test | Description | Tgt. OS | Firewall Configuration |  |  |  | Target behavior when packets reach destination (observed using Wireshark) |
|---|---|---|---|---|---|---|---|
|  |  |  | 1 | 2 | 3 | 4 |  |
| 21 | MTU is 2000, Src is Firewall, Echo Request payload | Win | ✓ | ☒ | ☒ | ✓ | No evidence of changes |
| 22 | MTU is 2000, Src is Firewall, Echo Request payload | Linux | ✓ | ☒ | ☒ | ✓ | No evidence of changes |
| 23 | MTU is 500, Src is Firewall, Echo Request payload | Win | ✓ | ☒ | ☒ | ✓ | No evidence of changes |
| 24 | MTU is 500, Src is Firewall, Echo Request payload | Linux | ✓ | ☒ | ☒ | ✓ | No evidence of changes |
| 25 | MTU is 1300, Src is Firewall, Echo Request payload | Win | ✓ | ☒ | ☒ | ✓ | No evidence of changes |
| 26 | MTU is 1300, Src is Firewall, Echo Request payload | Linux | ✓ | ☒ | ☒ | ✓ | No evidence of changes |
| 27 | Unknown codes (1-255), Src is Firewall, Echo Request payload | Win | ✓ | ☒ | ☒ | ✓ | No evidence of changes |
| 28 | Unknown codes (1-255), Src is Firewall, Echo Request payload | Linux | ✓ | ☒ | ☒ | ✓ | No evidence of changes |

Table 69. Attacking internal network, Packet Too Big (Type 2)

| Test | Description | Tgt. OS | Firewall Configuration |  |  |  | Target behavior when packets reach destination (observed using Wireshark) |
|---|---|---|---|---|---|---|---|
|  |  |  | 1 | 2 | 3 | 4 |  |
| 29 | Code 0, Src is Firewall, Echo Request payload | Win | ✓ | ☒ | ☒ | ✓ | No evidence of effects |
| 30 | Code 0, Src is Firewall, Echo Request payload | Linux | ✓ | ☒ | ☒ | ✓ | No evidence of effects |
| 31 | Code 1, Src is Firewall, Echo Request payload | Win | ✓ | ☒ | ☒ | ✓ | No evidence of effects |
| 32 | Code 1, Src is Firewall, Echo Request payload | Linux | ✓ | ☒ | ☒ | ✓ | No evidence of effects |
| 33 | Unknown codes (2-255), Src is Firewall, Echo Request payload | Win | ✓ | ☒ | ☒ | ✓ | No evidence of effects |
| 34 | Unknown codes (2-255), Src is Firewall, Echo Request payload | Linux | ✓ | ☒ | ☒ | ✓ | No evidence of effects |
| 35 | Testing Length field (0 to 255) | Win | ✓ | ☒ | ☒ | ✓ |  |
| 36 | Testing Length field (0 to 255) | Linux | ✓ | ☒ | ☒ | ✓ |  |

Table 70. Attacking internal network, Time Exceeded (Type 3)

| Test | Description | Tgt. OS | Firewall Configuration | | | | Target behavior when packets reach destination (observed using Wireshark and OS specific commands) |
|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | |
| 37 | Code 0, Src is Firewall, Echo Request payload | Win | ✓ | ☒ | ☒ | ✓ | No evidence of effects |
| 38 | Code 0, Src is Firewall, Echo Request payload | Linux | ✓ | ☒ | ☒ | ✓ | No evidence of effects |
| 39 | Code 1, Src is Firewall, Echo Request payload | Win | ✓ | ☒ | ☒ | ✓ | No evidence of effects |
| 40 | Code 1, Src is Firewall, Echo Request payload | Linux | ✓ | ☒ | ☒ | ✓ | No evidence of effects |
| 41 | Code 2, Src is Firewall, Echo Request payload | Win | ✓ | ☒ | ☒ | ✓ | No evidence of effects |
| 42 | Code 2, Src is Firewall, Echo Request payload | Linux | ✓ | ☒ | ☒ | ✓ | No evidence of effects |
| 43 | Unknown codes (4-255), Src is Firewall, Echo Request payload | Win | ✓ | ☒ | ☒ | ✓ | No evidence of effects |
| 44 | Unknown codes (4-255), Src is Firewall, Echo Request payload | Linux | ✓ | ☒ | ☒ | ✓ | No evidence of effects |
| 45 | Flooding pointer values (0-1024), Src is Firewall, Echo Request payload | Win | ✓ | ☒ | ☒ | ✓ | No changes in end device, possible measurement and comparisons using Task Manager |
| 46 | Flooding pointer values (0-1024), Src is Firewall, Echo Request payload | Linux | ✓ | ☒ | ☒ | ✓ | No changes in end device, possible measurement and comparisons using command top |
| 47 | Flooding pointer values (3024000000-3024001000), Src is Firewall, Echo Request payload | Win | ✓ | ☒ | ☒ | ✓ | No changes in end device, possible measurement and comparisons using Task Manager |
| 48 | Flooding pointer values (3024000000,3024001000), Src is Firewall, Echo Request payload | Linux | ✓ | ☒ | ☒ | ✓ | No changes in end device, possible measurement and comparisons using command top |

Table 71. Attacking internal network, Parameter Problem (Type 4)

| Test | Description | Tgt. OS | Firewall Configuration | | | | Target behavior when packets reach destination (observed using Wireshark) |
|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | |
| 49 | Neighbor Cache Exhaustion, Src is spoofed prefix /64 plus 16 bits range, Dst is Victim | Win | ✓ | ☒ | ☒ | ☒ | Both victim and linux firewall affected |
| 50 | Neighbor Cache Exhaustion, Src is spoofed prefix /64 plus 16 bits range, Dst is Victim | Linux | ✓ | ☒ | ☒ | ☒ | Both victim and linux firewall affected |
| 51 | Neighbor Cache Exhaustion, Src is Victim, Dst is spoofed prefix /64 plus 16 bits range | Win | ✓ | ☒ | ☒ | ☒ | Both victim and linux firewall affected |
| 52 | Neighbor Cache Exhaustion, Src is Victim, Dst is spoofed prefix /64 plus 16 bits range | Linux | ✓ | ☒ | ☒ | ☒ | Only linux firewall affected |

Table 72. Attacking internal network, Echo Request (Type 128)

| Test | Description | Tgt. OS | Firewall Configuration | | | | Target behavior when packets reach destination (observed using Wireshark) |
|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | |
| 53 | Src is Attacker, packet sent without request | Win | ✓ | ☒ | ☒ | ✓ | No evidence of effects |
| 54 | Src is Attacker, packet sent without request | Linux | ✓ | ☒ | ☒ | ✓ | No evidence of effects |
| 55 | Src is Attacker, Victim firewall disabled | Win | ✓ | ☒ | ☒ | ✓ | Windows end system start to send ICMPv6 Parameter Problem with code 1 to Src |

Table 73. Attacking internal network, Echo Reply (Type 129)

| Selected | Internal Attack Description | Tgt. OS | Results |
|---|---|---|---|
| ☒ | Flood Victim with NS | Win | The test failed, the target host drops the packets without generating new processes or messages |
| ☒ | Flood Victim with NS | Linux | The test failed, the target host drops the packets without generating new processes or messages |
| ☒ | Flood Victim with NS, Dst is all router multicast | Firewall | Victim starts reachability process, test not selected because of Dst |

Table 74. Attacking internal network, Router Solicitation (Type 133)

| Selected | Internal Attack Description | Tgt. OS | Results | | | | |
|---|---|---|---|---|---|---|---|
| ✓ | Inject arbitrary prefix address, Src is Firewall Link-local | Win | The prefix is successfully addedd and new address generated | | | | |
| ✓ | Inject arbitrary prefix address, Src is Firewall Link-local | Linux | The prefix is successfully addedd and new address generated | | | | |
| ✓ | Inject MTU value, Src is Firewall Link-local | Win | The MTU changed to 1350 | | | | |
| ✓ | Inject MTU value, Src is Firewall Link-local | Linux | The MTU changed to 1350 | | | | |
| Test | Description | Tgt. OS | Firewall Configuration | | | | Target behavior when packets reach destination (observed using Wireshark) |
| | | | 1 | 2 | 3 | 4 | |
| 56 | Remote implementation of arbitrary prefix injection, Src is Firewall | Win | ✓ | ☒ | ☒ | ☒ | Test fails: Hop Limit < 255, Src is not Link-local |
| 57 | Remote implementation of arbitrary prefix injection, Src is Debian Linux | Win | ✓ | ☒ | ☒ | ☒ | Test fails: Hop Limit < 255, Src is not Link-local |
| 58 | Remote implementation of arbitrary prefix injection, Src is Windows 7 | Win | ✓ | ☒ | ☒ | ☒ | Test fails: Hop Limit < 255, Src is not Link-local |
| 59 | Remote implementation of arbitrary prefix injection, Src is Firewall, L is 0 | Win | ✓ | ☒ | ☒ | ☒ | Test fails: Hop Limit < 255, Src is not Link-local |
| 60 | Remote implementation of arbitrary prefix injection, Src is Firewall, R is 0 | Win | ✓ | ☒ | ☒ | ☒ | Test fails: Hop Limit < 255, Src is not Link-local |
| 61 | Remote implementation of arbitrary prefix injection, Src is Firewall, L and R are 0 | Win | ✓ | ☒ | ☒ | ☒ | Test fails: Hop Limit < 255, Src is not Link-local |
| 62 | Remote implementation of arbitrary prefix injection, Src is Firewall | Linux | ✓ | ☒ | ☒ | ☒ | Test fails: Hop Limit < 255, Src is not Link-local |
| 63 | Remote implementation of arbitrary prefix injection, Src is Debian Linux | Linux | ✓ | ☒ | ☒ | ☒ | Test fails: Hop Limit < 255, Src is not Link-local |
| 64 | Remote implementation of arbitrary prefix injection, Src is Windows 7 | Linux | ✓ | ☒ | ☒ | ☒ | Test fails: Hop Limit < 255, Src is not Link-local |
| 65 | Remote implementation of arbitrary prefix injection, Src is Firewall, L is 0 | Linux | ✓ | ☒ | ☒ | ☒ | Test fails: Hop Limit < 255, Src is not Link-local |

Table 75. Attacking internal network, Router Advertisement (Type 134)

| Test | Description | Tgt. OS | Firewall Configuration | | | | Target behavior when packets reach destination (observed using Wireshark) |
|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | |
| 66 | Remote implementation of arbitrary prefix injection, Src is Firewall, R is 0 | Linux | ✓ | ☒ | ☒ | ☒ | Test fails: Hop Limit < 255, Src is not Link-local |
| 67 | Remote implementation of arbitrary prefix injection, Src is Firewall, L and R are 0 | Linux | ✓ | ☒ | ☒ | ☒ | Test fails: Hop Limit < 255, Src is not Link-local |
| 68 | Remote implementation of MTU injection, Src is equal to Dst | Win | ✓ | ☒ | ☒ | ☒ | Test fails: Hop Limit < 255, Src is not Link-local |
| 69 | Remote implementation of MTU injection, Src is equal to Dst | Linux | ✓ | ☒ | ☒ | ☒ | Test fails: Hop Limit < 255, Src is not Link-local |

Table 76. Attacking internal network, Router Advertisement (Type 134)

| Selected | Internal Attack Description | Tgt. OS | Results |
|---|---|---|---|
| ☒ | The aim of this attack is to flood the target with NS, using spoofed src address | Win Linux | The test failed, the target hosts drop the packets without generating new processes or messages |
| ☒ | Attacker sends NS with Src MAC and Dst Solicited Multicast both of the target, the option Src link layer is the firewall | Win | Packets are ignored by the receiver |
| ✓ | Attacker sends NS with Src MAC and Solicited Multicast both of the target, the option Src link layer is the firewall | Linux | Packets arrive at Dst and the device start to send NA to itself |

| Test | Description | Tgt. OS | Firewall Configuration | | | | Target behavior when packets reach destination (observed using Wireshark) |
|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | |
| 70 | Src MAC is Firewall, Dst and Src IPv6 is the Target | Linux | ✓ | ☒ | ☒ | ☒ | Packets arrive but droped, Hop Limit < 255 |

Table 77. Attacking internal network, Neighbor Solicitation (Type 135)

| Se-lected | Internal Attack Description | Tgt. OS | Results | | | | |
|---|---|---|---|---|---|---|---|
| ☒ | Internal Neighbor Cache flooding with Solicited and Override flags | Win | Failed, ignored by victim | | | | |
| ✓ | Internal Neighbor Cache flooding with Solicited and Override flags | Linux | Test is successful, Neigh Cache overridden | | | | |
| ☒ | Mitm attack using Debian Linux as attacker. NA to Target, NAs to Firewall with both Win temporary and permanent IPv6 | Win | Test is successful, from attacker (Debian Linux Internal) is possible to intercept traffic from/to the target. Test not selected, Win temporary not disclosed to external network | | | | |
| Test | Description | Tgt. OS | Firewall Configuration | | | | Target behavior when packets reach destination (observed using Wireshark) |
| | | | 1 | 2 | 3 | 4 | |
| 71 | Remote implementation of internal cache flooding, Src is spoofed IPv6, Dst is Target | Linux | ✓ | ☒ | ☒ | ☒ | Echo Request arrives at Dst, Target start sending NS, but the arrived NA is droped, Hop Limit is <255 |

Table 78. Attacking internal network, Neighbor Advertisement (Type 136)

| Se-lected | Internal Attack Description | Tgt. OS | Results | | | | |
|---|---|---|---|---|---|---|---|
| ✓ | Half Mitm attack using Debian Linux as attacker. Redirect to victim | Win | Target is blocking Redirect by default. Without host firewall attack successful, half of the traffic is intercepted | | | | |
| Test | Description | Tgt. OS | Firewall Configuration | | | | Target behavior when packets reach destination (observed using Wireshark) |
| | | | 1 | 2 | 3 | 4 | |
| 72 | Remote implementation of internal half Mitm, Src is spoofed Firewall, Dst is Victim, Redirect Target is Attacker and Redirect Dst is External Adr | Win | ✓ | ☒ | ☒ | ☒ | Test fails, with and without active host firewall: Hop Limit < 255, Src is not Link-local, Win start conversation with tmp address. Note: for ASA tests, packets blocked, but for experiment not possible to use other external host |
| 73 | Remote implementation of internal half Mitm, Src is spoofed Firewall, Dst is Victim, Redirect Target is Attacker and Redirect Dst is External Adr | Linux | ✓ | ☒ | ☒ | ☒ | Test fails: Hop Limit < 255, Src is not Link-local. Note: for ASA tests, packets blocked, but for experiment not possible to use other external host |

Table 79. Attacking internal network, Redirect (Type 137)

# G. Firewall Configurations

## G.1. Netfilter Open Configuration

Listing 1. Netfilter open

```
#!/bin/bash

#flush iptables
ip6tables -F

#activate forwarding
sysctl -w net/ipv6/conf/all/forwarding=1

#default policy
ip6tables -P INPUT ACCEPT
ip6tables -P FORWARD ACCEPT
ip6tables -P OUTPUT ACCEPT

#forwarding chain
ip6tables -A FORWARD -i eth0 -j ACCEPT
ip6tables -A FORWARD -i eth1 -j ACCEPT
```

## G.2. ASA Default Configuration

Listing 2. ASA Default

```
: Saved
: Written by enable_15 at 14:36:57.759 UTC Fri Oct 14 2016
!
ASA Version 9.1(5)
!
hostname ASA-EXP
domain-name asaexperiment.org
enable password PmNe1e0C3tJdCLe8 encrypted
xlate per-session deny tcp any4 any4
xlate per-session deny tcp any4 any6
xlate per-session deny tcp any6 any4
xlate per-session deny tcp any6 any6
xlate per-session deny udp any4 any4 eq domain
xlate per-session deny udp any4 any6 eq domain
xlate per-session deny udp any6 any4 eq domain
xlate per-session deny udp any6 any6 eq domain
passwd 2KFQnbNIdI.2KYOU encrypted
names
!
interface Ethernet0/0
 switchport access vlan 2
!
```

```
interface Ethernet0/1
!
interface Ethernet0/2
 shutdown
!
interface Ethernet0/3
 shutdown
!
interface Ethernet0/4
 shutdown
!
interface Ethernet0/5
 shutdown
!
interface Ethernet0/6
 shutdown
!
interface Ethernet0/7
 shutdown
!
interface Vlan1
 nameif inside
 security-level 100
 no ip address
 ipv6 address 2001:abcd:acad:2::1/64
 ipv6 address fe80::1 link-local
!
interface Vlan2
 nameif outside
 security-level 0
 no ip address
 ipv6 address 2001:abcd:acad:1::1/64
 ipv6 address fe80::1 link-local
!
ftp mode passive
dns server-group DefaultDNS
 domain-name asaexperiment.org
pager lines 24
mtu inside 1500
mtu outside 1500
icmp unreachable rate-limit 1 burst-size 1
asdm image disk0:/asdm-752.bin
no asdm history enable
arp timeout 14400
no arp permit-nonconnected
timeout xlate 3:00:00
timeout pat-xlate 0:00:30
timeout conn 1:00:00 half-closed 0:10:00 udp 0:02:00 icmp 0:00:02
timeout sunrpc 0:10:00 h323 0:05:00 h225 1:00:00 mgcp 0:05:00 mgcp-pat 0:05:00
timeout sip 0:30:00 sip_media 0:02:00 sip-invite 0:03:00 sip-disconnect 0:02:00
timeout sip-provisional-media 0:02:00 uauth 0:05:00 absolute
timeout tcp-proxy-reassembly 0:01:00
timeout floating-conn 0:00:00
dynamic-access-policy-record DfltAccessPolicy
user-identity default-domain LOCAL
```

```
aaa authentication ssh console LOCAL
http server enable
http fe80::/64 inside
http 2001:abcd:acad:2::/64 inside
no snmp-server location
no snmp-server contact
snmp-server enable traps snmp authentication linkup linkdown coldstart warmstart
crypto ipsec security-association pmtu-aging infinite
crypto ca trustpool policy
telnet timeout 5
ssh scopy enable
ssh stricthostkeycheck
ssh 2001:abcd:acad:2::/64 inside
ssh timeout 5
ssh key-exchange group dh-group1-sha1
console timeout 0

threat-detection statistics access-list
no threat-detection statistics tcp-intercept
username admin password f3UhLvUj1QsXsuK7 encrypted privilege 15
!
class-map inspection_default
 match default-inspection-traffic
!
!
policy-map type inspect dns preset_dns_map
 parameters
  message-length maximum client auto
  message-length maximum 512
policy-map global_policy
 class inspection_default
  inspect dns preset_dns_map
  inspect ftp
  inspect h323 h225
  inspect h323 ras
  inspect ip-options
  inspect netbios
  inspect rsh
  inspect rtsp
  inspect skinny
  inspect esmtp
  inspect sqlnet
  inspect sunrpc
  inspect tftp
  inspect sip
  inspect xdmcp
!
service-policy global_policy global
prompt hostname context
no call-home reporting anonymous
call-home
 profile CiscoTAC-1
  no active
  destination address http https://tools.cisco.com/its/service/oddce/services/DDCEService
  destination address email callhome@cisco.com
  destination transport-method http
  subscribe-to-alert-group diagnostic
  subscribe-to-alert-group environment
  subscribe-to-alert-group inventory periodic monthly
  subscribe-to-alert-group configuration periodic monthly
  subscribe-to-alert-group telemetry periodic daily
Cryptochecksum:5538f7101b9219167b6700a3ec406957
: end
```

## G.3. ASA Default with ICMP Module Configuration

Listing 3. ASA with ICMP module enabled

```
: Saved
: Written by enable_15 at 17:34:08.819 UTC Fri Oct 14 2016
!
ASA Version 9.1(5)
!
hostname ASA-EXP
domain-name asaexperiment.org
enable password PmNe1e0C3tJdCLe8 encrypted
xlate per-session deny tcp any4 any4
xlate per-session deny tcp any4 any6
xlate per-session deny tcp any6 any4
xlate per-session deny tcp any6 any6
xlate per-session deny udp any4 any4 eq domain
xlate per-session deny udp any4 any6 eq domain
xlate per-session deny udp any6 any4 eq domain
xlate per-session deny udp any6 any6 eq domain
passwd 2KFQnbNIdI.2KYOU encrypted
names
!
interface Ethernet0/0
 switchport access vlan 2
!
interface Ethernet0/1
!
interface Ethernet0/2
 shutdown
!
interface Ethernet0/3
 shutdown
!
interface Ethernet0/4
 shutdown
!
interface Ethernet0/5
 shutdown
!
interface Ethernet0/6
 shutdown
!
interface Ethernet0/7
```

```
 shutdown
!
interface Vlan1
 nameif inside
 security-level 100
 no ip address
 ipv6 address 2001:abcd:acad:2::1/64
 ipv6 address fe80::1 link-local
!
interface Vlan2
 nameif outside
 security-level 0
 no ip address
 ipv6 address 2001:abcd:acad:1::1/64
 ipv6 address fe80::1 link-local
!
ftp mode passive
dns server-group DefaultDNS
 domain-name asaexperiment.org
pager lines 24
mtu inside 1500
mtu outside 1500
icmp unreachable rate-limit 1 burst-size 1
asdm image disk0:/asdm-752.bin
no asdm history enable
arp timeout 14400
no arp permit-nonconnected
timeout xlate 3:00:00
timeout pat-xlate 0:00:30
timeout conn 1:00:00 half-closed 0:10:00 udp 0:02:00 icmp 0:00:02
timeout sunrpc 0:10:00 h323 0:05:00 h225 1:00:00 mgcp 0:05:00 mgcp-pat 0:05:00
timeout sip 0:30:00 sip_media 0:02:00 sip-invite 0:03:00 sip-disconnect 0:02:00
timeout sip-provisional-media 0:02:00 uauth 0:05:00 absolute
timeout tcp-proxy-reassembly 0:01:00
timeout floating-conn 0:00:00
dynamic-access-policy-record DfltAccessPolicy
user-identity default-domain LOCAL
aaa authentication ssh console LOCAL
http server enable
http fe80::/64 inside
http 2001:abcd:acad:2::/64 inside
no snmp-server location
no snmp-server contact
snmp-server enable traps snmp authentication linkup linkdown coldstart warmstart
crypto ipsec security-association pmtu-aging infinite
crypto ca trustpool policy
telnet timeout 5
ssh scopy enable
no ssh stricthostkeycheck
ssh 2001:abcd:acad:2::/64 inside
ssh timeout 5
ssh key-exchange group dh-group14-sha1
console timeout 0

threat-detection statistics access-list
```

```
no threat-detection statistics tcp-intercept
username admin password SND/6wmAgmpgep14 encrypted privilege 15
!
class-map inspection_default
 match default-inspection-traffic
!
!
policy-map type inspect dns preset_dns_map
 parameters
  message-length maximum client auto
  message-length maximum 512
policy-map global_policy
 class inspection_default
  inspect dns preset_dns_map
  inspect ftp
  inspect h323 h225
  inspect h323 ras
  inspect ip-options
  inspect netbios
  inspect rsh
  inspect rtsp
  inspect skinny
  inspect esmtp
  inspect sqlnet
  inspect sunrpc
  inspect tftp
  inspect sip
  inspect xdmcp
  inspect icmp
!
service-policy global_policy global
prompt hostname context
no call-home reporting anonymous
call-home
 profile CiscoTAC-1
  no active
  destination address http https://tools.cisco.com/its/service/oddce/services/DDCEService
  destination address email callhome@cisco.com
  destination transport-method http
  subscribe-to-alert-group diagnostic
  subscribe-to-alert-group environment
  subscribe-to-alert-group inventory periodic monthly
  subscribe-to-alert-group configuration periodic monthly
  subscribe-to-alert-group telemetry periodic daily
Cryptochecksum:f6683b26250bf4261a0f68b833b7a365
: end
```

## G.4. Netfilter with Best Practices Configuration

Listing 4. Netfilter with Best Practices

```
#!/bin/bash
```

```
#define in out variables
INSIDEIF=eth0
INSIDENET=2001:abcd:acad:2::1/64
OUTSIDEIF=eth1
OUTSIDENET=2001:abcd:acad:1::1/64

#activate forwarding
sysctl -w net/ipv6/conf/all/forwarding=1

#clean all
ip6tables -F
ip6tables -X ICMPV6-TO-OUT
ip6tables -X ICMPV6-TO-IN
ip6tables -X SSH-IN
ip6tables -X SSH-OUT
ip6tables -Z

#create ad hoc chains
ip6tables -N ICMPV6-TO-OUT
ip6tables -N ICMPV6-TO-IN
ip6tables -N SSH-IN
ip6tables -N SSH-OUT

#default policy is to drop (whitelist approach)
ip6tables -P INPUT DROP
ip6tables -P FORWARD DROP
ip6tables -P OUTPUT DROP

#loopback traffic is accepted
ip6tables -A INPUT -s ::1 -d ::1 -j ACCEPT

#accept new ssh traffic on internal interface and network only
ip6tables -A INPUT -i $INSIDEIF -s $INSIDENET -p tcp --dport 22 -m state --state NEW -j
    ACCEPT
#accept established and related traffic on all interfaces
ip6tables -A INPUT -p tcp --dport 22 -m state --state ESTABLISHED,RELATED -j ACCEPT
#accept ndp messages directed to the router/firewall (this is needed because of the policy)
ip6tables -A INPUT -p icmpv6 --icmpv6-type router-advertisement -j ACCEPT
ip6tables -A INPUT -p icmpv6 --icmpv6-type neighbor-solicitation -j ACCEPT
ip6tables -A INPUT -p icmpv6 --icmpv6-type neighbor-advertisement -j ACCEPT
ip6tables -A INPUT -p icmpv6 --icmpv6-type redirect -j ACCEPT



#accept ssh related and established traffic, and ns,na traffic on output chain
ip6tables -A OUTPUT -p tcp -m state --state RELATED,ESTABLISHED -j ACCEPT
ip6tables -A OUTPUT -p icmpv6 --icmpv6-type router-advertisement -j ACCEPT
ip6tables -A OUTPUT -p icmpv6 --icmpv6-type neighbor-solicitation -j ACCEPT
ip6tables -A OUTPUT -p icmpv6 --icmpv6-type neighbor-advertisement -j ACCEPT
ip6tables -A OUTPUT -j LOG --log-prefix "output drops"
ip6tables -A OUTPUT -j DROP

#drop icmpv6 packets with link-local src/dst address in forwarding chain
ip6tables -A FORWARD -p icmpv6 -d fe80::/10 -j DROP
ip6tables -A FORWARD -p icmpv6 -s fe80::/10 -j DROP
```

```
#drop echo reply with dst multicast address in forwarding chain
ip6tables -A FORWARD -p icmpv6 -d ff00::/8 --icmpv6-type echo-reply -j DROP

#icmpv6 traffic from internal to be forwarded to external
ip6tables -A FORWARD -s $INSIDENET -d $OUTSIDENET -p icmpv6 -j ICMPV6-TO-OUT
#ssh traffic from internal to be forwarded to external
ip6tables -A FORWARD -i $INSIDEIF -o $OUTSIDEIF -s $INSIDENET -d $OUTSIDENET -p tcp --dport
    22 -m state --state NEW,ESTABLISHED,RELATED -j SSH-OUT

#icmpv6 traffic from external to be forwarded to internal
ip6tables -A FORWARD -d $INSIDENET -p icmpv6 -j ICMPV6-TO-IN
#ssh traffic from external to be forwarded to internal
ip6tables -A FORWARD -d $INSIDENET -p tcp --sport 22 -m state --state ESTABLISHED,RELATED -j
     SSH-IN
ip6tables -A FORWARD -j LOG --log-prefix "FORWARDING DROPS"

#---------------------------------
#forwarding rules from IN to OUT -
#---------------------------------

#accept ssh to be forwarded to external network
ip6tables -A SSH-OUT -i $INSIDEIF -o $OUTSIDEIF -s $INSIDENET -d $OUTSIDENET -p tcp --dport
    22 -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
#accept error messages
ip6tables -A ICMPV6-TO-OUT -s $INSIDENET -d $OUTSIDENET -p icmpv6 --icmpv6-type destination-
    unreachable -j ACCEPT
ip6tables -A ICMPV6-TO-OUT -s $INSIDENET -d $OUTSIDENET -p icmpv6 --icmpv6-type packet-too-
    big -j ACCEPT
ip6tables -A ICMPV6-TO-OUT -s $INSIDENET -d $OUTSIDENET -p icmpv6 --icmpv6-type time-
    exceeded -j ACCEPT
ip6tables -A ICMPV6-TO-OUT -s $INSIDENET -d $OUTSIDENET -p icmpv6 --icmpv6-type parameter-
    problem -j ACCEPT
#echo request with rate limit
ip6tables -A ICMPV6-TO-OUT -s $INSIDENET -d $OUTSIDENET -p icmpv6 --icmpv6-type echo-request
     -m limit --limit 900/min -j ACCEPT
#echo reply is dropped because of internal policy
ip6tables -A ICMPV6-TO-OUT -s $INSIDENET -d $OUTSIDENET -p icmpv6 --icmpv6-type echo-reply -
    j DROP
#NDP messages only if they haven’t traversed a router (this is to underline the required hop
     limit of 255)
ip6tables -A ICMPV6-TO-OUT -s $INSIDENET -d $OUTSIDENET -p icmpv6 --icmpv6-type router-
    advertisement -m hl --hl-eq 255 -j ACCEPT
ip6tables -A ICMPV6-TO-OUT -s $INSIDENET -d $OUTSIDENET -p icmpv6 --icmpv6-type neighbor-
    solicitation -m hl --hl-eq 255 -j ACCEPT
ip6tables -A ICMPV6-TO-OUT -s $INSIDENET -d $OUTSIDENET -p icmpv6 --icmpv6-type neighbor-
    advertisement -m hl --hl-eq 255 -j ACCEPT
ip6tables -A ICMPV6-TO-OUT -s $INSIDENET -d $OUTSIDENET -p icmpv6 --icmpv6-type redirect -m
    hl --hl-eq 255 -j ACCEPT
#drop remaining icmpv6 packets (for clarity, but redundant because of the policy)
ip6tables -A ICMPV6-TO-OUT -s $INSIDENET -d $OUTSIDENET -p icmpv6 -j LOG --log-prefix "
    Firewall IN-OUT: dropped ICMPv6"
ip6tables -A ICMPV6-TO-OUT -s $INSIDENET -d $OUTSIDENET -p icmpv6 -j DROP

#--------------------------------
#forwarding rule from OUT to IN -
```

```
#--------------------------------

#accept established and related ssh to be forwarded to internal network
ip6tables -A SSH-IN -d $INSIDENET -p tcp --sport 22 -m state --state ESTABLISHED,RELATED -j
    ACCEPT
#accept error messages ---- TODO: evaluate if it is worth to use state for error msg, and
    only dst addresses
ip6tables -A ICMPV6-TO-IN -d $INSIDENET -p icmpv6 --icmpv6-type destination-unreachable -j
    ACCEPT
ip6tables -A ICMPV6-TO-IN -d $INSIDENET -p icmpv6 --icmpv6-type packet-too-big -j ACCEPT
ip6tables -A ICMPV6-TO-IN -d $INSIDENET -p icmpv6 --icmpv6-type time-exceeded -j ACCEPT
ip6tables -A ICMPV6-TO-IN -d $INSIDENET -p icmpv6 --icmpv6-type parameter-problem -j ACCEPT
#echo request and reply , no ping from outside (internal policy), but allow reply to come
    back with rate limit
ip6tables -A ICMPV6-TO-IN -d $INSIDENET -p icmpv6 --icmpv6-type echo-request -j DROP
ip6tables -A ICMPV6-TO-IN -d $INSIDENET -p icmpv6 --icmpv6-type echo-reply -m limit --limit
    900/min -j ACCEPT
#drop explicitly and log ndp messages
ip6tables -A ICMPV6-TO-IN -d $INSIDENET -p icmpv6 --icmpv6-type router-advertisement -j LOG
    --log-prefix "Firewall OUT-IN: dropped ra"
ip6tables -A ICMPV6-TO-IN -d $INSIDENET -p icmpv6 --icmpv6-type router-solicitation -j LOG
    --log-prefix "Firewall OUT-IN: dropped rs"
ip6tables -A ICMPV6-TO-IN -d $INSIDENET -p icmpv6 --icmpv6-type neighbor-advertisement -j
    LOG --log-prefix "Firewall OUT-IN: dropped na"
ip6tables -A ICMPV6-TO-IN -d $INSIDENET -p icmpv6 --icmpv6-type neighbor-solicitation -j LOG
     --log-prefix "Firewall OUT-IN: dropped ns"
ip6tables -A ICMPV6-TO-IN -d $INSIDENET -p icmpv6 --icmpv6-type redirect -j LOG --log-prefix
     "Firewall OUT-IN: dropped redirect"
ip6tables -A ICMPV6-TO-IN -d $INSIDENET -p icmpv6 --icmpv6-type router-advertisement -j DROP
ip6tables -A ICMPV6-TO-IN -d $INSIDENET -p icmpv6 --icmpv6-type router-solicitation -j DROP
ip6tables -A ICMPV6-TO-IN -d $INSIDENET -p icmpv6 --icmpv6-type neighbor-advertisement -j
    DROP
ip6tables -A ICMPV6-TO-IN -d $INSIDENET -p icmpv6 --icmpv6-type neighbor-solicitation -j
    DROP
ip6tables -A ICMPV6-TO-IN -d $INSIDENET -p icmpv6 --icmpv6-type redirect -j DROP
ip6tables -A ICMPV6-TO-IN -d $INSIDENET -p icmpv6 -j DROP
```

## H. Monitor Windows Firewall - PoC

Listing 5. Monitor Windows Firewall - PoC

```
#!/usr/bin/python

import threading
import time
from scapy.all import *

def packet_callback(packet):
      if ICMPv6ParamProblem in packet[0]:
         adr = packet[IPv6].src
         code = packet[ICMPv6ParamProblem].code
         print code
      print adr

def receiver(iface):
   sniff(iface=iface, filter='ip6', prn=packet_callback, store=0, timeout=20)

class receiverThread(threading.Thread):

   def __init__(self, iface):
      threading.Thread.__init__(self)
      self.iface = iface

   def run(self):
      print "Starting Receiving Packets"
      rec = receiver(self.iface)

class echoSenderThread(threading.Thread):

   def __init__(self, iface, target):
      threading.Thread.__init__(self)
      self.iface = iface
      self.target = target

   def run(self):
      print "Starting sending pongs"
      for x in range(1,100):
         p = IPv6(dst=self.target)/ICMPv6EchoReply()
         send(p, iface=self.iface, verbose=False)
         time.sleep(2)

win = "2001:abcd:acad:2:b485:2aec:9447:fd83"
linux = "2001:abcd:acad:2:a00:27ff:fe84:bb37"
allNodes = "ff02::1"
rec = receiverThread("eth0")
sendEcho = echoSenderThread("eth0", win)

rec.start()
sendEcho.start()
```