2011

Network-aware Active Wardens in IPv6

Grzegorz Lewandowski
Syracuse University

Abstract

Every day the world grows more and more dependent on digital communication. Technologies like e-mail or the World Wide Web that not so long ago were considered experimental, have first become accepted and then indispensable tools of everyday life. New communication technologies built on top of the existing ones continuously race to provide newer and better functionality. Even established communication media like books, radio, or television have become digital in an effort to avoid extinction.

In this torrent of digital communication a constant struggle takes place. On one hand, people, organizations, companies and countries attempt to control the ongoing communications and subject them to their policies and laws. On the other hand, there oftentimes is a need to ensure and protect the anonymity and privacy of the very same communications.

Neither side in this struggle is necessarily noble or malicious. We can easily imagine that in presence of oppressive censorship two parties might have a legitimate reason to communicate covertly. And at the same time, the use of digital communications for business, military, and also criminal

purposes gives equally compelling reasons for monitoring them thoroughly.

Covert channels are communication mechanisms that were never intended nor designed to carry information. As such, they are often able to act "below" the notice of mechanisms designed to enforce security policies. Therefore, using covert channels it might be possible to establish a covert communication that escapes notice of the enforcement mechanism in place. Any covert channel present in digital communications offers a possibility of achieving a secret, and therefore unmonitored, communication. There have been numerous studies investigating possibilities of hiding information in digital images, audio streams, videos, etc. We turn our attention to the covert channels that exist in the digital networks themselves, that is in the digital communication protocols.

Currently, one of the most ubiquitous protocols in deployment is the Internet Protocol version 4 (IPv4). Its universal presence and range make it an ideal candidate for covert channel investigation. However, IPv4 is approaching the end of its dominance as its address space nears exhaustion. This imminent exhaustion of IPv4 address space will soon force a mass migration towards Internet Protocol version 6 (IPv6) expressly designed as its successor. While the protocol itself is already over a decade old, its adoption is still in its infancy. The low acceptance of IPv6 results in an insufficient understanding of its security properties.

We investigated the protocols forming the foundation of the next generation Internet, Internet Protocol version 6 (IPv6) and Internet Control

Message Protocol (ICMPv6) and found numerous covert channels. In order to properly assess their capabilities and performance, we built *cctool*, a comprehensive covert channel tool. Finally, we considered countermeasures capable of defeating discovered covert channels. For this purpose we extended the previously existing notions of active wardens to equip them with the knowledge of the surrounding network and allow them to more effectively fulfill their role.

Keywords: *security, covert channel, IPv6, active warden, IPsec, network-aware, traffic analysis, traffic normalization*

NETWORK-AWARE ACTIVE WARDENS IN IPv6

By
Grzegorz Lewandowski
M.S. Syracuse University

DISSERTATION

Submitted in partial fulfillment of the requirements for
the degree of Doctor of Philosophy in Computer Engineering
in the Graduate School of Syracuse University

August 2011

Contents

List of Tables

List of Figures

Acknowledgments

While the research described in this dissertation is concerned with a sterile, technical environment of network security, it was conducted with assistance, cooperation, guidance and encouragement that came from living, breathing people. The packets, protocols and octets get their due in the following chapters, all that I can offer to the people are my most sincere thanks and acknowledgements.

First and foremost I would like to thank my advisor, Dr. Steve J. Chapin. It is perhaps trivial to say that without him this dissertation would not exist, but it is true nonetheless. His patient mentoring, constant support and challenging questions are what made this work possible. Many thanks to the other members of my dissertation committee, Drs. Wenliang Du, Roman Markowski, Jae C. Oh, Leonard Popyack, and the chair Dr Carlos Caicedo, for their time and valuable feedback.

Special appreciation goes to Dr. Shiu-Kai Chin whose class on principles of network security has shown me how interesting security problems can be, and to Dr. Norka Lucena whose question started me on this investigation

and without whom the investigation would not have gotten very far at all.

I would also like to express my gratitude to the faculty and staff of Department of Electrical Engineering and Computer Science and the Systems Assurance Institute of Syracuse University, whose support and resources allowed me to conduct this investigation. Additional thanks to Peter Morrissey and members of the networking group who patiently dealt with the requirements and results of my network experiments.

Good luck and thanks to my colleagues and friends – Chris Sarmoria, Karthick Jayaraman and Paul Talaga – for sharing that unique experience that is graduate school.

And last but not least, my warmest thanks to Pavlina for standing by me through it all.

Chapter 1

Introduction

A communication channel is a mechanism through which a message is transmitted to its intended audience. Given that messages can potentially be of a sensitive nature, it is not surprising that many communication channels are subject to policies imposing restrictions on the flow of information in the channels. Keeping that in mind, we can divide existing communication channels into two broad categories: *overt communication channels* and *covert communication channels*. Overt channels are acknowledged communication methods that are widely known, and if intended to carry sensitive information, they are subject to communication policies. Obviously, in order to make the policies effective, all communications channels should be overt. However, covert communication channels are communication mechanisms that appear accidentally, they were never intended to be functional communication channels and they exist purely by accident. It is therefore likely that security

polices governing information flow in overt channels will be unaware of the existence of covert channels and the information flow therein will go not only unrestricted but entirely unnoticed. If that is the case, covert channels are able to deliver what cryptography cannot, that is they allow one to send "invisible" messages where not only message content is secret, but the very fact that the message exists is hidden.

Modern computer systems provide excellent examples of complex systems consisting of multiple overt communication channels governed by an assortment of security policies. For example, within virtually any computer there exist multiple processes and while communication between them is certainly possible, it has to be authorized by appropriate security policies. Unfortunately, covert communication channels exist in these systems as well. Despite the presence of security mechanisms, it is often possible to transfer information across process boundaries by cleverly manipulating a shared resource. Sending data directly to another process might be blocked by a security policy, but it might be possible to transmit information by locking a shared file, or by increasing CPU load, as both events are detectable by the other process.

While the covert channels mentioned above allow sending hidden messages within a single computer, covert channels exist in computer networks as well and they provide the ability to secretly communicate between remote computers. Instead of a shared file, or CPU load, network covert channels manipulate network protocols themselves. Since network protocols are ex-

pressly designed for communication and are therefore overt communication channels monitored by respective security mechanisms, the network covert channels operate by making a distinction between protocol's control messages and its payload, and by exploiting the fact that it is generally only the payload that is the subject of security checks. To accomplish its goal, a network covert channel will modify existing network traffic, attempting to hide information with the traffic's protocol format and structure, aiming either for the protocol's control fields or, at least, for the parts of the protocol that are difficult to understand for any observer, e.g. because of employed encryption.

Numerous studies have found multiple covert channels existing in a variety of network protocols. However, given the choice of protocols, which protocol is the most beneficial to exploit for covert communications? Currently, one of the most ubiquitous protocols in deployment is the Internet Protocol version 4 (IPv4). Its universal presence and global range make it an ideal candidate for covert channel exploits. However, IPv4 is approaching the end of its dominance as its address space nears exhaustion [81] that will soon force a mass migration towards Internet Protocol version 6 (IPv6), expressly designed as its successor. Therefore, we turn our attention to network covert channels present in IPv6 protocol.

The IPv6 covert channels present less immediate security threat than well-known attacks such as, for example, buffer overflows. After all, unlike a buffer overflow, a covert channel cannot be used to directly attack and compromise

a remote machine. However, covert channels are a definite security risk, by the virtue of allowing an undetected (and therefore unmonitored) communication. When combined with another form of attack, a covert channel can be particularly damaging as it might, for example, be used to maintain long-term control over a compromised machine residing on a secure network. In this scenario, the attacker could steal information over a long period of time, effectively "multiplying" the impact of the initial security breach.

For example, Clark and Lewin [17] describe attacks performed by embedding hidden code in sensitive hardware that is then sold to the victim. The attacker can then activate the embedded code disrupting the functioning of the hardware with potentially catastrophic results. Clark and Lewin speculate that such attack was carried out against Syrian surveillance radar contributing to the success of Israel's September 2007 bombing raid. Efficacy of this type of intrusion greatly depends on the covertness of the activation mechanism, making covert channel communication an excellent implementation choice.

1.1 Background

The act of covert communication is, by its very nature, a contest between the communicating parties who wish their conversations to remain hidden and another party that attempts to detect the communication or perhaps to disrupt it. Covert channel techniques are concerned with performing unde-

tected communications, while on the other side of the contest, intrusion detection systems, traffic normalizers and active wardens employ various means necessary to uncover or otherwise defeat covert channels. This section surveys the basics of covert channels and their countermeasures, highlighting the information necessary for our investigation of IPv6 covert channels and network-aware wardens.

1.1.1 Prisoners' Problem

In the context of the "classical" prisoners' problem [79], Alice and Bob are two agents who wish to communicate covertly (see Figure 1.1). As described in [57], 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.

The dotted boxes in Figure 1.1 indicate that Alice and Bob could either act as sender and receiver, or could modify the messages in transit [57].

In this framework, Wendy always acts as an active warden [20, 29, 50]. Active wardens can modify the content of the network traffic with the purpose of eliminating any form of hidden communication. When modifying network packets, active wardens should maintain the syntactic and semantic integrity of the packet to avoid breaking the overt communication. They reinforce protocol specifications through mechanisms such as zeroing reserved fields, randomizing ID numbers, and requiring or prohibiting the use of option fields.

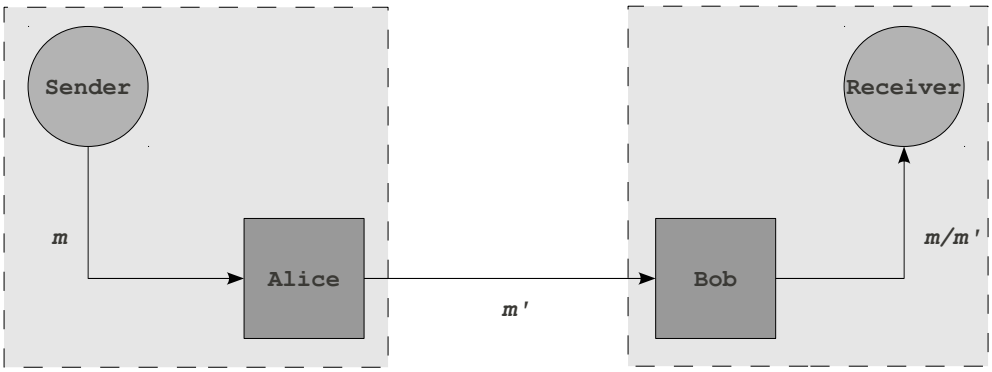

Figure 1.1: Framework for Covert Communication.

1.1.2 Definitions

Covert channel *Covert channels* were first proposed by Lampson [55] in the context of the confinement problem as communication channels that are neither designed nor intended to carry information. The definition was later expanded to include all communication paths that allow information transfer in violation of a system's security policies [87]. In the context of network protocols, covert channel communication is generally achieved by manipulating an overt communication.

Cover traffic *Cover traffic* is the traffic that is being manipulated by covert channel participants. It might originate from one of the participants but it is also possible to "hijack" a 3rd party communication for the purpose of covert communication.

Storage covert channel A *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.

Timing covert channel A *timing covert channel* is a signaling mechanism based on influencing system response times. Again, the definition was initially created to describe an intra-machine channels and subsequently extended to network convert channels. Network timing covert channels rely on modifying timing of network messages to convey information.

Active warden An *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. As much as possible a warden should maintain the syntactic and semantic integrity of the modified traffic to avoid breaking the cover communication.

1.1.3 Countermeasures

The most effective defensive mechanisms against network storage channels for IPv4 are protocol scrubbers [61], traffic normalizers [37], and active wardens [5, 6, 20, 29]. Protocol scrubbers and traffic normalizers focus on eliminating ambiguities found in the traffic stream, carefully crafted with the purpose

of evading network intrusion detection systems. Ambiguous network packets are those which could have different interpretations at endpoints depending on the implementation of the protocol stack. Covert channels are certainly a form of ambiguous traffic. Handley and Paxson [37] describes IP, UDP, TCP, and ICMP normalizations based on protocol specification, highlighting the importance of preserving the end-to-end protocol semantics. In the same order of ideas, active wardens, as presented by Fisk et al. [29], are network services resembling a firewall that modify all traffic under the assumption that it is carrying steganographic content. Active wardens defeat steganography by making semantics-preserving alterations to packet headers (e.g. zeroing the padding bits in a TCP packet). These techniques, although effective for many covert channels, do not record any state or gather network topology information.

1.1.4 IPv6

Internet Protocol version 6 (IPv6) [26] was designed as the replacement for the currently prevalent Internet Protocol version 4 (IPv4). The primary motivation for introducing IPv6 was the foreseen exhaustion of IPv4 address space. Although still in its infancy as far as global deployment is concerned, IPv6 importance and adoption are growing steadily [15, 33, 46, 58, 69, 86, 88]. Since 2007, all major desktop operating systems enable IPv6 by default. In 2008, the six root name servers are accessible via IPv6, making it possible for two IPv6-enabled hosts to perform name resolution and then communicate

without relying on IPv4 at all [41]. In 2010, Verizon [89] and Comcast announced that they are conducting trial runs of IPv6 for their ISP operations. Moreover, the exhaustion of IANA IPv4 address pool occurred as predicted in early 2011 [81] and the exhaustion of RIR pools is expected to begin later in 2011 [40].

The IPv6 header structure has a fixed length of 40 bytes. Several fields from IPv4 were removed (header length, identification, flags, fragment offset, header checksum, and options). Functionality previously provided by the removed fields is now implemented via extension headers. An IPv6 packet can have more than one extension header. IPv6 specification recommends that when present, the headers are layered in order. Six extension headers are defined:

- Hop-by-Hop Options header
- Routing header
- Fragment header
- Destination Options header
- Authentication header (AH)
- Encapsulating Security Payload (ESP) header

All the headers but the last two are defined by RFC 2460 [26], while Authentication and Encapsulating Security Payload headers and their func-

tionality are described in separate RFCs as part of the IP security framework (IPsec): RFC 4301 [53], RFC 4302 [51], and RFC 4303 [52].

1.1.5 IPsec

Internet Protocol Security (IPsec) suite [53] provides traffic integrity protection and traffic confidentiality for Internet Protocol-based communications. When applied to IPv6, IPsec relies on two IPv6 extension headers for its operation. These are: Authentication header [51] and Encapsulating Security Payload [52]. Authentication Header provides packet integrity by including Integrity Check Value (ICV) that is calculated over packet's payload as well as some of the packet's header fields. Encapsulating Security Payload (ESP) can ensure packet's payload confidentiality and optionally integrity. Unlike Authentication Header's, ESP's ICV does protect packet's header but only its payload.

IPsec can secure communication between two hosts using end-to-end *transport mode*, or between two security gateways in network *tunnel mode*. A hybrid scenario is also possible, when a host communicates with a security gateway[1]. In the transport mode, IPv6 traffic is protected by incorporating IPsec headers into its packets. The traffic payload can be encrypted and authenticated, while traffic headers can only be authenticated since they are used by the usual network processing mechanisms. In the tunnel mode, origi-

[1] An intermediate system that implements the IPsec framework, e.g. a firewall implementing IPsec.

nal IPv6 traffic is packaged (encapsulated) into new IPv6 packets and treated as payload. Therefore it can be both encrypted and authenticated.

1.2 Problem

The main purpose of this work is to research the question *"Do network storage covert channels exist in Internet Protocol version 6 and, if they do exist, are existing active wardens an effective countermeasure against them? Additionally, if the currently existing active warden techniques cannot be effectively used to combat IPv6 covert channels, how can active wardens be extended to improve their capabilities?"*.

The investigation comprises the following steps:

1. A survey of IPv6, ICMPv6 and related protocols conducted to establish existence of covert channels.

2. A software tool implemented to verify discovered covert channel viability on both private networks and the Internet.

3. An analysis of possible countermeasures to defeat the identified covert channel exploits.

4. An extension of current active warden techniques to allow the warden to take advantage of its knowledge of the surrounding network to combat covert channels.

1.3 Organization

This dissertation is organized as follows. Chapter 2 presents the results of the investigation of covert channels in Internet Protocol version 6 (IPv6). It lists the examined protocol specifications and discovered channels, and details used covert channels taxonomy and applicable properties. Chapter 3 examines the concept of active wardens, proposes extending of the existing active warden functionality, presents several warden models and lists devised countermeasures for previously described covert channels. Chapter 4 covers the software package that implements both covert channel and active warden functionality. In addition, it presents the results of experiments performed using the described tool. Finally, Chapter 5 summarizes the findings and contributions and also presents the possible direction of future research.

Chapter 2

Covert Channels in IPv6

There exist numerous studies describing covert channels in network protocols. The protocols like IPv4, ICMP and TCP are thoroughly covered, IPv6, however, is not. This chapter outlines the work concerning other network protocols, then describes models and properties used in this study, and presents a comprehensive list of IPv6 covert channels.

The detailed organization is as follows. Section 2.1 describes related studies targeting network covert channels in other protocols, largely focusing on IPv4 and related protocols. Section 2.2 describes the communication model for covert channel attacks. Section 2.3 defines the properties of syntax and semantics preservation, while section 2.4 details handling of checksums included in cover traffic. Section 2.5 presents a list of covert channels discovered in IPv6 and ICMPv6 protocols. Finally sections 2.6 and 2.7 discuss the impact of tunneled traffic and IPsec on covert channel communications.

2.1 Covert Channels in Network Protocols

Existing research in network covert channels [12] encompasses the study of network- and transport-layer protocols such as IP, TCP, ICMP, as well as application-layer protocols, such as HTTP. It focuses on version 4 of the Internet Protocol (IPv4) and the corresponding versions of related protocols: TCPv4, ICMPv4, etc [1, 3, 4, 8, 12, 23, 24, 28, 32, 48, 64, 73, 74, 76, 77, 82, 85]. The majority of the literature discusses network storage channels [1, 8, 23, 24, 28, 32, 48, 73] rather than network timing channels [3, 4, 12, 76, 77], likely because of the synchronization issues present in timing channels and their low bandwidth in comparison to storage channels. It is, however, surprising that given the increasing importance of IPv6, most of the research still concerns IPv4.

Handel and Sandford [36] pioneers covert channels within network communication protocols. It describes different methods of creating and exploiting hidden channels in the OSI network model, based on the characteristics of each layer. Szczypiorski [85] describes a hidden communication system at the data link layer of the OSI network mode that takes advantage of imperfections in the transmission medium, such as interferences and noise. Rowland [73], Dunigan [28], and Rutkowska [74] present examples of implementation of covert channels that exploit header fields of the TCP/IP protocol suite (for IPv4). These three papers focus their attention in the network and transport layers of the OSI network model.

Abad [1] describes how to embed data in the IP checksum using selected hash collisions. The IPv4 checksum can be exploited because the algorithm used to calculate it is susceptible to collision attacks. In IPv6, checksums are calculated by keyed message authentications codes (MAC) based on symmetric encryption algorithms such as DES or on one-way hash functions such as MD5 or SHA-1. One-way hash algorithms will reduce, but probably not eliminate (because of recent MD5 collisions ([49, 91]), the possibility of existence of similar channels in IPv6.

Giffin et al. [32] analyzes a low-bandwidth covert channel that uses TCP timestamps. The channel is based on a modification of a TCP header field, in particular, the low order bit of the timestamp option. In a slow connection, this channel is harder to detect than the ones described in [28, 73] because under such network conditions the low order bit of the timestamp appears randomly distributed facilitating the transmission of encrypted messages.

Ahsan and Kundur [3, 4] proposes five covert channel approaches: four of them based on manipulations of the TCP, IGMP, and ICMP protocol headers and one of them based on packet sorting within the IPsec protocol. The former are storage channels while the latter is a timing channel. The network timing channel works by sorting packets by the sequence number field present in both the authentication header (AH) and the encapsulated security payload header (ESP) defined in IPsec. The hidden information is the difference between the original sequence of packets and the sorted sequence.

Project Loki [23, 24] explores the concept of ICMP tunneling, exploiting covert channels through the data portions of the ICMP_ECHO and ICMP_ECHOREPLY packets. The Loki client allows a remote attacker to wrap and transmit commands in ICMP payloads. Lokid, the Loki server, unwraps and executes the commands, sending the results back wrapped in ICMP packets. Back Orifice 2000 with the BOSOCK32 plug-in also implements covert channels via ICMP. Firewalls can disallow entirely the passing of ICMP traffic, preventing the existence of this kind of tunneling. Project Loki also runs over UDP on port 53, simulating DNS traffic. Sneakin [80] provides an incoming shell through outgoing Telnet-like traffic.

2.2 Communication Model

As outlined in section 1.1.1, the communication model for network storage channels involves two parties, Alice and Bob, 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 (see Figure 1.1).

Different scenarios emerge depending on whether or not Alice and Bob

are the same as the sender and the receiver of the cover traffic. Additionally, if Bob is not the receiver, he might restore the traffic to its original form, or he might allow it to continue to its destination as modified by Alice. Lucena [56] considers six scenarios created by these possibilities. In Figure 2.1, the cover traffic is m, and the modified traffic containing covert communication is m'.

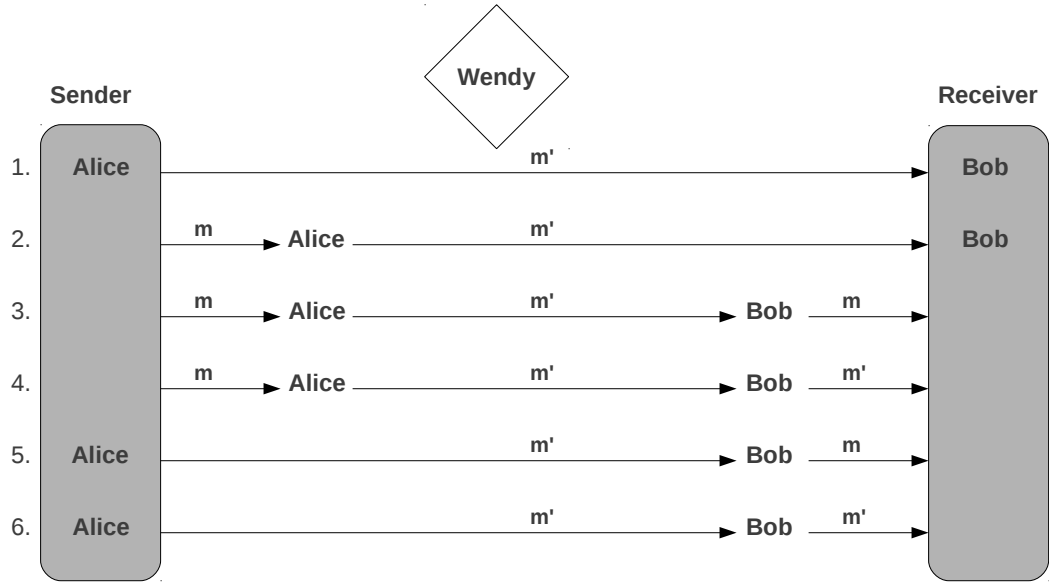

Figure 2.1: Communication Scenarios.

The six pictured scenarios are as follows:

1. Alice is the sender and Bob the receiver – the traffic is m' along its entire path. This scenario is simple as far as communication is concerned, since Alice and Bob do not have to worry about proper positioning, traffic modification and its restoration. However, the scenario requires that Alice and Bob can communicate directly without arousing suspicion.

2. Alice is separate from the sender, she modifies the sender's traffic that is already on its way to Bob who is the receiver – the traffic from the sender to Alice's location is m, while from there to the endpoint it is m'. This scenario requires that Alice is positioned properly to intercept sender's communication with Bob. Alice also might have to contend with traffic security in order to modify it without attracting attention.

3. Both Alice and Bob are separate from the sender and receiver, and Bob restores the traffic to its original form – the traffic from the sender to Alice's location is m, from Alice's to Bob's it is m', and from there to the endpoint it is m again, because extraction of the hidden content and restoration of the original cover traffic occurred at Bob's location. This scenario is more complex than the previous ones since both Alice and Bob have to position themselves correctly to be able to see the traffic exchanged between the sender and the receiver. The traffic security mechanisms can be a concern as well. On the other hand, the scenario offers potentially the best secrecy as neither Alice nor Bob are participants in the overt communication. Moreover, Bob restores the modified traffic m' to its original form m, minimizing its exposure to any observer.

4. Both Alice and Bob are distinct from the sender and the receiver respectively, but Bob does not restore the traffic to its original form – the traffic from the sender's location to Alice's is m, and from Alice's to

the receiver's location it is m'. This scenario is similar to the previous one, except that the modified traffic m' continues past Bob's location through the rest of the traffic's path. This results in the greater exposure of the covert communication to any observer positioned between Bob and the receiver. Additionally, Alice has to take into account the effect that her modifications might have on the cover communication when the m' traffic reaches its final destination.

5. Alice is the sender with Bob being separate from the receiver and extracting the embedded information and restoring the original traffic – the traffic from the initial point to Bob's location is m', and from Bob's location to the receiver's it is m. In this scenario it is easy for Alice to embed the covert messages into the traffic as she is the sender. On the other hand, Bob has to position himself so that he can monitor traffic between Alice and the receiver.

6. Alice is the sender and Bob is separate from the receiver and does not restore the cover traffic – the traffic from end to end is m'. Similarly to the previous scenario, Alice has easy access to the traffic and Bob has to worry about proper positioning to intercept the cover traffic. Unlike in the previous scenario, Alice has to be concerned about the impact of her traffic modifications when the traffic reaches its final destination.

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.

The scenarios listed above demonstrate the trade-offs inherent in covert channel communication. If Alice and Bob elect to generate their own cover traffic, they benefit by having ready access to the traffic thus entirely avoiding problems related to their positioning. Alice can easily modify the traffic, indeed she might not even need to modify the cover traffic at all, since she can simply generate the cover traffic with the covert message already embedded. Furthermore, because Bob serves as the receiver, there is no need for undoing Alice's modifications. In fact, if both Alice and Bob are the sender and the receiver respectively, the cover traffic has no function outside of being the cover for their communication. Consequently, 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.

On the other side of the trade-off, Alice and Bob might forego generating cover traffic and attempt to "piggyback" on someone else's communication. The main benefit of this scenario is that Alice and Bob are not obviously communicating in any visible way and therefore the secrecy of their interchange is improved. At the same time they have to contend with several problems that otherwise would not arise.

- Alice and Bob have to position themselves so that they can intercept a

suitable communication. Moreover they will not be able to control the timing of their communication and might have to wait for it to occur.

- Alice and Bob might have to overcome cover traffic security mechanisms intended to protect against modifications or to obscure traffic content.

- Alice and Bob have to consider the impact their modifications might have on the cover traffic. If the modifications are disruptive, a degradation of network performance will likely occur and might attract attention to the covert communication. If Alice and Bob wish to avoid disruptions, either Alice has to limit herself to "safe" modifications or they have to rely on Bob's ability to restore traffic to its original form which becomes the key factor. It is possible that the covert channel embedding performed by Alice alters the traffic to such degree that Bob, while being able to read m' and understand the covert communication, does not have enough information to convert m' back to m. For example, Alice might overwrite original value of a protocol field, and while Bob can read the modified value and understand the covert message, he might not know the original value and therefore be unable to restore m' to m.

Following sections present a more methodical treatment of traffic modification, their impact as well as mechanisms Bob might employ to aid in traffic restoration.

2.3 Syntax and Semantics Preservation

Previous work in application-level protocol steganography [56] has defined the concepts of syntax and semantics preserving steganography. The definitions can be applied to covert channels as well, and in this context the property of *syntax preservation* determines whether the modified traffic m' still adheres to the protocol syntax. Note that under this property the semantics of m' can still be different than m. 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. It is important to note that if traffic modification performed by Alice does, in fact, violate protocol syntax or semantics, the whole communication can still be syntax or semantics preserving provided that Bob is able to restore protocol compliance and that any outside observer is unable to spot that m' lacks proper syntax or semantics. Intuitively, the property of semantics preservation is stronger and implies syntax preservation as altering protocol syntax seems to ensure damaging its semantics as well.

2.3.1 Location-based Syntax and Semantics Preservation

As described above, the concepts of syntax and semantics preservation are sufficient to describe point-to-point protocols or, more generally, the proto-

cols used between entities whose understanding of the protocol's syntax and semantics is identical.

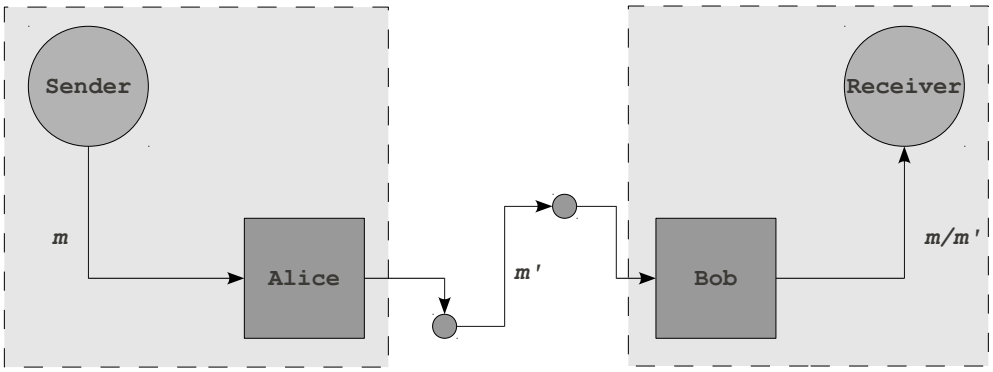

Figure 2.2: Complex Topology.

However, this binary distinction between preservation and non-preservation does not suffice for protocols with more complex topology like Internet Protocol version 6. To achieve the best performance, IPv6 carefully defines multiple levels of protocol knowledge and understanding that different network nodes performing distinct functions are required to implement. Moreover, for efficiency reasons IPv6 specification insists that nodes do not examine packets past what is necessary to perform the node's function.

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. Thus it is necessary to expand the existing notions of syntax and semantics preservation by introducing the *level* of preservation required by various nodes along the packet's path.

More formally, we will say that network nodes can possess certain level of

syntax or semantics *understanding*. A node possessing a level x of syntax or semantics understanding performs, respectively, *level x syntax or semantics integrity check* simply by processing a packet. If a packet modified by a covert channel successfully passes the level x check, we will declare that the covert channel in question is *x-level syntax or semantics preserving*.

An interesting case is presented by protocol's *reserved* fields. Since their value is fixed (usually zeroed), and it is supposed to be ignored by the receiver, they do not carry any meaning and modifying such field does not alter packet's semantics. If the modification in question avoids changing packet's syntax as well, the reserved field is ideal for the purpose of embedding covert messages. And indeed, many network covert channel investigations focus on network protocol's reserved fields and find them useful for covert communication.

The levels of syntax and semantics knowledge applicable to IPv6 protocol as well as examples of network devices possessing given levels are listed below.

none (*none*) a node does not have any knowledge of IPv6 syntax nor semantics, arbitrarily mangled packet should be able to pass this check; example: lower-level protocol devices

forwarding node (*forward*) a node has enough knowledge of IPv6 to perform packet forwarding; it can process and understand the basic syntax of IPv6 header and IPv6 header extensions; it understands semantics of IPv6 header and Hop-by-Hop Options header; example: gateway nodes

current destination (*dest*) a node is listed as packet's current destination, but it is not the final destination[1]; the node will process the packet until it can establish the final destination; it understands all IPv6 syntax and semantics that a *forwarding node* does, additionally it can understand IPv6 Routing header; the example: a node listed in a Routing header of a source-routed packet

final destination (*final*) a node that is packet's final destination; it will process the entirety of the packet, including all present extension headers, extract its payload and pass it to the higher-level protocol processor

IPsec-enabled final destination (*IPsec*) a *final destination* node that is a part of IPsec security association; in addition to all capabilities of a final destination node, an IPsec-enabled final destination node participates in an IPsec-protected communication; it can perform IPsec-based verification of received packets; it understands Authentication and Encapsulating Security Payload headers and have negotiated access to an appropriate security association

An important observation is that while any packet will likely undergo a number of different level integrity checks, and while it has to successfully pass all of them in order to reach its destination, it is only necessary to maintain syntax and semantics preservation of the level of the next check. Therefore

[1]The final destination could be listed in a Routing Header. Alternatively some form of address translation might also cause the current destination to be different from the final destination

a packet can successfully complete its transit despite having only a low level of preservation along some parts of its route as long as it is "repaired" at the right moment, i.e. before a stricter check takes place. This scenario is illustrated in Figure 2.3, if nodes X and Y are not knowledgeable enough to realize the m' has a low level of semantics preservation, the communication will succeed because the node B (Bob) can restore the correct protocol semantics.

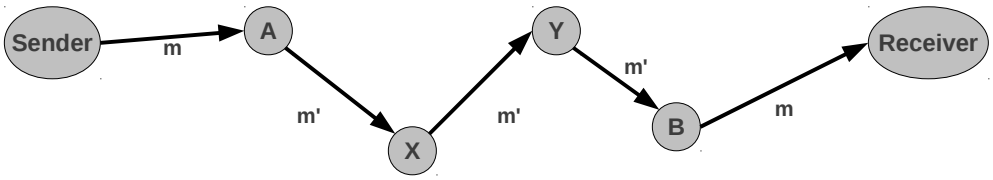

Figure 2.3: Varying level of syntax and semantics preservation.

As a corollary to the previous observation, the objectives of covert communications' participants can be defined in terms of syntax and semantics preservation.

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. It is assumed that the original cover traffic m is fully compliant with IPv6 syntax and semantics and therefore the only concern of the communicating parties is syntax and semantics preservation

of the modified traffic m'.

Recalling the communication scenarios presented above, it is clear that Bob's ability to restore modified traffic m' to its original form m is an important factor in determining the level of packet alterations that Alice can make. If Bob can restore the traffic, Alice's modifications have to only successfully traverse the network nodes (and their associated integrity checks) that are located between her and Bob. Were Bob unable to restore the traffic, Alice's changes would persist until the traffic reached its final destination and it is usually there that the strictest integrity checks take place.

There are several reasons why Bob might be unable to undo Alice's modifications, they are briefly presented here and discussed in more detail in their respective sections below.

- Traffic modified by Alice might be protected by IPsec when it is passing by Bob's location and since IPsec is designed to detect traffic modifications, any changes that Bob might make will cause IPsec to discard the traffic; see Section 2.7.

- Alice's modifications might have altered the traffic to such degree that Bob is unable to deduce its original form. While the restoration algorithm does primarily depend on a covert channel in question, a generic restoration method is proposed in Section 2.4.

- An incorrect position, a change in network routing, or simply an outage can prevent Bob from ever seeing Alice's messages. If Bob indeed

never sees packets modified by Alice, obviously he cannot undo the modifications she made.

2.4 Upper-layer Checksums

While IPv6 header itself does not contain a packet checksum, many commonly used upper-layer protocols do. For example ICMPv6 headers include a checksum that covers both ICMPv6 header and IPv6 pseudo-header in its computation. Consequently, Alice's and Bob's modifications might alter the protected fields and invalidate the checksum. Other protocols that have similar mechanisms to ICMPv6 are commonly used protocols like TCP and UDP.

These checksums are not security mechanisms and can be freely recomputed and rewritten by both Alice and Bob. Therefore preserving the synchronization between the checksum and the current content of the packet is a trivial task for them. However, Alice might consider leaving the checksum in its original value to help Bob with packet restoration.

2.4.1 Brute force-based Packet Restoration

As explained previously, Bob's ability of restoring traffic to its original shape, or at least to a higher level of syntax/semantics preservation, is important for success of covert channel communications. Generally, whether Bob is able to perform the restoration, and specific recovery algorithm, depends on the

covert channel in question. However, there exists a generic method of packet restoration, based on reverse engineering packet's checksum.

Bob can attempt to use a brute force approach for recovering original values of packet fields. Using this method, Bob will try all possible values of the fields Alice has altered and compute all respective checksum values. When he discovers a value whose checksum matches the original packet's checksum, he will have discovered the original packet content. The easiest case would be when Alice uses a field that is a boolean flag where Bob would only had to compute the checksum once to know what was the original flag value. Even in case of 8 bit wide field, Bob is required to compute the digest at most 255 times.

When using this approach for packet restoration, Bob has to account for the possibility of *checksum collisions*. While trying possible field values, Bob might encounter a situation where there exists more than one field value that results in the correct checksum value. The probability of a collision directly depends on the checksum calculation algorithm. Upper-layer protocols like ICMPv6, TCP and UDP employ a 16-bit one's complement summation [11] for calculating their respective checksum values. In this case, if Alice's modifications are entirely contained within a single 16-bit input to the checksum calculation, there will not be any collisions as there exist only a single value resulting in the checksum match. However, if the introduced packet alterations belong to two or more 16-bit checksum inputs, Bob will always find multiple combinations that result in the correct checksum value. In

consequence, the brute force-based restoration approach has limited use for communication schemes where Alice uses multiple covert channels or when she modifies fields wider than 16 bit.

Subsection 2.7.3 considers viability of analogous approach when applied to IPsec Integrity Check Values (ICVs).

It is important to note that leaving the original packet's checksum value has an impact on packet's semantics-preservation level. The final destination node will certainly verify any checksums present in the packet and it is likely that the modified packet will not pass this verification. As a result, a modified packet with an unaltered checksum will have a semantics-preservation level lower than *final destination*. Still, if the covert channel itself brings the packet's semantics-preservation level below *final destination* level, the easier packet restoration provided by the unaltered checksum might be worthwhile.

2.5 Covert Channels

The protocols and specifications covered by this investigation are listed in Table 2.1.

protocol	**RFC**
IPv6	RFC 2460 [26]
ICMPv6	RFC 4443 [19]
Multicast Listener Discovery (MLD) for IPv6	RFC 2710 [25]
Neighbor Discovery (ND) for IPv6	RFC 4861 [65]

protocol	**RFC**
Router Renumbering for IPv6	RFC 2894 [21]
IPv6 Node Information Queries	RFC 4620 [22]
Extensions to IPv6 Neighbor Discovery for Inverse Discovery	RFC 3122 [18]
Multicast Listener Discovery Version 2 (MLDv2) for IPv6	RFC 3810 [90]
Mobility Support in IPv6	RFC 3775 [47]
SEcure Neighbor Discovery (SEND)	RFC 3971 [7]
Experimental Mobility Protocols	
Multicast Router Discovery	RFC 4286 [35]
Mobile IPv6 Fast Handovers	RFC 5568 [54]

Table 2.1: Investigated protocols.

2.5.1 Properties

Properties of covert channels important to this investigation are:

- degree of packet alteration – syntax- and semantics-preservation level of altered packets. As explained in section 2.3, the actual preservation level can vary during packet's transit.
- checksum violations – whether packet alteration violates any checksums contained in the packet. IPv6 itself does not contain any checksums,

but upper-layer protocols it transports (e.g. UDP, TCP) might include checksums. Additionally, packets protected by IPsec carry secure checksums used for integrity protection.

- possibility of packet restoration – whether Bob can determine and restore the original packet content. Modifications introduced by Alice might override and destroy original values of packet fields. As described previously, Bob's ability to restore the original traffic form has implications on covert communication's performance and secrecy.

- communication range – an important feature of IPv6 covert channels is that IPv6 protocol has practically unlimited range, therefore covert communications "piggybacking" on IPv6 traffic can be conducted globally. This is in contrast to protocols like, for example, Ethernet, where communication range is severely limited. However, some of the protocols using ICMPv6 framework are designed to operate on a single network segment, thus significantly reducing their usefulness as a cover for covert channel communication.

- channel bandwidth – amount of data that can be transfered in given covert channel per packet of cover traffic.

- channel noise – some fields within IPv6 packets might be modified as the packets traverse the network. If Alice uses the same fields for the purpose of covert channel communication, the additional modifications will, in effect, introduce noise into the channel.

As noted previously, their potential for unlimited communication range is an important feature of IPv6 covert channels. Therefore, this study focuses on covert channels with a communication range greater than a single network segment. This requirement has several consequences concerning other channel properties.

First, several of the IPv6-related protocols listed above 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. Even if Alice succeeds in sending a cover packet to an address outside of its originating network segment, the communication can be easily defeated by simple address-based filtering mechanism. The single-segment protocols are thus not analyzed in detail and only briefly mentioned. These are: Multicast Listener Discovery, Neighbor Discovery, Inverse Discovery, Multicast Listener Discovery v2, Secure Neighbor Discovery, Multicast Router Discovery.

Second, to achieve a range greater than a single network segment, a packet carrying covert message must successfully transit between different segments, which implies being processed by a forwarding node. A node will only forward a packet whose syntax and semantics-preservation levels are greater or equal to *forwarding node* level. Consequently, the covert channels with preservation levels lower then *forwarding node* are outside the scope of this study.

While covert channels relying on modifications altering packets' syntax certainly exist, they do not provide sufficient secrecy to be covered in detail by this study. Protocol syntax is relatively straightforward to enforce and such

enforcement will defeat the syntax-altering covert channels. The mechanism is described in Section 3.4.1, and the covert channels are omitted from the list below.

The list largely focuses on semantics-preserving covert channels, with the preservation level of *final destination* or better. When channels of a lower level are presented, a special attention is given to describing method by which Bob can revert the channel traffic to its original form.

Protocols defined by experimental track RFCs that do not define Internet standards are outside of the scope of this study. These are: Node Information Queries and Experimental Mobility Protocols.

2.5.2 IPv6 Header

Figure 2.4 shows the fields in the IPv6 header as well as the plausible covert channels observed.

ID	**Field**	**Covert Channel**	**Bandwidth**
1	*Traffic Class*	Set a false traffic class	8 bits/packet
2	*Flow Label*	Set a false flow label	20 bits/packet
3	*Payload Length*	Increase value to insert extra data	Varies
4	*Next Header*	Set a valid value to add an extra extension header	Varies
5	*Hop Limit*	Increase/decrease value	$\approx$ 1 bit/packet
6	*Source*	Set a false source address	16 bytes/packet

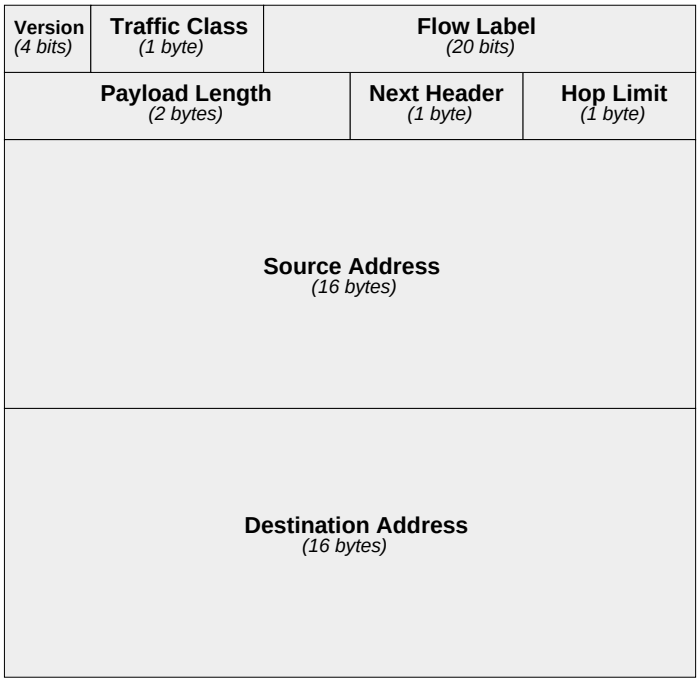

Figure 2.4: IPv6 Header Format.

Field	*Traffic Class*	**RFC**	2460
ID	1	**SPL**	*IPsec*
***cctool* ID**	TC	**checksums**	—

Alice can set a false *traffic class* value. The bandwidth of this channel varies up to 8 bits per packet, depending on whether or not the field is modified by intermediate nodes. The IPv6 specification allows the intermediate nodes to change the value of the traffic class field as they forward the packet. For example, Differentiated Services traffic conditioner [67] might modify the traffic that passes through it. Therefore, when Alice and Bob communicate using this covert channel, they have to be prepared to handle noise. To resist

the effect of in transit modifications, Alice and Bob have to employ an error correction mechanism that will likely reduce the available bandwidth.

Field	*Flow Label*	**RFC**	2460
ID	2	**SPL**	*IPsec*
***cctool* ID**	FL	**checksums**	—

Fabricating a *flow label*, Alice can send 20 bits of data per packet. Authentic flow labels are pseudo-randomly and uniformly selected numbers, ranging from 1 to 0xFFFFF, Alice should try to preserve the same conditions when creating a fake flow label.

Field	*Payload Length*	**RFC**	2460
ID	3	**SPL**	*forward*
***cctool* ID**	PL	**checksums**	AH ICV, pseudo-header

Alice can increase the value of the *payload length* and append extra data at the end of the packet. The bandwidth of this channel varies depending on the size of the original packet, but the modified packet cannot be larger than 65536 bytes. When modifying traffic en route Alice and Bob should also take care not to exceed the current path MTU. Since IPv6 intermediate nodes do not fragment packets, exceeding MTU will cause the packet to be dropped. If encryption is used without authentication, stego techniques like the ones described in [57] are appropriate. If authentication is used, Alice

and Bob need to take extra steps to maintain the covertness of the channel because the payload length is included in the calculation of the AH integrity check value (ICV).

Field	*Next Header*	**RFC**	2460
ID	4	**SPL**	*forward*
***cctool* ID**	NH	**checksums**	AH ICV

Because extension headers other than Hop-by-Hop Header are not examined nor processed by intermediate nodes of a communication path, Alice can change the *next header* content to insert an entire extension header covertly. This channel will, obviously, require that Alice increases the payload length accordingly. The bandwidth of this channel depends on the total length of the extension header inserted. An end-point node that does not recognize the value in the *next header* field[2] will discard the packet and send an ICMP notification to the source. Alice and Bob could potentially use the ICMP reply as a means of covert communication. Alternatively, if Bob wants to restore the traffic to its original form, he has to only strip the inserted extension header from the packet.

Field	*Hop Limit*	**RFC**	2460
ID	5	**SPL**	*IPsec*
***cctool* ID**	HL	**checksums**	—

[2]The Protocol Numbers document [42] lists of all possible *next header* field values.

Alice can initiate a covert communication channel by setting an initial *hop limit* value, h, and manipulating the *hop limit* value of subsequent packets. Bob interprets the covert message by checking the variations in the hop limit values of packets traversing his location. One scheme has Alice signaling a 0 by decreasing the hop count from the prior packet, and a 1 by increasing the hop count relative to the prior packet. A drawback of this channel is that packets do not necessarily travel the same route, so the number of intermediate hops may vary, introducing noise. To overcome this, Alice can choose a δ that is greater than the expected noise, and use hop counts less than $h - \delta$ signal a 0, and hop counts greater than $h + \delta$ to signal a 1. Bob then compares the received hop count to h to deduce the bit. The bandwidth of this channel is limited. Alice needs to modify n packets to send $n - 1$ bits of information.

Field	*Source Address*	**RFC**	2460
ID	6	**SPL**	*dest*
***cctool* ID**	SA	**checksums**	AH ICV, pseudo-header

Alice can forge the source address field to send 16 bytes of covert data. However, the existing mechanisms designed to detect source address spoofing are likely to discard packets modified in such way.

2.5.3 Hop-by-Hop Options Header

The Hop-by-Hop Options header carries optional information that needs to be checked by every node the packet traverses. Because of its different option types, both defined and undefined, and its variable length, this extension header offers possibilities for high-bandwidth covert channels. As described in the protocol specification [26], the *option type* field is an octet structure that has three subfields: the first two bits specify what action should be taken when an unrecognized option is received; the next bit determines whether or not the option data can change en route; the last five bits represent the option number [3]. The analysis introduced below discusses relevant types of option such as the padding, jumbogram, and routing alert options (see Figure 2.5). When authentication is used, the covert channels in the hop-by-hop header may require recalculating or circumventing the ICV (see discussion in section 2.7).

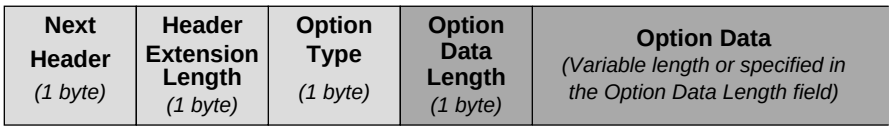

Figure 2.5: Format of the Hop-by-Hop Options Header.

[3] These last five digits are also called "option type" or "rest". However, the option type is fully specified only when using the entire octet.

ID	Field	Covert Channel	Bandwidth
HBH.7	*Option Type: PadN*	Set a false padding value	Up to 256 bytes/packet
HBH.8	*Option Type: Unknown*	Fabricate one or more options	Up to 2038 bytes/packet
HBH.9	*Option Type: Jumbogram*	Insert or create a jumbogram	Varies
HBH.10	*Option Type: Router Alert*	Set a false router alert	2 bytes/packet

Field	*PadN*	**RFC**	2460
ID	HBH.7	**SPL**	*final*
***cctool* ID**	HBHOHPAD	**checksums**	AH ICV

Individual options in the option data field need to preserve header alignment. Two types of padding are defined for that: Pad1 and Pad$N$. Pad1 inserts a single octet, Pad$N$ appends two or more bytes as an individual option type. Alice can exploit any of the padding types, but this channel focuses only in the Pad$N$ *option type*. A simple form of using this option is to embed covert data in an already-existing padding. The bandwidth of that channel will depend then in the length of the padding option. A more crafted way would be inserting a padding option when the header does not contain one. Alice could send this way up top 256 bytes/packet because the Pad$N$

option has a maximum length of 256 bytes. The last alternative, illustrated in Figure 2.6, requires modification of the IPv6 payload length.

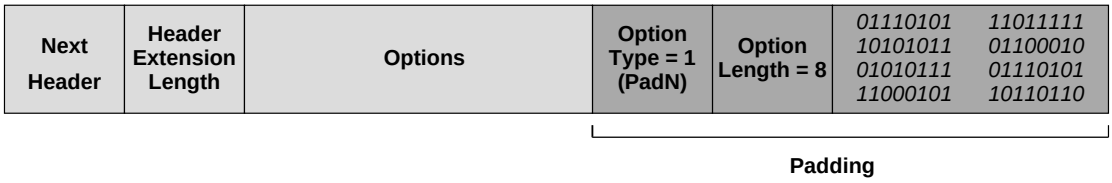

Figure 2.6: Covert Channel in the Hop-by-Hop Options Header.

Field	*Unknown Option*	**RFC**	2460
ID	HBH.8	**SPL**	*final*
***cctool* ID**	HBHOHFAKE	**checksums**	AH ICV

Alice can fabricate an option type, different from the ones listed in [44], as long as she maintain the semantics of the field described at the beginning of the subsection. She needs to make the first two bits of the *option type* equal to 00. That will instruct intermediate nodes to "skip and continue processing" when they do not recognize the option type [26]. The maximum length of option data is 256 bytes. Therefore, up to 256 bytes of covert data can be inserted that way. Moreover, because the hop-by-hop header can include many options, by repeating the insertion with different option type values, up to 2,038 bytes can be added in total[4]. Inserting new options increases the total length of the IPv6 packet.

[4]The length of the header payload is 2054 bytes, which can be filled by 7 options carrying 256 bytes each and 1 option of 246 bytes considering that the headers of individual options will require 16 bytes.

Field	*Jumbogram Option*	**RFC**	2675
ID	HBH.9	**SPL**	*forward*
***cctool* ID**	HBHOHJUMBO	**checksums**	AH ICV, pseudo-header

Jumbograms [10] are IPv6 packets with payload length longer than 65535 bytes. Alice can use jumbograms as a means of covert communication in two ways. The first one relies on modifying an existing jumbogram length with the purpose of appending covert data and it is a jumbogram equivalent of channel 3 in IPv6 header. The second method involves converting a regular datagram into a jumbogram and filling in the extra bytes with hidden content. The first method requires changing *Jumbo Payload Length* field in the option, while the second method requires insertion of a new option including new payload length and setting the payload length in the IPv6 header to 0. Since jumbograms are discarded by intermediate nodes that do not support them, Alice and Bob need to make sure that all nodes in the communication path understand jumbograms.

An additional consideration is that intermediate nodes will discard jumbograms that are smaller than 65536 bytes, consequently the path MTUs has to be greater than 64KB.

Field	*Value*	**RFC**	2711
ID	HBH.10	**SPL**	*final*
***cctool* ID**	HBHOHROUTER	**checksums**	AH ICV

Router alert options contain a 2-byte *value* field where Alice can embed data to establish a covert communication. Alice could also add an entire router alert option type, if it does not exist. That alternative will require readjustment of the packet length in the IPv6 header.

An important consideration when using this channel is that the router alert option by design increases the scrutiny of the packet containing it.

2.5.4 Routing Header

The Routing header contains a list of intermediate nodes a packet in transit should visit on the way to its destination. The IPv6 Parameters document [44] enumerates different types of routing, but only one of them, *Type 0*, is fully described in the specification [26]. Figure 2.7 shows the format of the Routing header when routing type is 0 and its possible channels.

The header was deprecated in RFC 5095 [2] where it was recommended that all IPv6 nodes treat the header as an unknown routing header type. The deprecation does not affect the covert channels directly except if implemented via ingress filtering.

ID	Field	Covert Channel	Bandwidth
RH.11	*Routing Type: 0 - Reserved*	Hide data in unused bits	4 bytes/packet
RH.12	*Routing Type: 0*	Set one or more false addresses	Up to 2048 bytes/packet

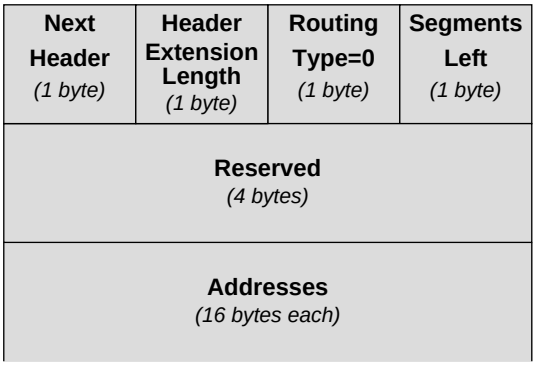

Figure 2.7: Format of the Routing Header

Field	*Reserved*	**RFC**	2460
ID	RH.11	**SPL**	*final*
***cctool* ID**	RHRESV	**checksums**	AH ICV

There exists a *reserved* field in routing header structure when the *routing type* is 0. Alice can hide 4 bytes of covert data per packet using this channel.

Field	*Address*	**RFC**	2460
ID	RH.12	**SPL**	*final*
***cctool* ID**	RHADDR	**checksums**	AH ICV

When the *routing type* is 0, Alice can fabricate "addresses" out of arbitrary data meaningful to Bob. She appends the covert data and sets the *segments left* field to 0 to prevent any node to attempt processing the fake addresses. Figures 2.8 and 2.9 display two different types of embedding:

- one where Alice chooses to create a new Routing header of routing type 0 to send Bob 48 bytes of covert information
- another one where she takes advantage of an already existing Routing header of routing type 0 to embed a covert message of 32 bytes.

Based on the maximum extension header payload length, Alice can potentially insert up 2048 bytes. Therefore, she will be extending the entire IPv6 packet by the same amount of bytes.

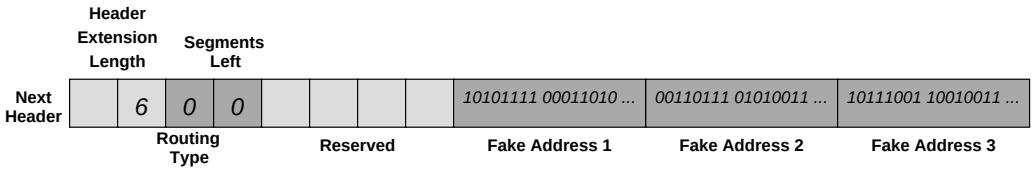

Figure 2.8: Covert Channel in the Routing Header, when Alice creates fake addresses in a packet that did not originally a routing extension header.

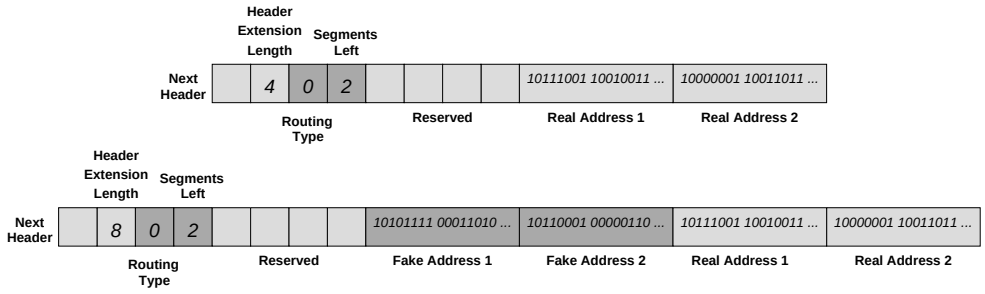

Figure 2.9: Covert Channel in the Routing Header, when Alice inserts fake addresses in a packet already containing a routing extension header. (a) Original routing extension header, (b) Routing header after Alice inserts the covert data.

Type 2 Routing Header is described by RFC 3775 [47] as part of mobility support in IPv6 specification. It is fundamentally similar to Type 0 Routing Header except that it can contain only one address. As a consequence *segments left* field of Type 2 Routing Header is always equal to 1 and the size of the header is fixed.

Consequently, channel RH.11 described above is present in Type 2 header in exactly the same form as in Type 0 header.

Since the address list contained within Type 2 Routing header always contains only one address, channel RH.12 works differently. If Alice wants to use the address field to send data, she has to overwrite the existing address. In order to preserve cover traffic integrity, Bob has to have means to revert to the original address or the cover traffic will be misrouted.

2.5.5 Fragment Header

As in IPv4, fragmentation of packets occurs when the MTU of a link is not large enough to handle a packet of a particular size. Unlike IPv4, IPv6 packets are not fragmented by routers along the path. Instead, sending nodes use path MTU discovery to determine the allowed maximum packet size on the way to a specific destination and fragment packets accordingly. Packets are reassembled only by destination nodes.

There are several important considerations regarding fragmented packets. First, fragments themselves are IPv6 packets, thus all previously described covert channels exist in the fragments as well. In addition, because the

number of packet fragments is obviously greater than the original number of packets sent by a host, the opportunities for information hiding increase accordingly. With that in mind Alice can refragment a packet solely to increase the bandwidth of existing covert channel that does not involve the fragment header.

Second, new covert channels appear when a large packet is fragmented.

Third, IPsec security mechanisms apply only to whole packets, so any modification to packet fragments cannot be detected before the packet is reassembled and in some cases is actually discarded during the packet reassembly and therefore undetectable by IPsec integrity protection.

Figure 2.10 displays the format of the Fragment Header and its potential covert channels.

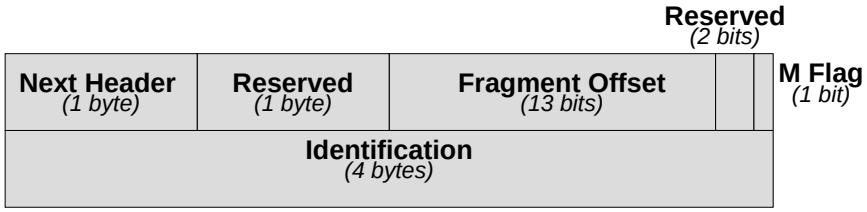

Figure 2.10: Format of the Fragment Header.

ID	Field	Covert Channel	Bandwidth
FH.13	*Reserved*	Hide data in the unused bits	8 bits/packet
FH.14	*Reserved*	Hide data in the unused bits	2 bits/packet
FH.15	*Next Header*	Set a false next header	At least 8 bits/fragment
FH.16	*All*	Insert an entire fake fragment	Up to 64 KB/fragment

Field	*Reserved (8 bits)*	**RFC**	2460
ID	FH.13	**SPL**	*IPsec*
***cctool* ID**	FHRESV	**checksums**	—

Alice can transmit 8 bits of covert data using the first *reserved* field of the header. This field is initialized to zero by the sending host, but it is ignored by the destination. Moreover, the Fragment header is entirely discarded before IPsec processing takes place, therefore the modification is invisible to AH ICV calculation.

Field	*Reserved (2 bits)*	**RFC**	2460
ID	FH.14	**SPL**	*IPsec*
***cctool* ID**	FHRESV2	**checksums**	—

2-bit *reserved* field has the same treatment as the 8-bit reserved field, so Alice can exploit it taking a similar approach.

Field	*Next Header*	**RFC**	2460
ID	FH.15	**SPL**	*IPsec*
***cctool* ID**	FHNEXT	**checksums**	—

The reassembly process at the destination node takes into account only the *next header* value of the first fragment and it ignores the *next header* values of subsequent fragments. Those conditions give Alice the opportunity to embed 8 bits of covert data per fragment as long as she keeps the next header value of the first fragment untouched. The total bandwidth of the channel depends then on the number of fragments. Nonetheless, it is possible to achieve higher bandwidth by refragmenting the fragmented packet into a larger number of fragments. For example, a fragmented packet composed of 3 fragments will allow Alice to send 2 bytes of covert data. If she refragments the packet into 10 fragments, she will increase the bandwidth by 7, for a total of 9 bytes.

Field	*Fragment Header*	**RFC**	2460
ID	FH.16	**SPL**	*IPsec*
***cctool* ID**	FHFAKE	**checksums**	—

Alice can potentially insert an entire fragment exploiting *all* fields of the fragment header. To avoid having this fragment included in the reassembly of the original packet, she can assign an invalid fragment ID field, so that the receiver will discard it. The bandwidth of this channel depends on the size of

the fragment. Figure 2.11 shows a graphical representation of this channel.

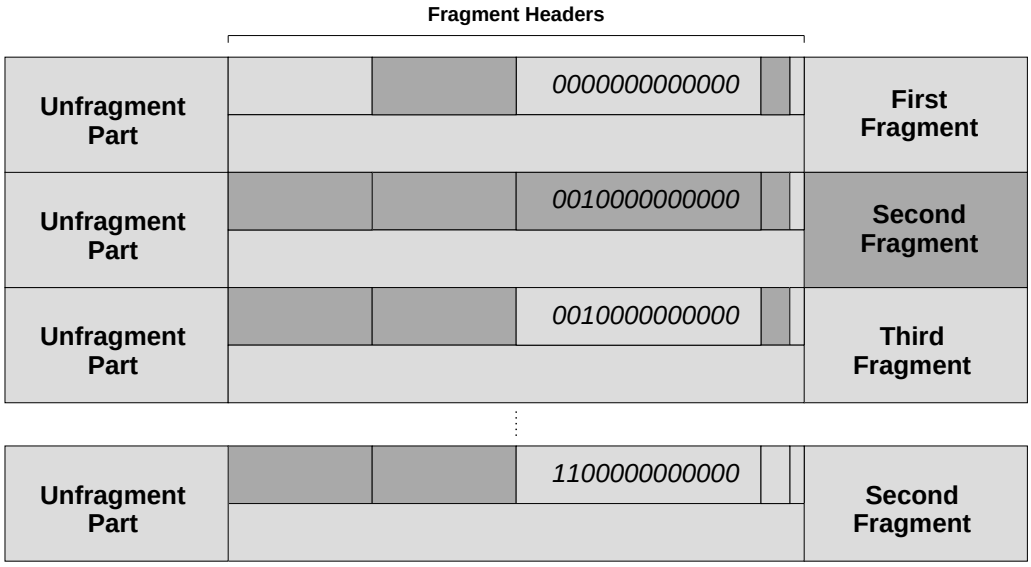

Figure 2.11: Covert Channel in the Fragment header. Alice inserts a fake fragment in the fragments stack, setting a *fragment offset* value that causes its data to be overwritten in reassembly.

There is also a possibility to fragment packets that do not need to be fragmented and send information encoded in the number of fragments. Mazurczyk and Szczypiorski [63] describe several covert channels relying on this technique.

2.5.6 Destination Options Header

The Destination Options header carries optional information relevant only to the destination nodes. It may appear twice in the IPv6 headers stack: a)

after the hop-by-hop header when has options that need to be processed by the first destination in the IPv6 header and the ones listed in the routing header; and b) after all other extension headers when it carries options to be processed only by the final destination.

Because options of both options headers, hop-by-hop and destination, follow the same format, the covert channels identified are similar to those shown in Figure 2.12. Details of how to exploit those channels are described in subsection 2.5.3. In addition, the Swiss Unix User Group reports an implementation of the covert channel DH.18 [34].

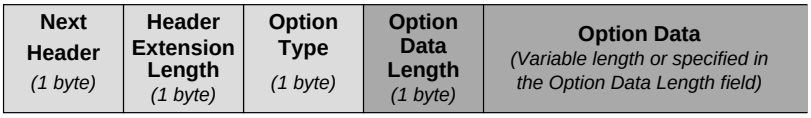

Figure 2.12: Covert Channels in the Destination Options header.

ID	Field	Covert Channel	Bandwidth
DH.17	*Option Data: Padding*	Set a false padding value	Up to 256 bytes/packet
DH.18	*Option Type: Unknown*	Fabricate one or more options	Up to 2038 bytes/packet

Field	*PadN*	**RFC**	2460
ID	DH.17	**SPL**	*final*
***cctool* ID**	DOHPAD	**checksums**	AH ICV

This is an equivalent of channel HBH.7.

Field	*Unknown Option*	**RFC**	2460
ID	DH.18	**SPL**	*final*
***cctool* ID**	DOHFAKE	**checksums**	AH ICV

This is an equivalent of channel HBH.8.

2.5.7 Authentication Header

The Authentication header (AH) is the one of the two headers that comprise IPsec. It provides connectionless integrity and data origin authentication of individual IP packets. It does so by calculating an integrity check value (ICV) per packet based on particular fields from other extension headers and from the IPv6 header as well. Whether a header field is actually used in the ICV computation or not depends on its mutability in transit. Only fields whose values do not change or change in a predictable way along the communication path are included in the computation. Other fields that may vary en-route, such as the *option data* field in options headers, are set to zero before being included calculation to avoid modifications in length or alignment. If a covert channel technique involves modifying a *immutable* or *mutable predictably* header field protected by authentication, Alice and Bob need to take special actions so their covert communication is not broken. This subsection discusses both potential covert channels in the Authentication header and possible solutions the agents can apply when using previously discussed

channels over authenticated headers. Figure 2.13 shows the structure of the authentication header and its potential covert channels.

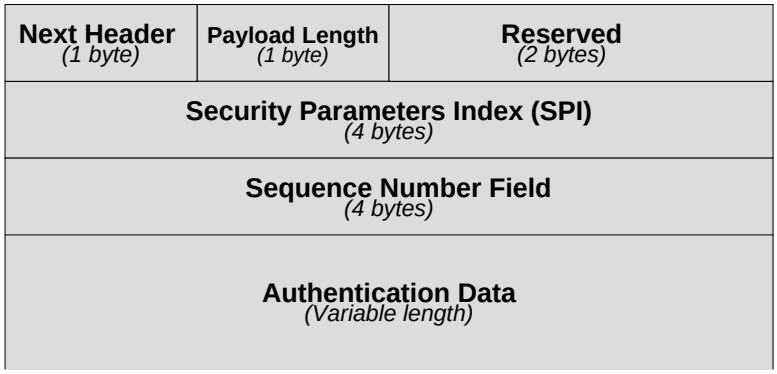

Figure 2.13: Format of the Authentication header.

ID	Field	Covert Channel	Bandwidth
AH.19	*Reserved*	Hide data in the unused bits	2 bytes/packet
AH.20	*All*	Insert an entire fake header	Up to 1022 bytes/packet

Field	*Reserved*	**RFC**	4302
ID	AH.19	**SPL**	*final*
***cctool* ID**	AHRESV	**checksums**	AH ICV

Alice can embed 2 bytes of data into *Reserved* field.

Field	*All*	**RFC**	4302
ID	AH.20	**SPL**	*forward*
***cctool* ID**	AHFAKE	**checksums**	AH ICV

When the authentication header is not present, Alice can fabricate one and insert it in the stack of extension headers. Alice has to set appropriate values for the *next header*, *payload length*, *security parameters index*, and *sequence number* to avoid detection. She places the covert data in the field that apparently contains *authentication data*. Obviously, the fake authentication header will not pass the IPsec integrity check at the receiving end. Therefore, Bob needs to strip it before the packet authentication check. Alice can send Bob up to 1022 bytes per packet through this channel (see Figure 2.14) Notice that this channel also involves modifying the size of the original packet, but this time the *payload length* in the IPv6 is not actually authenticated because there is no real AH.

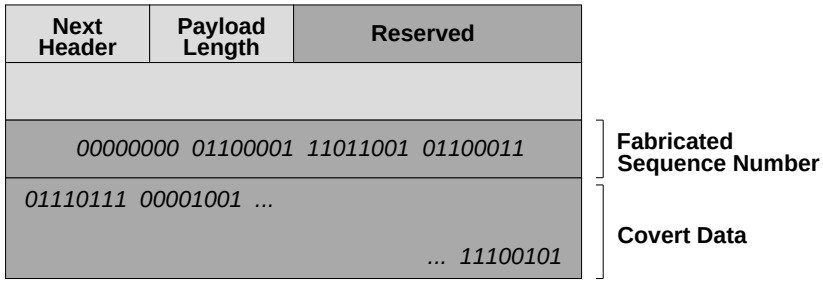

Figure 2.14: Covert Channel in the Authentication Header. Alice inserts fake authentication header in the stack of headers, simulating a *sequence number* to defeat active wardens.

2.5.8 Encapsulating Security Payload Header

Also part of IPsec, the Encapsulating Security Payload (ESP) Header provides confidentiality for all data transmitted end-to-end in IP packets. The

general structure of the ESP header and its plausible covert channels are illustrated in Figure 2.15.

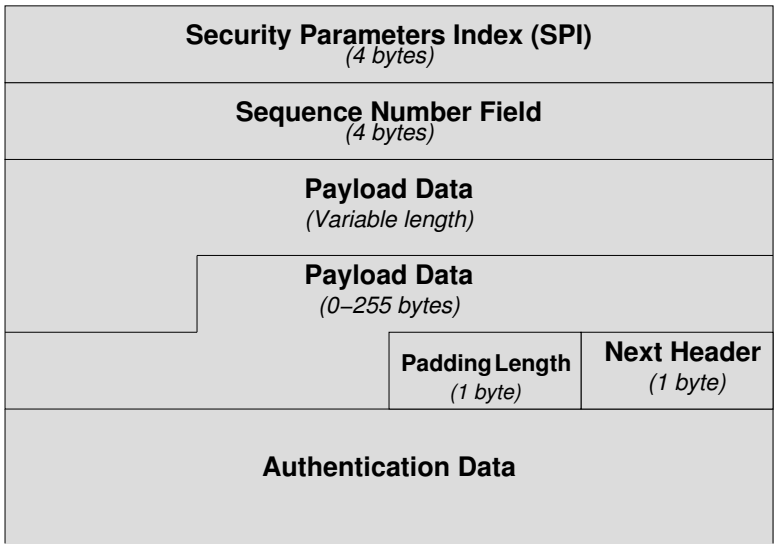

Figure 2.15: Format of the ESP Header.

ID	Field	Covert Channel	Bandwidth
ESP.21	*Padding*	Set a false padding value	Up to 255 bytes/packet
ESP.22	*All*	Insert an entire fake header	Up to 1022 bytes/packet

Field	*Padding*	**RFC**	4303
ID	ESP.21	**SPL**	*final*
***cctool* ID**	ESPPAD	**checksums**	AH ICV, ESP ICV

Although the *padding* field in the ESP header is optional, all IPv6 must support them. Alice can send up to 255 bytes per packet exploiting this

channel.

Field	*All*	**RFC**	4303
ID	ESP.22	**SPL**	*forward*
***cctool* ID**	ESPFAKE	**checksums**	AH ICV, ESP ICV

When the ESP header is not present, Alice can fabricate an entire ESP-like header to transmit covert information. Because the ESP header is an encapsulating header, she will need to include the original payload when creating her own. As in channel AH.20, a fake ESP header will not pass through the IPsec verification. Therefore, Bob needs to remove it, restoring the packet to its original form, before the packet reaches the final destination. Figure 2.16 shows an example of this channel.

Effects of the Encapsulated Security Payload Header:

As shown in Figure 2.15(a), the ESP header includes an authentication field. However, the ESP integrity check applies only to the ESP internal fields, the encapsulated headers, and the payload. That implies that, in transport mode, the presence of the ESP header does not affect the covert channels previously described, with exception of the ones belonging to the destination options header because that header is placed after the ESP header (i.e., it is encapsulated). To exploit the destination options header channels, Alice and Bob need access to the encryption keys. In tunnel mode, the "inner"

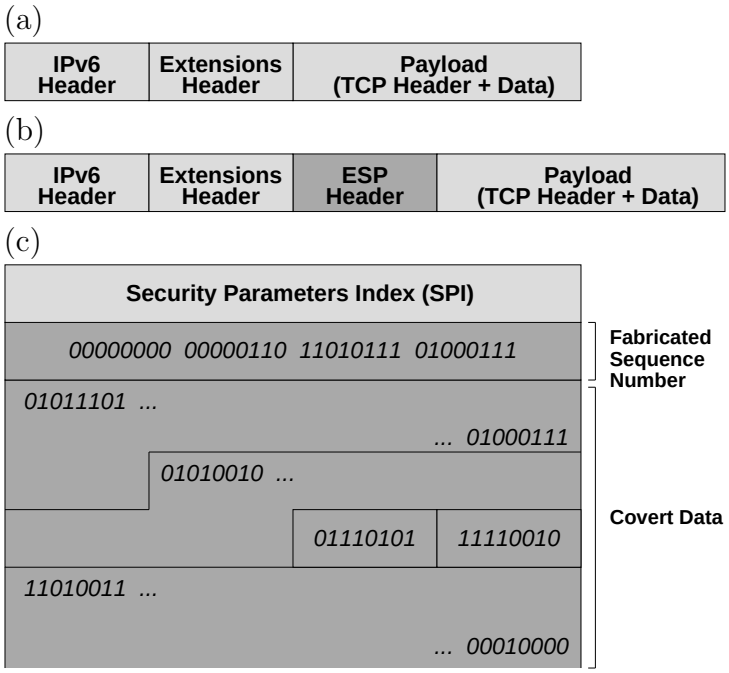

Figure 2.16: Covert Channel in the Encapsulating Security Payload (ESP). (a) Packet before inserting the fake ESP, (b) Packet after insertion, (c) Details of the fabricated header.

IP header and all its extensions are encapsulated from source to destination in the "outer" IP header. However, ESP tunnels can still be used for secret communication if Alice piggybacks an encrypted covert message to the "outer" header payload [57].

2.5.9 Mobility Header

Mobility header is defined by RFC 3775 [47] and is required for mobility support in IPv6. It is used to carry messages and special mobility options[5]

[5]These options use a different format than the ones used in Hop-byHop Options and Destination Options headers.

during mobile IPv6 operation. The structure of IPv6 Mobility Header is shown in Figure 2.17.

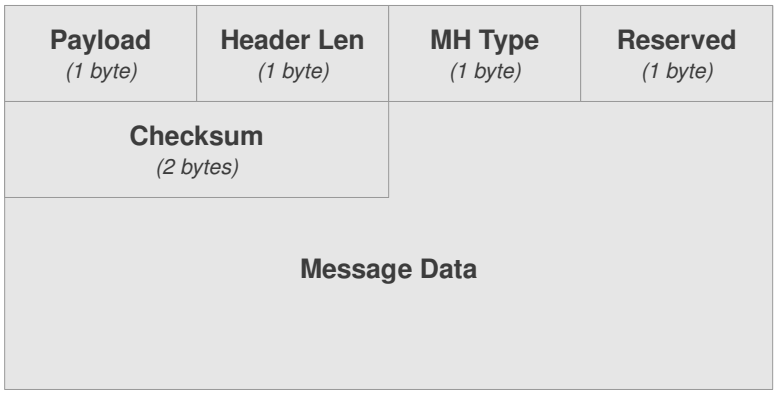

Figure 2.17: IPv6 Mobility Header Format.

Field	*Reserved (multiple)*	**RFC**	3775
ID	MH.23	**SPL**	*final*
***cctool* ID**	MHRESV	**checksums**	AH ICV, ESP ICV, MH

Mobility Header contains *Reserved* field, and additionally, different mobility messages and mobility options that it can carry, have their own reserved fields. They can be used by Alice in the same way as other similar fields in other headers.

2.5.10 ICMPv6 Header

Internet Control Message Protocol for Internet Protocol version 6 (ICMPv6) is an IPv6 equivalent of ICMP. It performs similar error reporting, diagnostic and discovery functions and uses similarly formatted messages, with a number of changes. Since all ICMPv6 messages are transported via IPv6 protocol, they contain IPv6 header and possibly also IPv6 extension headers. Consequently, all covert channels described in previous sections are still present in ICMPv6 messages.

Basic ICMPv6 specification defines only a few message types. However, the specification allows addition of new messages types and a number of them have been added, constituting new parts of ICMPv6 protocol.

The structure of an ICMPv6 message is defined by RFC 4443 [19]. All ICMPv6 messages contain the IPv6 header and zero or more IPv6 extension headers. The ICMPv6 header follows in the sequence of extension headers and it is identified by a *Next Header* value of 58 in the header immediately before it.

Figure 2.18: ICMPv6 Header Format.

An important observation about ICMPv6 error messages is that they are generated in response to an exceptional situation, by any node (e.g. a router) that encounters an error. As such, it is unlikely that the originating node has an established IPsec security association with the destination node. Consequently, ICMPv6 error messages are rarely protected by IPsec security.

Field	*Code*	**RFC**	4443
ID	—	**SPL**	—
***cctool* ID**	—	**checksums**	AH ICV, ESP ICV, ICMP

Code field can be manipulated for the purpose of covert communication. It offers potential bandwidth of 8 bits per packets. Since the legitimate contents of the field and the effects of the covert manipulation vary depending on the message type, more detailed discussion is presented below in sections covering channels in specific ICMPv6 messages.

Field	*Checksum*	**RFC**	4443
ID	ICMP.24	**SPL**	*dest*
***cctool* ID**	ICMPCHECK	**checksums**	AH ICV, ESP ICV, ICMP

Alice can abuse ICMPv6 *checksum* to carry covert data. The success and secrecy of this scheme directly depends on whether any of the intermediate nodes attempts to verify the checksum's correctness. If it is only the destination node that performs the verification, Alice's message can successfully reach Bob. Bob can then re-calculate the checksum and restore to its legal

value ensuring that the cover message is not discarded as damaged in transit.

2.5.11 Destination Unreachable Message

Destination Unreachable ICMPv6 message is generated by a node or by IPv6 network stack when it encounters a packet that cannot be delivered to its destination address for reasons other than congestion. The *Code* field is set depending to the reason for the delivery failure. The body of the message contains a 4-byte unused section and as much of the invoking packet data as possible without exceeding the minimum IPv6 MTU.

Field	*Unused*	**RFC**	4443
ID	ICMP.25	**SPL**	*forward*
***cctool* ID**	DUMUNUSED	**checksums**	AH ICV, ICMP

Destination Unreachable message contains a 4 byte long unused field. An attacker can insert data into the field achieving covert transmission bandwidth of 32 bits per packet. Since the field is supposed to be set to zero, the receiver can restore its original value.

2.5.12 Packet Too Big Message

A Packet Too Big message is sent by a router that discovers it cannot forward the IPv6 packet as requested because the outgoing link MTU is smaller than

the packet's size. The *Code* field of a Packet Too Big message is always 0, while its body carries MTU of the outgoing link and as much of the invoking packet as possible without exceeding the minimum IPv6 MTU.

Field	*Code*	**RFC**	4443
ID	ICMP.26	**SPL**	*final*
***cctool* ID**	PTBCODE	**checksums**	AH ICV, ESP ICV, ICMP

The *Code* field of Packet Too Big message is not used by the sender and it is supposed to be ignored by the receiver which is the behavior identical to any reserved field. Alice can inject her covert message into the field, resulting in a bandwidth of 8 bits per packet. Since the field is supposed to be set to zero, Bob can easily restore its original value. The restoration is not required however, as the receiver is supposed to ignore the value.

Field	*MTU*	**RFC**	4443
ID	ICMP.27	**SPL**	*final*
***cctool* ID**	PTBMTU	**checksums**	AH ICV, ESP ICV, ICMP

32-bit *MTU* field is supposed to carry the information about the link's maximum transmission unit. Alice can overwrite this information entirely to gain 32 bit per packet covert communication bandwidth and possibly causing confusion as to the correct MTU value. Alternatively, Alice can take advantage of the fact that the majority of the existing network MTUs are relatively small and do not require 32 bit integer space. She can use

high-order bits to transmit her messages and leave the low-order bits and the original MTU value intact. For example, an MTU of 1500 bytes requires only 11 bits to transmit and Alice can achieve a bandwidth of 21 bits per packet. This scheme avoids the MTU confusion by relying on Bob to restore the original MTU value.

If Alice cannot rely on Bob and does not want to risk transmitting illegal MTU values, she can still use this channel if she limits her messages to values between the actual MTU and the minimum MTU required by IPv6 (1280 bytes [26]).

2.5.13 Time Exceeded Message

A Time Exceeded Message is sent by a router when it encounters a packet with *hop limit* field equal 0 or by a node when a packet re-assembly time is exceeded. The body of Time Exceeded message comprises an unused field and as much of the invoking packet as possible.

Field	*Unused*	**RFC**	4443
ID	ICMP.28	**SPL**	*final*
***cctool* ID**	TEMUNUSED	**checksums**	AH ICV, ESP ICV, ICMP

Time Exceeded message contains an unused field of 4 bytes. Alice can use the field to send covert messages.

2.5.14 Parameter Problem Message

When an IPv6 node finds a problem within packet's IPv6 header or extension headers and the problem prevents it from completing the packet's processing, it might send a Parameter Problem message. The message contains an offset pointing to the detected error and as much of the defective packet as possible.

Field	*Pointer*	**RFC**	4443
ID	ICMP.29	**SPL**	*final*
***cctool* ID**	PPMPOINT	**checksums**	AH ICV, ESP ICV, ICMP

32 bit *pointer* field can be manipulated to carry covert data. Alice can use the entire field, for 32 bit per packet throughput, or she can take advantage of the fact that many packets have lengths that do not require using a 32 bit integer pointer, even more so since the pointer is likely to point into a header not the body of a message. Since the likely values of the pointer field do not require all 32 bits, Alice can use high-order bits and preserve the original pointer value for easier message restoration.

If Bob does not restore the message to its original form, the receiver will be misled as to the source of the problem that prompted the Parameter Problem message.

2.5.15 Echo Request Message

Echo Request message is a part of diagnostic ping mechanism that every IPv6 node is required to implement. Echo Request message *Code* field is always 0, and its body carries *Identifier* and *Sequence Number* fields, as well as arbitrary data.

Field	*Code*	**RFC**	4443
ID	ICMP.30	**SPL**	*forward*
***cctool* ID**	ERQCODE	**checksums**	AH ICV, ESP ICV, ICMP

The *code* field of Echo Request message is supposed to be set to zero. Alice can overwrite the field and transmit 8 bits per packet. Since the original value is known, Bob can restore packet integrity.

Even though the specification does not direct the receiver to ignore this value, in practice it is likely to be ignored making this field an equivalent to a reserved field. If the field value is ignored by the receiver, packet restoration is not required.

Field	*Data*	**RFC**	4443
ID	ICMP.31	**SPL**	*forward*
***cctool* ID**	ERQDATA	**checksums**	AH ICV, ESP ICV, ICMP

The body of Echo Request message carries arbitrary data and it one of the most widely recognized covert channels. Numerous ICMP covert channel

implementations exist. v00d00n3t demonstrated this kind of covert channel.

2.5.16 Echo Reply Message

Since Echo Reply message is a mirror of the invoking Echo Request message, all covert channels are exact equivalents.

Field	*Code*	**RFC**	4443
ID	ICMP.32	**SPL**	*forward*
***cctool* ID**	ERYCODE	**checksums**	AH ICV, ESP ICV, ICMP

The *code* field of Echo Reply message is supposed to be set to zero. Alice can overwrite the field and transmit 8 bits per packet. Since the original value is known, Bob can restore packet integrity.

Field	*Data*	**RFC**	4443
ID	ICMP.33	**SPL**	*final*
***cctool* ID**	ERYDATA	**checksums**	AH ICV, ESP ICV, ICMP

The body of Echo Reply message carries arbitrary data and it one of the most widely recognized covert channels. Numerous ICMP covert channel implementations exist. v00d00n3t demonstrated this kind of covert channel.

2.5.17 Router Renumbering for IPv6

Router Renumbering for IPv6 protocol defines a mechanism for reconfiguring multiple routers simultaneously, including environments where the number of routers is unknown. Due to sensitivity of Router Renumbering protocol messages, the specification requires use of IPsec for all Command messages and suggests IPsec for other messages. The requirement specifies that both data authentication and message integrity must be ensured, which implies either use of IPsec Authentication Header or Encapsulating Security Payload Header with non-null authentication algorithm.

Router Renumbering protocol is specified by RFC 2894 [21] and introduces a new ICMPv6 message type (Type 138).

All Router Renumbering protocol messages carry the header shown on Figure 2.19.

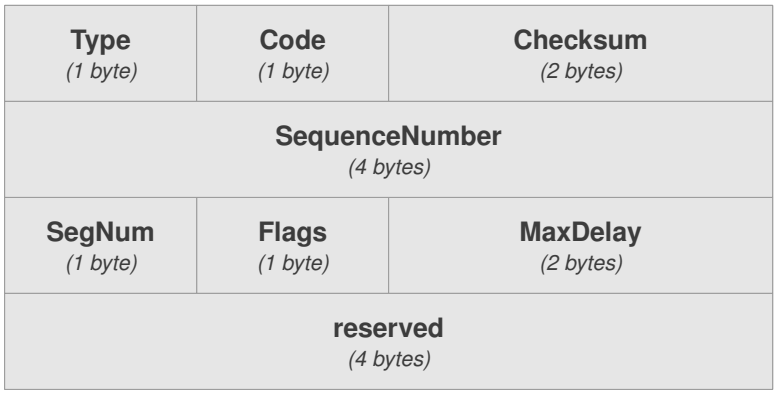

Figure 2.19: Router Renumbering Header Format.

Field	*SegmentNumber*	**RFC**	2894
ID	RR.34	**SPL**	*final*
***cctool* ID**	RRSEGNUM	**checksums**	AH ICV, ESP ICV, ICMP

SegmentNumber field is intended to help differentiate between several Router Renumbering messages bearing the same *SequenceNumber*. Since the *SegmentNumber* value does not imply any ordering, and since many Router Renumbering message are likely to have only one segment, Alice can easily alter the value without affecting the cover messages.

SegmentNumber field is 1-byte wide offering a bandwidth of 1 byte per packet.

Field	*Reserved (Flags)*	**RFC**	2894
ID	RR.35	**SPL**	*final*
***cctool* ID**	RRRESV	**checksums**	AH ICV, ESP ICV, ICMP

Router Renumbering header contains *Flags* field which, in turn, contains 3 unused flags. The flags can be used to insert 3 bits of data per packet.

Field	*MaxDelay*	**RFC**	2894
ID	RR.36	**SPL**	*final*
***cctool* ID**	RRMAXD	**checksums**	AH ICV, ESP ICV, ICMP

MaxDelay field transmits a maximum delay by which a recipient must delay the response to the message. The actual delay is expected to be a

random number between 0 and the *MaxDelay* value. Alice can overwrite this field to insert her data, taking care to ensure that the new value is lesser than the original one. As a consequence, the recipient node will probably delay its response by less than the sender expected, but still within the specified range. *MaxDelay* field is 16-bit wide, but the actual bandwidth per packet depends on the field's original value.

Field	*Reserved*	**RFC**	2894
ID	RR.37	**SPL**	*final*
***cctool* ID**	RRRESV2	**checksums**	AH ICV, ESP ICV, ICMP

32 bit *Reserved* field present in Router Renumbering header offers the same covert communication possibilities as other similar fields in other headers.

2.5.18 Router Renumbering Command Message

In addition to the header analyzed above, Router Renumbering Command Message includes a body that consists of zero or more Prefix Control Operation (PCO) sections. Each section might have a different length and, in turn, each consists of a Match-Prefix Part and zero or more of Use-Prefix Parts.

Even though there exist covert channel in Router Renumbering PCO sections, they are not listed here as they only appear in Command messages which are required to be protected by IPsec.

2.5.19 Router Renumbering Report Message

Router Renumering Report Message consists of zero or more Match Reports sections. Report Messages are likely easier targets for covert communication as the specification does not require that they are protected by IPsec and moreover, they do not have any effect on the network operations.

Field	*Reserved*	**RFC**	2894
ID	RR.38	**SPL**	*final*
***cctool* ID**	RRRESV3	**checksums**	AH ICV, ESP ICV, ICMP

14 bit *Reserved* field present in Router Renumbering MatchReport section offers the same covert communication possibilities as other similar fields in other headers. The available bandwidth varies depending on the number of MatchReport sections in given Router Renumbering message.

2.5.20 Mobility Support in IPv6

RFC 3775 [47] describes mobility support in IPv6 protocol. The support allows an IPv6 node to remain reachable at its home address despite changing its location within the network. While the mobile node resides in its new location, traffic addressed to the node's home address is transparently routed to the new address. Additionally the new address can be registered with the node's correspondents to allow direct communication with the mobile node bypassing the home address.

Mobility Support in IPv6 defines 4 new ICMPv6 messages: Home Agent Address Discovery Request Message, Home Agent Address Discovery Reply Message, Mobile Prefix Solicitation Message, and Mobile Prefix Advertisement Message.

2.5.21 Home Agent Address Discovery Request Message

The Home Agent Address Discovery Request message is sent by the mobile node in case it needs to discover the address (or addresses) of available Home Agents.

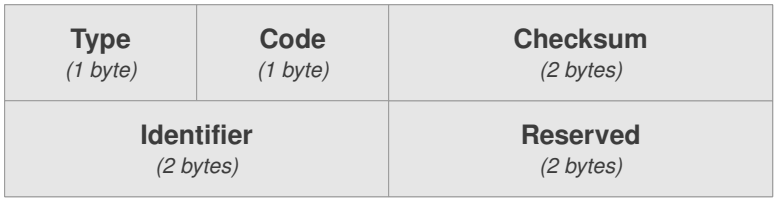

Figure 2.20: Home Agent Address Discovery Request Format.

Field	*Code*	**RFC**	3775
ID	HAA.39	**SPL**	*final*
***cctool* ID**	HAADRCODE	**checksums**	AH ICV, ESP ICV, MH, ICMP

Code field of the Home Agent Address Discovery Request message is set to zero, Alice can overwrite it to transmit 8 bits of covert information. Since

the original value is known, Bob can restore is easily. The restoration might not be required however since the field is constant it is likely to be ignored by the receiver.

Field	*Reserved*	**RFC**	3775
ID	HAA.40	**SPL**	*final*
***cctool* ID**	HAADRRESV	**checksums**	AH ICV, ESP ICV, MH, ICMP

The *Reserved* field of the Home Agent Address Discovery Request Message is initialized to zero and is supposed to be ignored by the receiver. Alice can use it to transmit 8 bits of covert information.

2.5.22 Home Agent Address Discovery Reply Message

The Home Agent Address Discovery Reply Message is sent in response to a received Home Agent Address Discovery Request. Its format is identical to the request message except that it carries the addresses of available Home Agents (see Figure 2.21). Due to the similar structure, *Code* and *Reserved* channels exist in this message as well as can be similarly used.

Field	*Addresses*	**RFC**	3775
ID	HAA.41	**SPL**	*final*
***cctool* ID**	HAADRADDR	**checksums**	AH ICV, ESP ICV, MH, ICMP

Figure 2.21: Home Agent Address Discovery Reply Format.

Alice can use the *Addresses* field to transmit covert information. If the field contains only one address, she should not overwrite it, as this would block the Home Agent discovery. In such case, she should append an extra "address" and fill it with her data as well as adjust payload length in IPv6 header.

If there is more than one address contained in the message, Alice can risk overwriting some of them, hoping that the remaining address will be sufficient to complete Home Agent discovery process.

2.5.23 Mobile IPv6 Fast Handovers

RFC 5568 [54] addresses the problem of high "handover latency" resulting from standard Mobile IPv6 operations. It obsoletes and amends older RFC 5268 redefining two of its ICMPv6 messages as Mobility Header messages. As a result, there are no new ICMPv6 covert channels in Fash Handovers protocol.

2.6 Covert Channels in Tunneled Traffic

Since both IPv4 and IPv6 are designed to encapsulate and transport other protocols, they can also be used to provide transport for one another, that is to tunnel IPv6 traffic inside IPv4 protocol or vice versa. The presence of network tunnels will obviously affect covert channel communications. The implications are primarily related to the location of the agents communicating covertly. In the presence of tunneled traffic, Alice and Bob need to locate themselves in particular spots along the communication path. The following scenarios are possible:

IPv4 traffic in an IPv6 tunnel: This case does not fundamentally differ from the usual IPv6 protocol traffic. Encapsulated IPv4 traffic is treated as any other IPv6 payload. The same covert channels are present as in the standalone IPv6 traffic.

IPv6 traffic in an IPv4 tunnel: Since not all networks support IPv6 na-

tively, this scenario is the most common case present today. Because this study focuses only on IPv6 covert channels, the concern here is with the channels present in the inner (encapsulated) header. The covert channels present in IPv6 are still present in the encapsulated traffic, as long as Alice and Bob can understand the IPv4 protocol. However, the tunnel can effectively block covert channel communication if it employs Internet Protocol Security (IPsec) suite. For detailed treatment of IPsec impact, see Section 2.7.

IPv6 traffic in an IPv6 tunnel: In this scenario both inner and outer headers follow the IPv6 specification, hence both can carry covert data using IPv6 covert channel techniques. In effect, both headers provide two independent covers for hiding information. However, the main reason for transporting IPv6 traffic in an IPv6 tunnel is to provide security via IPsec used by the tunnel. It is therefore likely that if Alice and Bob attempt to exploit covert channels present in this scenario, they will have to contend with IPsec security. See section 2.7.

In the above scenarios, if Alice and Bob want to exploit covert channels present in the tunnel (encapsulating) traffic, they have to make sure that they are positioned correctly with respect to tunnel endpoints and they can both see tunnel traffic. If Alice is positioned "before" tunnel's source, she will not be able to modify tunnel traffic as it does not exist yet. If Bob is "behind" tunnel's destination, the traffic modified by Alice will be stripped by

the tunnel endpoint and Alice's modifications will be lost. Cross-tunnel communication is however possible if conducted using tunneled (encapsulated) traffic.

2.7 Impact of Internet Protocol Security

Since one of the goals of IPsec is to protect the traffic from unauthorized modification and since Alice and Bob do modify the traffic to achieve their covert communication, the presence of IPsec protection must have an effect on the communication. If changes introduced by Alice are detected by IPsec mechanisms, the altered traffic will be discarded blocking the covert channel. Additionally, a security alert might be raised, prompting added scrutiny and further threatening covertness of Alice's and Bob's conversation. The challenge facing Alice and Bob is, on one hand, to gain access to IPsec-protected packets that might be encrypted, and on the other hand to prevent IPsec from noticing the traffic alterations, either by limiting the changes to parts of IPv6 header not protected by IPsec, or by undoing them before IPsec processing can take place.

The ICV computation consists of applying message authentication code (MAC) algorithms over immutable and mutable but predictable fields from the IPv6 header and its extension headers. Several of the proposed covert channels involve changing values ofsome of those protected fields. The existence of this channel can cause failure of the integrity check, which triggers

an auditable event in IPsec. That may cause both the immediate detection of the channel and the disruption of the overt communication. Alice and Bob must take actions to avoid such situations.

To avoid a failed check on the ICV, Alice must either be the sender, and therefore compute the ICV including the covert data, or Bob must intercept the packet before it reaches its destination, and remove the covert data, as described in [57].

However, despite the challenge, the IPv6 extension headers that IPsec employs can offer additional opportunities for covert channel exploitation.

The simplest scenario to consider is when Alice and Bob have access to IPsec security context information. Possessing that information allows them to freely modify traffic and rewrite IPsec ICV values as needed. In this scenario, the presence of IPsec does not matter, except that it provides additional covert channels existing in IPsec headers.

Assuming that Alice and Bob do not have access to IPsec keys, the impact depends both on IPsec mode of operation and the relative location of Alice and Bob with respect to IPsec endpoints.

2.7.1 IPsec Transport Mode

When IPsec operates in end-to-end transport mode, it can protect the integrity of some parts of IPv6 header as well as the entirety of its payload, it can also ensure confidentiality of packet's payload but not its headers. Therefore, Alice and Bob do not have to be concerned about traffic encryp-

tion as it does not affect IPv6 headers, but they do have to contend with IPsec protection of IPv6 headers' integrity. The following scenarios exist:

1. Alice and Bob are positioned outside of IPsec endpoints. In this case, since IPsec is operating in transport mode, it means that either Alice and Bob are the sender and the receiver (i.e. they generate cover traffic), or they are concealed on the sender's and receiver's hosts with access to network communications stack. As a result, Alice and Bob can see the traffic before (or after) it is protected by IPsec security. In this scenario, IPv6 covert channels can be exploited as if IPsec was not present. Moreover, Alice and Bob cannot use any channels present in Authentication Header or Encapsulating Security Payload as they are not available yet for Alice and they are already gone when Bob receives the traffic.

2. When Alice and Bob are between IPsec endpoints, they both see traffic protected by IPsec mechanisms. As a result, they can attempt to use all IPv6 covert channels, including channels present in AH and ESP, but they have to be aware whether modifications made by Alice violate IPsec ICV checksums. Alice and Bob can either limit themselves to non ICV-violating channels, or they have to ensure that Bob can undo Alice's changes before they are discovered by the IPsec destination endpoint.

3. If Alice is positioned before IPsec processing takes place while Bob is

between the endpoints, Alice can freely modify traffic, but is not able to access IPsec headers as they are not created yet. Since Bob can see IPsec-protected traffic, he is not able to perform traffic modifications and is therefore unable to undo Alice's modifications if they fall within IPsec-protected parts of IPv6 headers. If Alice decides to modify the parts that will be protected by IPsec, she has to consider that the modified traffic m' will continue to the final destination and will have to pass the final destination-level integrity check.

4. When Alice is positioned so that she can see IPsec-protected traffic, while Bob is behind the IPsec endpoint, their respective abilities are reversed. Alice's modifications have to contend with IPsec integrity protection, while Bob can modify traffic without obstacles. Alice is forced to rely on covert channels that do not violate IPsec ICV mechanism lest they be detected and discarded by the IPsec endpoint present between her and Bob.

2.7.2 IPsec Tunnel Mode

The above scenarios are altered when IPsec operates in tunnel mode. In this mode, IPsec endpoints include the entirety of original IPv6 packet into the new packet's payload. Consequently, both IPsec's integrity and confidentiality protections extend to the entire original packet. The modified scenarios are listed below.

1. Alice and Bob are located outside of IPsec endpoints. This scenario is equivalent to the scenario presented in the previous section. Since Alice sees the traffic before it is protected by the IPsec tunnel and Bob sees it after IPsec protection, they can operate as if IPsec was not present. Consequently, Alice and Bob can use all covert channels except for ones present in IPsec headers.

2. Alice and Bob are between IPsec endpoints. Thanks to their location both of them can see two IPv6 layers, inner (encapsulated) traffic and outer (tunnel) traffic. In this scenario covert channel opportunities depend on tunnel's use of encryption.

    - If the IPsec tunnel uses only integrity protection, both inner and outer traffic is visible to both Alice and Bob, thus they can use both IPv6 layers for covert channels. The restriction is that any modifications to the inner (encapsulated) traffic have to be undone by Bob or they will be detected by the tunnel endpoint. Outer (tunnel) traffic behaves exactly like any other IPv6 traffic with the addition of IPsec-specific headers that can be exploited.
    - If the tunnel uses encryption, it means that the inner traffic is hidden and cannot be accessed neither by Alice nor by Bob. Outer (tunnel) traffic can be used as above.

3. Alice is outside of the tunnel, Bob is between IPsec endpoints. In this scenario, tunnel traffic cannot be used as Alice cannot see it. Inner

(encapsulated) traffic is therefore the only option that Alice and Bob have. However, if the tunnel uses encryption, any modifications that Alice might do will not be visible to Bob as he is not able to see inner, encrypted traffic. Therefore use of IPsec encryption renders the communication between Alice and Bob impossible. If the tunnel uses only IPsec integrity protection, Bob will not be able to undo Alice's modifications. As a result, Alice has to preserve integrity to final destination level.

4. Alice is between IPsec endpoints, Bob is outside of the tunnel. Similarly to the previous case tunnel traffic cannot be used since one of the communicating parties (Bob) cannot see it. Inner traffic cannot be used either because any modifications performed by Alice will be detected by IPsec endpoint and discarded. No communication is therefore possible.

2.7.3 Using IPsec ICVs for Brute-force Packet Restoration

Another aspect of IPsec impact on covert channel communication is the possibility of using IPsec ICVs for packet restoration. Similarly to the protocols named in section 2.4, IPsec has a checksum used for detecting traffic alterations. Unlike the protocols mentioned before, IPsec checksum covers the actual IPv6 header and not pseudo-header, therefore it protects more IPv6 header fields and it is potentially better for the purpose of packet restoration.

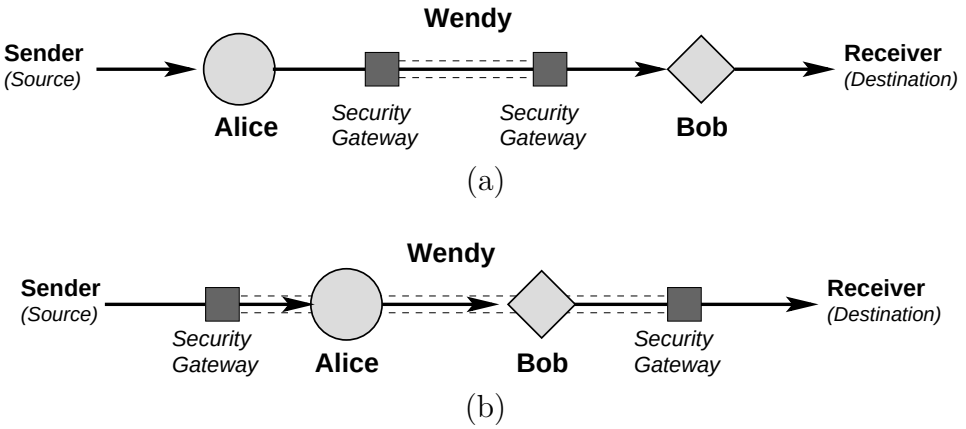

Figure 2.22: Location of Alice, Bob, and Wendy under IPsec Tunneling Mode. (a) Alice and Bob embed and extract, respectively, covert data outside the tunnel; Wendy is within the tunnel, so an attempt to modify the traffic might cause the authentication to fail. (b) Alice, Bob, and Wendy are all inside the tunnel; the warden can prevent covert channels in the outer header.

Whether this approach is feasible depends primarily on the algorithm used in packet's ICV calculation. Specifically it depends on whether ICV can be computed without access to IPsec security context. For example, if the ICV algorithm involves a signed message digest, Bob cannot forge the signature without having access to IPsec security context, but he is able to compute the digest value. Consequently, Bob might attempt to compute multiple digest values for all possible field values to ascertain which was the original value and then restore the value. Since any digest algorithm used by IPsec has to be cryptographically strong, the use of IPsec checksum for packet restoration avoids the checksum collision problem existing in non-secure checksums used by ICMPv6, TCP and UDP protocols.

The approach is not possible with ICVs calculated using key-dependent

one way hash functions since the calculation of the value is dependent on access to the appropriate key. ICV calculation algorithms currently defined for use with IPsec [30, 59, 60, 62] all use key-dependent functions.

Chapter 3

Network-aware Active Wardens

Chapter 2 describes a study of covert channels existing in IPv6 and related protocols and presents a comprehensive list of discovered attacks. These channels pose a definite information security threat to any network system employing IPv6 protocol. Chapter 3 proposes and analyzes a number of covert channel defenses, updating the traditional specification-based techniques to work with IPv6, then introducing the idea of network-aware wardens. The chapter's organization is as follows: Section 3.1 summarizes the existing technologies that provide defense against covert channel attacks. Sections 3.2 and 3.3 discuss the relevant attack and defense models respectively. Section 3.4 presents a detailed description of covert channel countermeasures, while Section 3.5 enumerates models of wardens' network awareness. Finally, Sections 3.6 and 3.7 analyze the impact of tunneled traffic and IPsec protection on active warden effectiveness.

3.1 Related Work

The most effective defensive mechanisms against network storage channels for IPv4 are protocol scrubbers, traffic normalizers, and active wardens. Protocol scrubbers [61] and traffic normalizers [37] focus on eliminating ambiguities found in the traffic stream, carefully crafted with the purpose of evading network intrusion detection systems. Ambiguous network packets are those which could have different interpretations at endpoints depending on the implementation of the protocol stack. Covert channels are certainly a form of ambiguous traffic. Handley and Paxson [37] describe IP, UDP, TCP, and ICMP normalizations based on protocol specification, highlighting the importance of preserving the end-to-end protocol semantics.

Active wardens were introduced by Simmons [79] as part of the Prisoners' Problem and have been subsequently discussed on several other occasions [5, 6, 20, 29]. Active wardens, as presented by Fisk et al. [29], are network services resembling a firewall that modify all traffic under the assumption that it is carrying steganographic content. They defeat steganography in structured carriers with objectively defined semantics, such as IPv4 protocol, by making semantics-preserving alterations to packet headers. In effect, Fisk's active warden performs similarly to a covert channel participant albeit sending noise, not real data. These techniques, although effective for many IPv4 covert channels, rely mainly on protocol specification as a reference for protocol syntax and semantics and do not record state or gather

network topology information. Moreover, they are concerned with IPv4 and upper-layer protocol and not with IPv6 and its associated protocols.

Another way to attack covert channels is a statistical analysis of network traffic properties. Well-known techniques of stego analysis can reveal an existence of a covert channel based on the changes in statistical properties of traffic. In such a case, it is possible to build a network intrusion detection system (NIDS) that would monitor the traffic, gather required statistical data and raise an alarm if a covert channel's existence is suspected. These techniques focus on intrusion detection and differ fundamentally from traffic normalizers and active wardens whose main purpose is defeating potential covert channels without necessarily being aware of their existence.

Traffic normalizers and packet scrubbers are a simple and efficient way of defeating covert channels by eliminating ambiguities in network traffic but they have limitations as well. The most significant one is the fact that they rely on the protocol specification to eliminate traffic ambiguities. While this covers simple covert channels (e.g. the ones utilizing restricted protocol fields), it fails in cases where the message content depends on the surrounding network. For example, a value of a packet's hop limit field depends on the route it traveled through the network. It might naturally vary from one packet to another and the protocol specification alone does not provide enough information to normalize the value. In such cases, scrubbing the packet might indeed endanger the network performance, affect the ongoing communication or disable some network functionality. Even harder case is

presented by protocol fields that are supposed to carry addresses. There exist several covert channels that can abuse such fields to send covert data. A traffic normalizer does not have enough information to modify such fields.

An active warden might attempt to modify the fields to replace the potential covert message with noise, but it also does not possess enough information to always distinguish between legitimate content and covert messages. In consequence, it might destroy legitimate data and disrupt network operation.

3.2 Attack Model

Similarly to the model presented in Chapter 2, the covert channel attacks considered below involve Alice and Bob who wish to keep their communication secret. To maintain their privacy while conducting the conversation, Alice and Bob forego generating their own cover traffic and use 3rd party overt communications for their own purpose. They have both resources and skills to modify passing traffic by either altering values of protocol fields or by inserting a protocol header, or even injecting an entire new packet into the traffic.

Assumption 1 *Alice and Bob can modify network traffic traveling between nodes.*

Since secrecy is their primary concern and their motivation for using covert channel techniques, Alice and Bob attempt to avoid attracting atten-

tion by preserving the overt traffic's syntax and semantics as much as possible. In consequence, they will only use channels with syntax and semantics preservation level of *final destination* or *IPsec* (see Section 2.3), unless they can ensure that Bob will be able to undo Alice's modifications and restore the cover traffic integrity.

In terms of scenarios presented in Section 2.2, Alice and Bob will only attempt scenario 4 if they are certain that Bob can perform traffic restoration. Otherwise, they will operate as detailed in scenario 3.

Assumption 2 *Maintaining integrity of cover traffic is a primary concern for Alice and Bob.*

It is possible that Alice and Bob know about the presence of an active warden. If they are aware of the warden's presence, they also know her location and capabilities and will adjust their methods to try to protect secrecy and integrity of their communication.

Assumption 3 *Alice and Bob might be aware of the presence of an active warden.*

Attackers that do not know about an active warden monitoring their conversation are said to be *blind*. Blind attackers might be concerned with avoiding other security systems, e.g. network intrusion detection systems, but not with overcoming active warden activity.

3.3 Defense Model

The primary goal of an active warden is to protect network information security by disrupting any covert channel communication contained in the observed traffic. The warden is not concerned with an actual attack detection, collection of attack information or stego-analysis of suspected communication.

Assumption 4 *Wendy attempts to break covert channel communication by modifying the observed overt communications.*

While defeating covert channels is Wendy's priority, she will try as much as possible to preserve the semantics of the overt traffic she modifies. In certain cases, however, Wendy might have to make a choice between a stricter adherence to traffic semantics preservation versus a greater assurance of covert channel elimination. For example, if the warden observes packets that violate protocol semantics, the cause of the violation could be either that the packets are modified by Alice to carry covert messages, or the packets could have been simply damaged in transit. If the packets are altered by Alice, the overt communication is likely not damaged, as the packets will later be "fixed" by Bob and the protocol semantics restored to their original state. Since Wendy does not know which of the two possible scenarios she is observing, she will consider the packets damaged and discard them. While this course of action will definitely impact the ongoing overt communication, it has a two-fold effect on the potential covert channel communication. First,

dropping the packet will also drop the covert data present, and second, it will force Alice and Bob to use semantics-preserving covert channels as they want to avoid impacting the cover traffic.

Assumption 5 *Wendy attempts to preserve the semantics of the modified overt communications, but might drop packets she perceives as damaged.*

In order to perform its function, the warden has to be located in such a way as to make it impossible for Alice and Bob to communicate via traffic bypassing the location of the warden. If Alice and Bob were able to achieve the bypass, the presence of the warden would be irrelevant. Since covert channel attacks are generally of concern in high-security environments where network is strictly monitored as a matter of course, a proper positioning of the warden is usually not difficult.

Assumption 6 *Wendy is always located so that she can observe all relevant network traffic.*

3.4 Countermeasures

Ideally, countermeasures deployed against covert channel communications will result in a complete disruption of covert communication, but it is also possible that only a partial success is achieved. Consequently, when considering the solutions presented below, their effect on specific covert channels is classified as follows:

channel defeated As a result of active warden's actions, covert communication using this channel is impossible.

channel partially defeated Some covert channel communications might be disrupted, but not all. Typically a *blind* attacker will be defeated, while more aware adversaries will be able to adjust their strategy and still succeed in their communication attempts, even though the adjustment will usually result in lowering the available covert channel bandwidth.

Independently from whether the channel in question is defeated or not, the applied countermeasures might have an effect on the legitimate overt traffic as mentioned in Section 3.3.

3.4.1 Specification-based Countermeasures

The protocol specification is the most obvious choice for a reference used in a covert channel defense. Packet scrubbers and traffic normalizers as well as active wardens use methods that are based on the specification.

Protocol Syntax Verification

Non-syntax-preserving covert channels are only briefly mentioned in Chapter 2. If such attacks are attempted, the attackers rely on the fact that intermediate IPv6 nodes are not required to perform full packet parsing while handling traffic, e.g. a router does not need to parse the fragment header

structure. These attacks can be defeated by enforcing a higher level syntax integrity check en route where the attacker does not expect it. Since protocol syntax is well-defined, the check is trivial and consists simply of complete packet parsing.

Traffic Normalization

Traffic normalization was initially proposed as an aid in network intrusion detection [37]. The normalizers were designed to combat intruders who might exploit traffic ambiguities to evade existing intrusion detection systems. By normalizing passing traffic, the ambiguities are removed and potential attacks are exposed. As originally described, normalizers are not concerned with mitigation of covert channel-based attacks.

Specific normalizations clearly depend on the protocol in question, but they generally follow the rules listed below:

1. If the correct value of a given protocol field is known, enforce the correct value, e.g. *Reserved* fields are usually supposed to be zeroed, a normalizer can ensure that this is indeed the case.

2. If the current value of a given field is incorrect, and the correct value cannot be established, discard the packet, or if possible, strip the offending part from the packet, e.g. verify packet's checksum and drop packets if the verification fails.

As mentioned, the original motivation behind traffic normalization was

to help intrusion detection via disambiguation of traffic. Interestingly, since covert channels can be considered a form of ambiguous traffic, a number of channels are defeated by traffic normalizers. In this study, the concept of traffic normalization is extended to cover IPv6 protocol, and re-defined to include only the rules that ensure full preservation of protocol semantics.

1. Reset all *Reserved* and *Unused* fields to their specified value (usually 0).

2. Reset all fields that legally can hold only one value to the legitimate value, e.g. *Code* field of Packet Too Big Message.

3. Verify that all fields containing protocol constants are set to known values, e.g. *Next Header* field should contain only values assigned by IANA [42]. If possible, remove unknown headers and options. If removal is not possible, discard the packet.

4. Verify all addresses contained in a packet. The fields subject to this verification include: IPv6 header *Source Address* field, Routing Header Type 0 *Addresses* field, as well as Home Agent Address Discovery Reply Message *Addresses* field. The addresses contained should not be loopback addresses, unspecified addresses, link-local addresses. Depending on the specific field, multicast addresses might be disallowed as well. Discard packets containing illegal values.

5. Compute and verify all checksums present in the packet. Discard pack-

ets that fail the verification[1].

6. Perform packet fragment re-assembly[2].

7. Verify that packet contents match its Router Alert option. Remove non-matching options.

8. Verify that *MTU* field contains a value larger than or equal to IPv6 minimal MTU[3]. Reset lower values to the minimal MTU.

9. Verify that *Pointer* field points to an appropriate location within the included packet. Discard packets with erroneous *Pointer* values.

The effectiveness of an active warden implementing the above rules is presented in Table 3.1 below.

field	**ID**	**result**	**description**
Next Header	4	defeated	*Next Header* values are protocol constants, unknown values are removed according to rule 3.
PadN	HBH.7	defeated	padding bits reset to zero according to rule 2.

[1]If Wendy is deployed to guard only against covert channels using *Checksum* field, it is possible to simply fix the checksum value. This procedure would invalidate checksum original functionality intended to guard against accidental packet corruption.

[2]IPv6 specification [26] states that IPv6 packets are not to be fragmented en route. It does not prohibit en route packet re-assembly.

[3]1280 octets

field	**ID**	**result**	**description**
Option	HBH.8	defeated	option types are protocol constants, unknown options are removed according to rule 3.
Value	HBH.10	defeated	*Value* field have to match packet contents or the option will be stripped according to rule 7.
Reserved	RH.11	defeated	*Reserved* field reset to zero according to rule 1.
Reserved	FH.13	defeated	*Reserved* field reset to zero according to rule 1.
Reserved	FH.14	defeated	*Reserved* field reset to zero according to rule 1.
Next Header	FH.15	defeated	*Next Header* fields carrying covert data are discarded during packet re-assembly performed according to rule 6.
All	FH.16	defeated	Fake packet fragment carrying covert data is discarded during packet re-assembly performed according to rule 6.

field	**ID**	**result**	**description**
Reserved	DH.17	defeated	*Reserved* field reset to zero according to rule 1.
Option	DH.18	defeated	option types are protocol constants, unknown options are removed according to rule 3.
Reserved	AH.19	defeated	*Reserved* field reset to zero according to rule 1.
Pad	ESP.21	defeated	padding bits reset to zero according to rule 2.
Reserved	MH.23	defeated	*Reserved* field reset to zero according to rule 1.
Checksum	ICMP.24	defeated	Packet is discarded according to rule 5.
Unused	ICMP.25	defeated	*Reserved* field reset to zero according to rule 1.
Code	ICMP.26	defeated	*Code* field reset to zero according to rule 2.
Unused	ICMP.28	defeated	*Unused* field reset to zero according to rule 1.
Code	ICMP.30	defeated	*Code* field reset to zero according to rule 2.

field	**ID**	**result**	**description**
Code	ICMP.32	defeated	*Code* field reset to zero according to rule 2.
Reserved	RR.35	defeated	*Reserved* field reset to zero according to rule 1.
Reserved	RR.37	defeated	*Reserved* field reset to zero according to rule 1.
Reserved	RR.38	defeated	*Reserved* field reset to zero according to rule 1.
Code	HAA.39	defeated	*Code* field reset to zero according to rule 2.
Reserved	HAA.40	defeated	*Reserved* field reset to zero according to rule 1.
Source Address	6	partially defeated	only data falling within illegal address ranges will be discarded following rule 4.; see Appendix A
MTU	ICMP.27	partially defeated	channels attempting to insert arbitrary data into the *MTU* field will be minimally affected; a clever attacker can devise a scheme that allows communication without triggering the normalization; see Appendix A

field	ID	result	description
Pointer	ICMP.29	partially defeated	packets containing arbitrary *Pointer* values will be discarded defeating the channel; a clever attacker can arrange a scheme where the communication is still possible using valid *Pointer* values

Table 3.1: Channels defeated by traffic normalization

Aggressive Traffic Normalization

Some of the normalizations proposed by Handley et al. [37] can be aggressive enough to erode the end-to-end protocol semantics. Their effect on the legitimate network traffic usually involves disabling of certain network functionality and forcing the default handling of traffic instead of allowing the optional functionality.

The rules listed below assume a similar approach, adapting it for use against IPv6 covert channels. As a result, additional covert channels are defeated at a cost of possible degradation of some network functionalities.

10. Reset *Traffic Class* field to 0.

11. Reset *Flow Label* field to 0.

12. Reset *Hop Limit* to a pre-configured value.

13. Remove Routing Header Type 0.

14. Set *MTU* field to a pre-configured value.

The results are listed in Table 3.2 below.

field	**ID**	**result**	**description**
Traffic Class	1	defeated	field set to zero according to rule 10.; any functionality (Differentiated Services [67], Explicit Congestion Notification [71]) that relies on the field is disabled; the packet is still delivered, as the routers will assume default handling
Flow Label	2	defeated	field set to zero according to rule 11.; flow labeling functionality of IPv6 [70] is disabled; the packet is still delivered with default handling assumed

field	ID	result	description
Hop Limit	5	defeated	field set to a pre-configured value according to rule 12.; if a routing loop with a hop limit length larger than the value set is passing through the warden, then the modified packets will loop forever, exhausting network resources
Addresses	RH.12	defeated	extension header removed according to rule 13.; loss of source routing capability
MTU	ICMP.27	defeated	field set to pre-configured value according to rule 14.; actual network MTU might not be transmitted correctly resulting in packet losses

Table 3.2: Channels defeated by aggressive traffic normalization

MRF-based Covert Channel Defense

Minimal Requisite Fidelity (MRF) is a concept introduced by Fisk et al. [29] as a measure of data alteration that results in the destruction of covert

communication while being still acceptable to the end points. Since network protocols have well-defined syntax and semantics, MRF can be precisely established for different protocol fields and an appropriate measure of alteration can be introduced preserving the functionality of traffic, while at the same time disrupting embedded covert messages.

An active warden performing MRF-based defense is essentially acting in the same way as a covert channel attacker (Alice), but instead of inserting covert messages, the warden injects random noise. The main difference is that the warden acts alone and therefore cannot perform traffic modification that would lower the semantics preservation level of the overt traffic, as she cannot rely on Bob (or equivalent) to restore the traffic. Effectively, in addition to the two generic rules for traffic normalization mentioned previously, an MRF-based active warden includes a third rule.

1. Normalization – enforce correct values if they are known
2. Normalization – destroy incorrect values if correct values cannot be established
3. MRF – scramble values whose correctness and incorrectness cannot be verified

As a result, an MRF-based active warden extended to work with IPv6 protocol semantics follows the rules listed below.

15. Map *Flow Label* to a different value. The mapping has to be constant across a single flow.

16. Set *Hop Limit* field value to a random value lower than the current one.

17. Randomize the order of traversed addresses listed in Routing Header *Addresses* field.

18. Set fieldMTU field to a random value lower than or equal to the current value and greater or equal to the minimum MTU required by IPv6 specification[4].

19. Map *Data* field to a different value. The mapping has to be reversed for reply packets.

20. Set *SegmentNumber* field to a random value. The value has to be unique among packets with the same *SequenceNumber*.

21. Set *MaxDelay* field to a random value lower than the original value.

22. Randomize the order of addresses listed in Home Agent Address Discovery Reply message *Addresses* field.

Table 3.3 lists the effects of MRF-based defense.

[4] 1280 octets

field	**ID**	**result**	**description**
Flow Label	2	partially defeated	changing the field value according to rule 15. breaks any covert channel that relies on inserting covert messages directly into the field; a clever attacker will bypass the defense by devising a scheme where the message is transmitted via the flow label changes within a flow and not through any specific value; the available covert channel bandwidth falls from 20 bits per packet to 1 bit per packet
Hop Limit	5	defeated	changing the field value according to rule 16. renders it unusable for covert channel attacks, but packets might be inadvertently dropped if the new value does not suffice for the packet to reach its destination node

field	ID	result	description
MTU	ICMP.27	defeated	the field is changed according to rule 17. and it is no longer usable for covert communication, but the MTU information cannot be transmitted accurately; network cannot perform at its optimal packet size
Data	ICMP.31	defeated	the field is mapped according to rule 18.
Data	ICMP.33	defeated	the field is mapped according to rule 18.
SegmentNumber	RR.34	defeated	the field is set according to rule 19.
MaxDelay	RR.36	defeated	the field is set according to rule 20.; as a side-effect, the reply message might be sent by the router faster than the sender expected

Table 3.3: Channels defeated by MRF-based defense

3.4.2 Network-aware Active Warden

Specification-based active wardens that perform according to the rules described above are able to defeat many of the covert channels listed in Chapter

2. Their knowledge of the protocol specification allows them to simulate endpoint semantics and then use the simulation to normalize the traffic. The goal of the normalization is to either establish correct packet field values and enforce the correctness, or if that is impossible, to find incorrect values and remove them. Failing that, an active warden will attempt to scramble the field values to destroy possible covert communication while preserving semantics of the packet.

Some of the more sophisticated covert channels are, however, able to resist specification-based active wardens. The common feature of these channels that allows them to avoid elimination by the defenses described above is that they modify protocol fields carrying information about the network itself. This information lies outside of the knowledge of the standard warden and as a result, the warden cannot establish whether it is correct or incorrect, making normalization impossible. Furthermore, since the information describes the network topology or state, it cannot be scrambled by the warden without being irrevocably damaged.

This study proposes to extend the active warden concept by adding a new capability of using the information about the surrounding network as a reference for traffic normalization and modification. Since the IPv6 protocol functions within the network, its packets should conform to the given network state. Since network storage covert channels rely on modifications of the network traffic, modified packets' conformance to the network should be disturbed by an active covert communication. An active warden pos-

sessing information about the network should be able to either restore the conformance or destroy offending packets and thus defeat the covert channel.

More colloquially, the observed traffic should not only comply with the protocol specification, but also "make sense" within the given network. This preserves the original idea of an active warden as a firewall and a traffic normalizer but at the same time expands its functionality to allow better defense against covert channels.

The wardens equipped with the new ability will still perform according to similar rules as before. They will first try to establish correct values of protocol fields and enforce the established values. If that is not possible, they will check whether observed values are clearly incorrect. Packets, headers or options carrying incorrect data will be dropped. Similarly to the simpler wardens, if the normalization as described above is not possible, the wardens will try to scramble the suspect fields to destroy the potential communication while preserving the packet semantics.

Network-based Normalization

Similarly to the specification-based traffic normalization described in Section 3.4.1, an active warden performing network-based normalization attempts to establish correct values for protocol fields and enforce the correctness, and if that is impossible, she tries to detect clearly incorrect values and remove them.

23. Verify that *Traffic Class* field points to a traffic class meaningful within

the network. It only makes sense for the traffic to belong to a traffic class if the class is recognized by the network. A network-aware warden should be aware of used traffic classes and reset unrecognized traffic class values. If the warden's network does not use the functionality provided by *Traffic Class* field, all values should be reset.

24. Verify that a flow label adheres to a flow. A flow label is intended to allow networks to distinguish separate network flows without parsing IPv6 packet payload. An active warden can perform a more thorough parsing and verify that given flow label value adheres to an actual network flow. Erroneous flow label values can be either reset to values previously used within the flow or to zero, although the latter option disables the flow label functionality.

25. Drop packets with *Hop Limit* value too low. An active warden possessing knowledge about network topology can determine whether packet's *Hop Limit* value is sufficient to reach its destination. A packet with insufficient hop limit value can be safely dropped as it cannot reach its destination anyway.

26. Reset *Hop Limit* to a safe value. An active warden can reset packet's hop limit to a random value lower than the original value, to preserve the functionality of *Hop Limit*, and higher than the value required to reach packet's destination, based on warden's topology knowledge.

27. Verify *Source Address* value. An active warden can establish whether

the source address contained within the packet corresponds to an actual network node. This mechanism is a more accurate version of ingress/egress filtering performed by firewalls.

28. Verify addresses included in the Routing Header address list. A warden can determine whether the listed addresses are actual network nodes, moreover the nodes should be routers as only routers perform packet forwarding. Similarly to the previous rule and to ingress/egress filtering rules, the location of the listed nodes should match the packet's direction of travel.

29. If the warden has access to an appropriate IPsec security association, perform full IPsec processing. Discard packets as dictated by IPsec specification. Raise alerts as appropriate.

30. If a warden does not have access to observed IPsec communications, she can still perform some of the IPsec processing. Verify that indicated SAs are in use. Perform sequence number check as before. Verify the size of Integrity Check Value.

31. Verify that *MTU* values carried by Packet Too Big messages are correct. A warden knowledgeable about network topology can verify that *MTU* values observed in packets are correct. Observed incorrect values are normalized based on the warden's knowledge of network topology.

32. Verify whether the header or the option indicated as erroneous by

Pointer field is really beyond sender's capabilities. An active warden can inspect the Parameter Problem message and decide whether the indicated problem really existed.

33. Verify addresses included in Home Agent Address Discovery Reply Message. A warden can verify that the included addresses point to actual nodes present on the network and that the nodes in question can actually perform as Home Agents supporting a mobile IPv6 node. Invalid addresses should be stripped from the packet.

The effects of the above rules are listed in Table 3.4.

field	**ID**	**result**	**description**
Traffic Class	1	partially defeated	*Traffic Class* is reset according to rule 23.; enforced adherence to known traffic class values defeats the attackers who directly embed data in the field; a clever attacker might devise a scheme that allows transmitting information via manipulating legitimate traffic class values; the available bandwidth is reduced from 8 bits per packet to 1 bit per packet
Flow Label	2	defeated	the field is reset according to rule 24.

field	**ID**	**result**	**description**
Hop Limit	5	defeated	the field is reset according to rule 25. and 26.
Source Address	6	partially defeated	the field is monitored according to rule 27.; covert channels that rely on inserting arbitrary values into *Source Address* are defeated; a clever attacker might devise a scheme that sends information by inserting legitimate node addresses; the channel bandwidth is reduced
Addresses	RH.12	partially defeated / defeated	the field is reset according to rule 28.; channels that insert data directly into the field are defeated, an attacker might devise a scheme that transmits data based on ordering of legitimate address entries; channel bandwidth is reduced; if the rule 17. is also applied, the channel is blocked at the cost of damaging the record of the packet's route

field	**ID**	**result**	**description**
All	AH.20	defeated	fake header is destroyed during IPsec processing according to rule 29.
All	AH.20	defeated	fake header is discarded during partial IPsec processing according to rule 30.
All	ESP.22	defeated	fake header is destroyed during IPsec processing according to rule 29.
MTU	ICMP.27	defeated	the field is reset according to rule 31.
Addresses	HAA.41	defeated	the field is monitored according to rule 33.; covert channels that insert arbitrary values into the field are defeated; attackers might devise a scheme that sends information via manipulating the order of legitimate entries; channel bandwidth is reduced; if the rule 22. is also applied, the channel is blocked

Table 3.4: Channels defeated by network-based normalization

3.4.3 Network Manipulation

In addition to the operations listed in Section 3.4.2, a network-aware active warden can also attempt to manipulate the network itself to assist with defeating covert channel communications. This study proposes two such manipulations:

multiple equivalent traffic classes: if the network uses traffic class functionality, Alice and Bob can abuse it for covert channel communication. An active warden described above can verify that monitored packets contain legal traffic classes, thus defeating some of the possible covert channels. However, if Alice and Bob agree to communicate by switching legal traffic class values, the warden's actions will have no effect as the covert communication takes place using only legitimate traffic classes. To defeat the more sophisticated attack the warden has to instruct the network to define multiple equivalent traffic classes to allow the warden to disrupt the covert channels through randomly altering packets' traffic class between the equivalent values.

variable routing: one of the problems active wardens face is the fact that Alice can make various modifications to the traffic and then rely on Bob to "fix" the traffic to minimize the impact of the alterations. This puts wardens at a disadvantage when compared to Alice and Bob since Wendy's changes cannot be undone and therefore Wendy has to be more concerned with syntax and semantics preservation. Wendy can

attempt to reduce the attackers' advantage by routing packets to their destinations via more than one route. If that is possible, Alice can no longer rely on Bob to intercept and restore the altered packets, and in order to protect the secrecy of their communication she has to switch to covert channels with semantics-preservation level of *final destination* or better.

3.4.4 Effectiveness of Countermeasures

Chapter 2 describes 41 covert channels discovered in IPv6 protocol. An active warden performing traffic normalization as defined above, defeats 25 of the described channels. Since the traffic normalizer performs only safe normalizations, there are no side-effects. Additionally, two channels are partially defeated with their bandwidth reduced as a result. If the warden includes the aggressive normalization rules, additional 5 channels are defeated, but at the cost of disabling some network functionality. An MRF-based active warden can replace some of the aggressive normalizations and add new defenses. As a result, an MRF-based warden can defeat 3 of the channels defeated by aggressive normalizations, without disabling associated network functionality, although one of the channels is only partially defeated. Moreover, some side-effects are still present, even though they are less severe. It can defeat 4 other channels as well. It is worth to note that some of the defenses require maintaining warden state across packets and network flows.

Altogether, the specification-based defenses are able to completely elimi-

nate 28 covert channels without incurring any side-effects, partially eliminate 2 channels, and eliminate 5 more channels by foregoing certain network functionality. Alternatively, 28 channels can be blocked without side-effects, 3 channels can be partially blocked, and 3 can be blocked with side-effects lesser than outright disabling of functionality.

Network-aware active wardens can perform superior normalizations as a result of their network knowledge. They can completely eliminate 3 previously resistant channels, and they can eliminate without side-effects another 3 channels that previously induced side-effects. Furthermore, they can significantly reduce bandwidth of another 4 channels. If network manipulation is possible the effectiveness of countermeasures increases further.

There are 2 covert channels presented in Chapter 2 that are not defeated by the countermeasures listed above. These are: channel 3 and channel HBH.9. Their resistance to IPv6 active wardens is due to the fact that they are in reality upper-layer protocol channels. While both channels involve changing *Payload Length* field or its Jumbogram equivalent, their real functionality lies in manipulating payloads whose syntax and semantics are defined by a different specification than IPv6 protocol specification. Conceivably, they can be defeated by an approach similar to an IPv6 active warden, but geared towards the upper-layer protocols they manipulate.

ID	**normalization**	**aggressive normalization**	**MRF-based defense**	**network-based normalization**
1		defeated/FL		defeated/NM
2		defeated/FL	partially	defeated
3				
4	defeated			
5		defeated/SE	defeated/SE	defeated
6	partially			partially
HBH.7	defeated			
HBH.8	defeated			
HBH.9				
HBH.10	defeated			
RH.11	defeated			
RH.12		defeated/FL		defeated/SE
FH.13	defeated			
FH.14	defeated			
FH.15	defeated			
FH.16	defeated			
DH.17	defeated			
DH.18	defeated			
AH.19	defeated			
AH.20				defeated

ID	**normalization**	**aggressive normalization**	**MRF-based defense**	**network-based normalization**
ESP.21	defeated			
ESP.22				defeated
MH.23	defeated			
ICMP.24	defeated			
ICMP.25	defeated			
ICMP.26	defeated			
ICMP.27	partially	defeated/SE	defeated/SE	defeated
ICMP.28	defeated			
ICMP.29	partially	defeated/SE		partially
ICMP.30	defeated			
ICMP.31			defeated	
ICMP.32	defeated			
ICMP.33			defeated	
RR.34			defeated	
RR.35	defeated			
RR.36			defeated/SE	
RR.37	defeated			
RR.38	defeated			
HAA.39	defeated			
HAA.40	defeated			

ID	normalization	aggressive normalization	MRF-based defense	network-based normalization
HAA.41				defeated

Table 3.5: Effectiveness of countermeasures. FL – loss of functionality, SE – other side-effects, NM – network manipulation.

3.5 Warden Models

Network-aware active wardens described in this study obviously require knowledge about their surrounding network. The sections below present different models of the wardens' knowledge and their impact on wardens' capabilities.

3.5.1 Perfect Warden

The analysis of countermeasures presented in Section 3.4 assumes that network-aware active wardens posses all required knowledge of all relevant network parameters. These wardens are called *perfect wardens*, and they require the following information about the network:

- Addresses of all relevant network nodes.
- Hop distances to all relevant network nodes.

- MTU values of all relevant links.
- Distinction between routers and hosts present on the network.
- Home Agent capabilities of present routers.
- IPv6 header and option support status in all relevant nodes.
- All traffic classes in use in the network.
- Currently used flow labels for all relevant flows.
- IPsec security association information.

If an active warden lacks certain type of necessary information, some of the network-based normalizations will be impossible. This lack of information will not however affect other normalizations. For example, if Wendy does not have the information about hop distances on the monitored network, she will not be able to perform *Hop Limit* normalizations, all other normalizations will still be possible.

3.5.2 Locally perfect warden

A *perfect warden* described above will perform the best normalizations theoretically possible. However, gathering all required information about the global network is likely not feasible.

In contrast to the perfect warden, a *locally perfect warden* has information that is limited to the warden's local network. The limitation of its network

knowledge makes network information gathering more practical. For example, an active warden can have all necessary knowledge about an autonomous system and be placed in a way that allows it to monitor all incoming and outgoing traffic (see Figure 3.1).

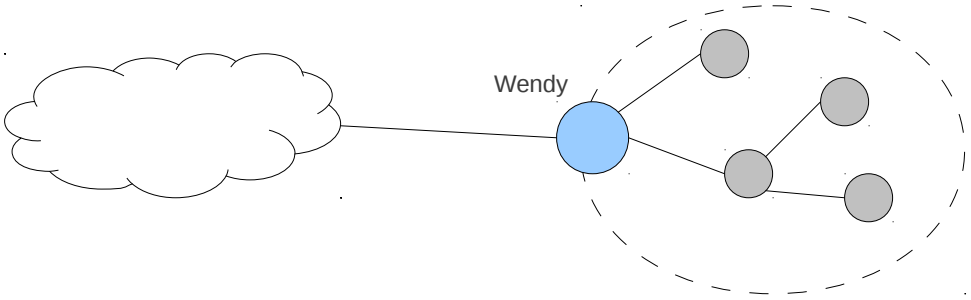

Figure 3.1: Local warden positioned to monitor incoming/outgoing traffic.

The fact that a locally perfect warden lacks some of the information necessary for network-based traffic normalizations will affect its capabilities. Generally, a warden can only normalize traffic that carries information about the network the warden is familiar with.

Assuming the scenario mentioned above as an example – where the warden has knowledge about an autonomous system and is placed so it can observe incoming and outgoing traffic – the limitation of its knowledge means that the normalizations can only be performed for outgoing or incoming traffic depending on the type of information the given packets carry.

For example, a Home Agent Discovery Reply message conveys information about addresses of routers capable of serving as a Home Agent for a mobile

IPv6 node. If such a message originates in the warden's network, the warden can verify whether the addresses listed are in fact home agents and normalize the message if needed. However, when the warden inspects a similar message coming from the outside network, the normalization is impossible because the warden's knowledge is not sufficient. An opposite situation takes place in case of *Hop Limit* normalizations. The *Hop Limit* field of incoming packets can be reset according to warden's knowledge, but outgoing packets cannot be modified the same way, as the warden does not know enough about the topology of the outside system.

rule	**direction**	**description**
23	incoming	only traffic classes belonging to local network can be verified; outgoing packet might carry traffic classes unknown to the warden
24	incoming/outgoing	flow labels belong to flows and can be observed by the warden regardless of flow direction
25	incoming	the warden does not know the distance to outside destinations
26	incoming	the warden does not know the distance to outside destinations
27	outgoing	only internal nodes addresses are known to the warden

rule	direction	description
28	outgoing	only internal nodes' addresses are known to the warden
29	incoming/outgoing	depends on the IPsec security context knowledge of the warden
30	incoming/outgoing	IPsec simulation can be performed for both flow directions
31	outgoing	only internal network MTUs are known to the warden
32	outgoing	only capabilities of internal nodes can be checked by the warden
33	outgoing	only internal home agents can be verified

Table 3.6: Effects of limited knowledge on traffic normalization

3.5.3 Multiple Wardens

A network-aware active warden can perform network-based traffic normalization if the observed traffic carries information about the network that the warden is familiar with. In consequence, it is possible that more than one warden contributes to a given packet's normalization as it traverses their respective networks. At the very least, a packet originating from one au-

tonomous system and traveling to another one could be inspected by two local wardens guarding the two respective systems. An additional factor in this scenario is that an *origin warden* will see the traffic as outgoing, while the *destination warden* will perceive the same traffic as incoming. As described in section 3.5.2, limited network knowledge will result in some of the proposed normalizations to become uni-directional only. However, since the two wardens perceive the same traffic as coming from different directions, together they can mitigate attacks that neither of them can defeat alone provided that the covert channel sender and receiver are placed within the wardens' networks.

3.6 Active Wardens and Tunneled Traffic

Section 2.6 discusses the possibilities for covert channel communication present when IPv6 traffic is tunneled using either the IPv6 or the IPv4 protocol. Since the tunneling has an impact on covert channel attacks, it can be expected to affect covert channel defenses as well.

The scenarios presented below assume that Wendy is positioned so that she can monitor the tunneled traffic.

IPv4 traffic in an IPv6 tunnel: This case does not fundamentally differ from the typical IPv6 protocol traffic. Encapsulated IPv4 traffic is treated as any other IPv6 payload. Active warden operates normally.

IPv6 traffic in an IPv4 tunnel: Since this study focuses on IPv6 covert

channels, the warden is concerned with the channels present in the inner (encapsulated) header. The covert channels are still present in the encapsulated traffic, but to monitor and defeat them the warden has to understand IPv4 protocol as well. The tunnel can effectively prevent Wendy from modifying the traffic if it is protected by IPsec. For details about IPsec impact, see Section 3.7.

IPv6 traffic in an IPv6 tunnel: In this scenario both inner and outer headers follow the IPv6 specification, hence both can carry covert data using IPv6 covert channel techniques. Effectively, the headers provide two independent covers for hiding information and should be inspected by Wendy. However, the main reason for transporting IPv6 traffic in an IPv6 tunnel is to provide security via IPsec used by the tunnel. It is therefore likely that the warden's modifications will be affected by IPsec security. See section 3.7.

3.7 Impact of Internet Protocol Security

Similarly to the effects on covert channels described in Section 2.7, the presence of IPsec has an impact on an active warden's operations. Since IPsec is designed to prevent an unauthorized modification of protected traffic, wardens have to take IPsec into account when altering packets or risk complete disruption of the overt communication when IPsec detects their modifications. The challenge facing Wendy is fundamentally similar to the one faced

by the attackers, Alice and Bob — how to modify the traffic without causing IPsec to drop the altered packets. Unlike Alice, however, Wendy cannot rely on Bob to undo the introduced changes.

The simplest scenario to consider is when Wendy has access to IPsec security association information. Since an active warden is a security mechanism, Wendy might obtain the necessary information from IPsec-communicating nodes, or perhaps from a central key management server. If the warden has access to the required IPsec keys, she can effectively operate as if the monitored traffic was not protected by IPsec and she can introduce modifications at will.

The remainder of this section considers scenarios present when Wendy does not have access to IPsec security context.

3.7.1 IPsec Transport Mode

IPsec transport mode is used when two communicating nodes protect their traffic end-to-end. In this mode, IPsec can ensure payload confidentiality as well as integrity of the payload and some parts of the IPv6 header. The impact of this protection is discussed in Table 3.7.

rule no.	**status**	**description**
1	active	most *Reserved* and similar fields present in IPv6 header are protected by IPsec; the fields present in IPv6 payload (e.g. ICMPv6) cannot be normalized if IPsec uses encryption
2	active	most fields affected by the rule are protected by IPsec; some of the fields might be encrypted and thus unmodifiable
3	active	removal of unknown headers and options violates IPsec ICV
7	active	Router Alert option is included in IPsec integrity protection
8	active	*MTU* field is protected by IPsec; it can also be encrypted and impossible to modify

Table 3.7: Traffic normalizations affected by IPsec

Despite the fact that the modifications introduced by the rules listed in Table 3.7 can violate IPsec integrity protection, the rules should remain active even in the presence of IPsec. Since the modifications restore the "normal" form of the traffic, there is a chance that the original form was improperly changed and was already violating IPsec integrity. The specific scenarios and their outcomes are considered below.

normal traffic: normal traffic is not modified and its IPsec protection is not affected

traffic damaged in transit: damaged traffic might actually be fixed by Wendy's modifications; if the warden does not repair the damaged traffic, her activity does not make the situation worse as the traffic was already damaged

improperly initialized traffic: traffic contains improperly initialized fields violating IPv6 specification; Wendy's modifications will restore specification compliance, but will be detected and discarded by IPsec, blocking the overt communication

traffic altered by Alice without knowledge of IPsec keys: traffic was already modified by Alice and her alterations violate IPsec integrity protection; Wendy's modifications will restore traffic integrity and remove the covert channel

traffic altered by Alice using IPsec keys: traffic was modified by Alice and she was able to recompute IPsec checksums; Wendy's modifications will block the covert channel, but cause IPsec to drop the packet

Table 3.8 lists aggressive traffic normalization rules affected by IPsec presence. Since some of these rules impact parts of the IPv6 payload as well as the IPv6 header, Wendy has to be concerned about both integrity protection and encryption mechanisms. The IPsec ESP header can provide payload

encryption and effectively hide certain fields from the active warden making their inspection and modification impossible.

rule no.	**status**	**description**
13	disabled	Routing Header is protected by IPsec AH ICV and its removal will be detected
14	disabled	ICMP *MTU* field is protected by IPsec AH ICV as well as ESP integrity protection and its modification will be detected by IPsec; additionally, ICMPv6 header belongs to IPv6 payload and is encrypted if ESP is used; as a result, IPsec encryption will mask *MTU* field entirely

Table 3.8: Aggressive traffic normalizations affected by IPsec

Even though the rule 14 should be disabled in the presence of IPsec, as indicated in Table 3.8, the impact of its absence is small as it is unlikely that ICMP Packet Too Big messages will ever be transmitted via IPsec transport mode.

rule no.	**status**	**description**
17	disabled	Routing Header is protected by IPsec AH ICV and the modification would be detected

rule no.	**status**	**description**
18	disabled	ICMP *MTU* field is protected by IPsec AH ICV; moreover it is a part of ESP payload and its encryption as well as integrity protection
19	disabled	*Data* field is protected by both IPsec AH and ESP
20	disabled	*SegmentNumber* field is protected by both IPsec AH and ESP
21	disabled	*MaxDelay* field is protected by both IPsec AH and ESP
22	disabled	the message is part of IPv6 payload and as such is protected by both AH ICV and ESP

Table 3.9: MRF-based defenses affected by IPsec

Unlike the traffic normalizations described above, the MRF-based defenses do not attempt to restore traffic to its "normal" state, instead their goal is to scramble possible covert communications by altering packets according to their Minimum Requisite Fidelity threshold. In consequence, any MRF-based rules prescribing modification of IPsec-protected fields will have to be disabled in the presence of IPsec or they will result in complete disruption of the IPsec-protected overt communication. The rules are listed in Table 3.9.

rule no.	status	description
31	active	*MTU* is protected by both AH ICV and ESP; if the payload is encrypted, the modification will be impossible
33	active	the message is part of IPv6 payload and as such is protected by both AH ICV and ESP

Table 3.10: Network-based normalizations affected by IPsec

Similarly to the specification-based normalizations, network-based normalizations attempt to restore the traffic to its "correct" form. The rules should therefore be active even in the presence of IPsec as there is a chance that the modifications made by Wendy actually recover IPsec compliance.

3.7.2 IPsec Tunnel Mode

When IPsec operates in tunnel mode, the inner traffic is entirely protected by IPsec mechanisms. If the tunnel employs encryption, Wendy will not be able to monitor the inner traffic at all as the payload content is scrambled.

If the IPsec tunnel does not use encryption, but only integrity protection, the warden's operations are still affected. Regardless of whether the tunnel uses AH or ESP integrity protection, the packet payload is always protected, so any modifications that Wendy makes will be affected even if they impact

header fields normally not included in IPsec checksum calculations.

As explained in the previous sections, the rules prescribing traffic normalizations should be active as they attempt to "fix" the traffic. The rules that serve to destroy covert channels by scrambling the traffic, should be disabled.

Chapter 4

cctool

A covert channel tool, named *cctool*, was created to verify the existence of the covert channels described in Chapter 2, as well as to test the functionality of active wardens proposed in Chapter 3. *cctool* can both capture live network traffic and process off-line capture files created by standard network tools like `tcpdump`.

This chapter is organized as follows. Section 4.1 examines existing tools that can be used for covert channel studies. Section 4.2 describes the design of *cctool* explaining the requirements, while Section 4.3 focuses on the details of the implementation. Finally, Sections 4.4 and 4.5 present experimental scenarios and their results.

4.1 Existing Tools

There exist numerous tools that can be used for covert channel studies, starting from basic network diagnostic tools that can be used to craft packets to more sophisticated covert channel tools that are specifically designed to provide covert communications. Similarly, there exist various tools that can gather network information. The sections below describe generic diagnostic tools, covert channel software and network information gathering packages.

4.1.1 Network Utilities

Netcat [38] is a networking utility capable of reading and writing data from/to a network. It exists in several variants implementing similar functionality with most versions supporting both TCP/IP and UDP sockets. Netcat is intended to be the most basic network utility and is designed to be a "back-end" tool to be controlled by other applications and scripts. In this capacity it is a basic building block of more complex systems and a very useful network exploration tool.

There exist several tools with functionality fundamentally similar to Netcat that extend its capabilities in various ways. *Socat* [72] is an extension of Netcat capable of using other network protocols (e.g. SSL). Moreover, it is designed to act as a network relay by opening two bi-directional network connections and transferring data between them. *Ncat* [31] is a more straightforward re-implementation of Netcat extended to allow use of SSL

protocol.

hping [75] is a packet generating utility initially designed to send and analyze TCP/IP packets. Its interface is intended to be similar to well-known ping tool, but unlike ping, hping is able to send more than simply ICMP echo requests. It supports generating TCP, UDP, ICMP and raw IP packets. Currently at version 3, hping is scriptable using Tcl and can be very useful for writing proof-of-concept exploits, including demonstrating covert channels. Hping requires patches to enable IPv6 support.

Scapy [9] is another example of a packet generator. It is a powerful interactive packet crafting application able to send or receive arbitrarily constructed packets in a number of network protocols. An important functionality of Scapy is its ability to both create and receive illegally formed packets as well as injecting packets into ongoing network sessions. Initially IPv6 was supported via a separate project named Scapy6 since then merged into Scapy itself.

SendIP [66] is a packet crafting tool that allows the user to send arbitrary IP packets. While not as flexible as Scapy or hping, it has a large number of options to specify the content of the sent packets. It allows to send intentionally malformed packets as well.

4.1.2 Covert Channel Tools

Project Loki [23, 24] explores the concept of ICMP tunneling, exploiting covert channels through the data portions of the ICMP_ECHO and ICMP_ECHO-

REPLY packets. The Loki client allows a remote attacker to wrap and transmit commands in ICMP payloads. Lokid, the Loki server, unwraps and executes the commands, sending the results back wrapped in ICMP packets. Project Loki can also run over UDP on port 53, simulating DNS traffic.

Similarly, Back Orifice 2000 with the BOSOCK32 plug-in and itun [14] implement covert channels via ICMP ECHO messages. Another closely related tool is ptunnel (Ping Tunnel) [83]. In addition to features implemented by itun, it ensures covert communication reliability by detecting and retransmitting lost packets. It is also able to handle multiple simultaneous communications and it can authenticate the communicating party. In order to control the number of concurrent en route packets, ptunnel implements send and receive window mechanism, fundamentally similar to the one employed by TCP.

Another ICMP tunnel backdoor tool is sneaky-sneaky [68]. Its features include encryption of the tunneled traffic and spoofing of the source address in the IPv4 header to maintain the anonymity of the communicating party. The true origin address is placed instead inside the encrypted message payload. Sneakin [80] provides an incoming shell through outgoing Telnet-like traffic. All of the tools described above were designed to handle only IPv4 traffic and are not capable of using IPv6.

Unlike the above programs, VoodooNet (or v00d00n3t) implements a similar set of capabilities as ICMP tunnel tools but uses ICMPv6 instead. It relies on ICMPv6 ECHO messages and also uses IPv6 header flow label field.

It provides both text chat and file transfer functionality.

Packet Transmogrifier [56] is a protocol steganography tool that can embed and extract covert messages into passing network traffic. Its purpose is to study application-level protocol steganography and it is therefore focused on upper-layer protocols like SSH, and not on IP layer headers.

4.1.3 Topology Information Gathering

Among the approaches and technologies that gather topology information with the purpose of detecting undesired traffic on the network are active mappers [78], NetFlow [16], network monitors such as Ntop [27], and certain implementations of the Simple Network Management Protocol (SNMP) [13], such as IBM Tivoli NetView [45], HP OpenView Network Node Manager [39], Marconi ForeView, and Sun Solstice Site Manager [84]. Shankar and Paxson [78] proposes an alternative approach to traffic normalizers [37] called active mappers that minimizes the performance penalties caused by packet reassembling. Active mapping involves building profiles of the network topology and the TCP/IP policies of hosts to help NIDSs disambiguate the interpretation of network traffic. The mappers gather topology information actively, sending specially crafted probing messages to each host on the network. Ntop, from www.ntop.org, is a traffic measurement and monitoring system with an embedded NIDS that gathers certain information about network topology and host relationships [27]. Ntop learns about topology based on network flows, so its knowledge of the topology actually depends

on the existence of these flows. Therefore, the view of the topology drawn by Ntop might be incomplete in certain situations (for example, when flows traveling to adjacent subnets do not pass by the system). NetFlow, whose version 9 supports IPv6, provides several services the most important being flow recording. It also provides information about traffic routing. The commercial SNMP products provide an understanding of the physical network topology through different information gathering mechanisms.

4.2 Design

cctool was envisioned as a comprehensive IPv6 covert channel tool capable of testing channels discovered in the process of this study. As a result, it was required that *cctool* can be used by Alice and Bob to communicate covertly, as well as by Wendy to perform covert channel disruption. The main requirements were:

- *cctool* must be able to intercept and modify IPv6 packets en route.
- *cctool* must also allow off-line processing of traffic captured independently.
- For live traffic capture, *cctool* must be deployable in locations along the traffic path.
- *cctool* must be able to embed covert data into IPv6 packets, extract covert data from the packets, as well as perform active warden functions

as described earlier.

- *cctool* must allow easy extensibility in adding new covert channels to its covert communication and active warden modules.

The architecture shown in Figure 4.1 was implemented to satisfy the objectives above. The architecture separates *cctool* functionality into several loosely coupled modules connected via abstract interfaces.

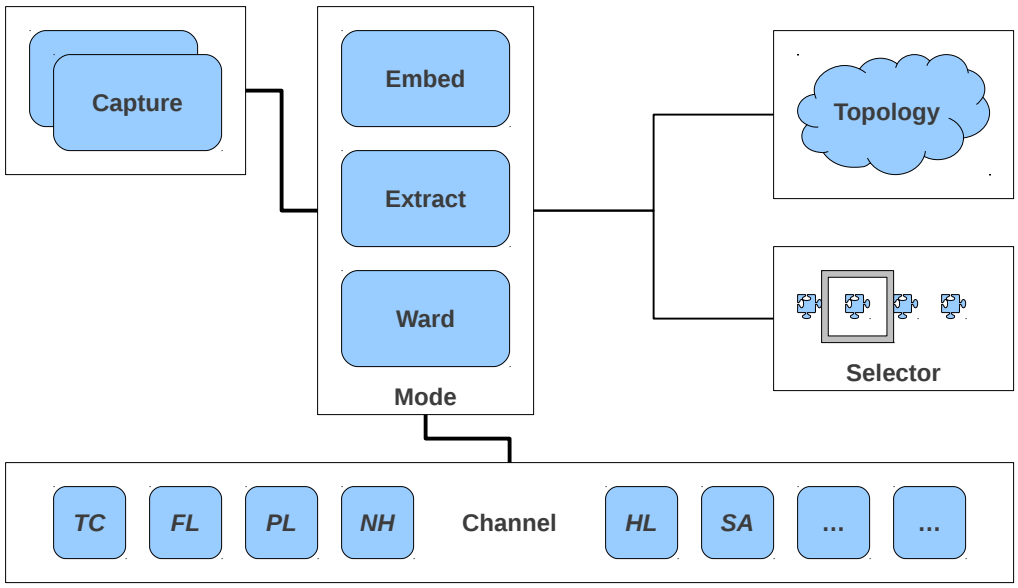

Figure 4.1: Architecture of *cctool*.

Packet `Capture` module handles the first three requirements. Since they specify two different ways of traffic acquisition, two different implementations of the module exist, one handling the live traffic capture and the other for processing off-line captures. To allow deployment in multiple locations, live

traffic capture uses Linux kernel mechanism and is therefore Linux compatible.

`Mode` module performs internal packet dispatching. Since the packet processing differs between data embedding, extraction and warden functionality, these are implemented as different `Modes`. Generally a `Mode` will consult with `Topology` and `Selector` modules to establish whether a packet is to be processed further. If the packet is to be processed, `Mode` will delegate packet processing to its current `Channel` instance. `Channel` instances understand the specifics of covert channels and are used by current `Mode` to perform covert channel-related functions like data embedding, extraction, etc.

`Topology` module provides data about the surrounding network. The data can be acquired dynamically by inspecting captured packets, or it can be provided by configuration. `Selector` module is designed to select packets for covert channel communication. It can use `Topology` module if network information is needed for making the decision.

Generally, the data flow through *cctool* architecture occurs similarly for both live and off-line traffic acquisition. For the live traffic, a packet is captured from the wire by the packet capture mode, then it is passed to the current `Mode` instance. The `Mode` module will pass the packet to the `Topology` module to allow it to update its network information. It will then prompt the `Selector` module to establish whether the packet should be processed further. If the `Selector` module decides that the further processing is not desirable, the packet is returned to the `Capture` module and re-inserted

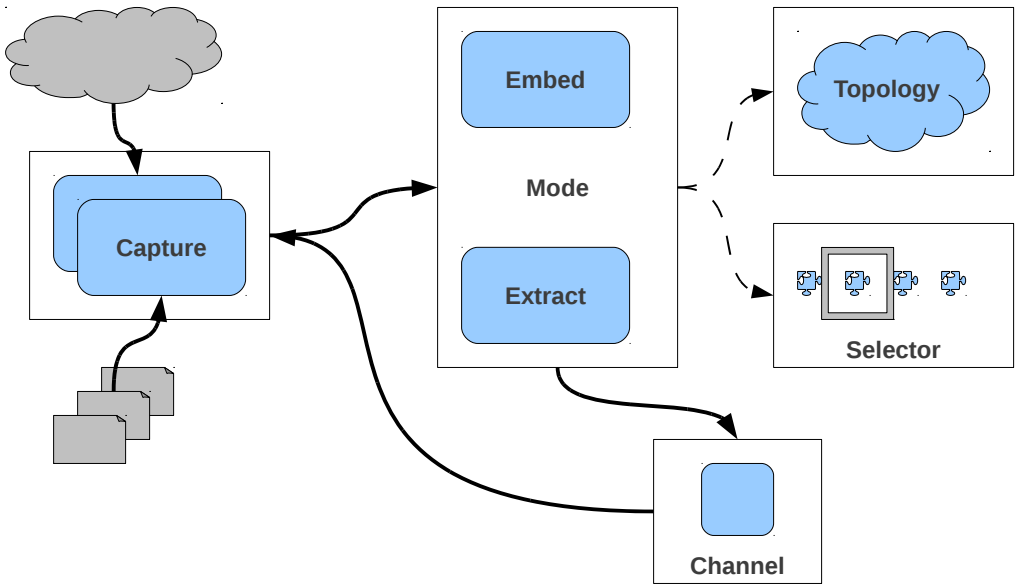

Figure 4.2: Flow of data in covert channel mode.

unchanged. If further processing is to be performed, it is delegated to the `Channel` instance that understands the details of currently selected covert channel. The modified packet is then returned to `Capture` module and injected back into the network.

The off-line traffic processing performs identically, except that the packets are acquired from a file and then written to another file.

The data flow is different when *cctool* operates in active warden mode. In this case, packets are acquired as before and passed to the `Mode` instance. Since the active warden mode assumes that all packets are to be processed, `Selector` module is not used. Typically, a warden is used to guard against all covert channels at once, so instead of a single `Channel` instance, it utilizes a

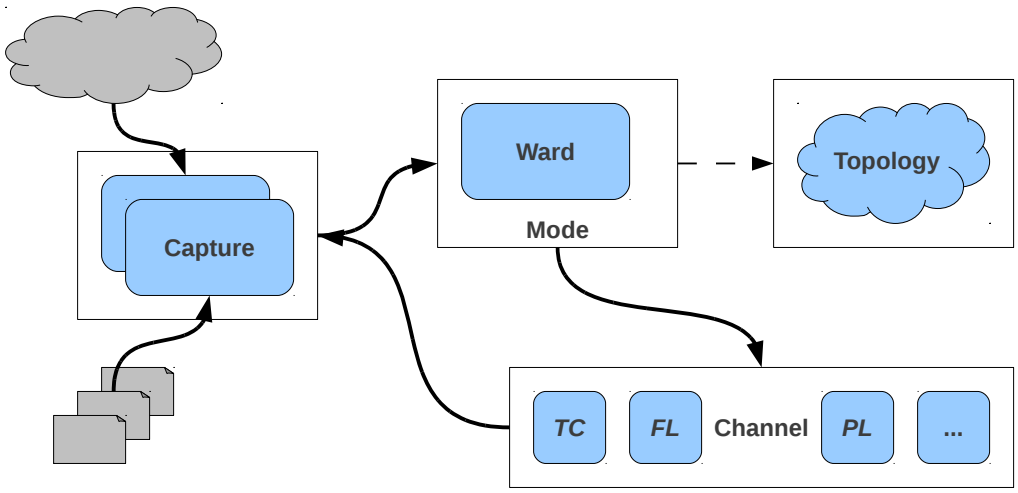

Figure 4.3: Flow of data in ward mode.

compound `Channel` object that can include all implemented covert channels. Packet re-insertion proceeds as previously described.

4.3 Implementation

cctool is implemented as a standalone C++ application. Most of *cctool* functionality is platform independent and relies only on standard POSIX C libraries. One exception is the packet capture module that utilizes `pcap` library for parsing off-line traffic files and `netfilter_queue` library for live traffic capture. Since `netfliter_queue` is only available for Linux kernel-based systems, the live traffic capture can only be deployed on Linux-based devices.

When operating off-line, *cctool* uses `pcap` library to acquire packets. The

packets are read from a pcap-compatible dump file, processed and then written to a new pcap file. Since many other network tools utilize the same file format, the traffic files can be obtained and processed in a variety of ways.

cctool's live capture functionality depends on `netfilter_queue` library which provides a userspace API for modifying packets that are queued by the Linux kernel. In order for the packets to be appropriately queued, *cctool* relies on `iptables` configuration, effectively becoming a part of standard Linux firewall.

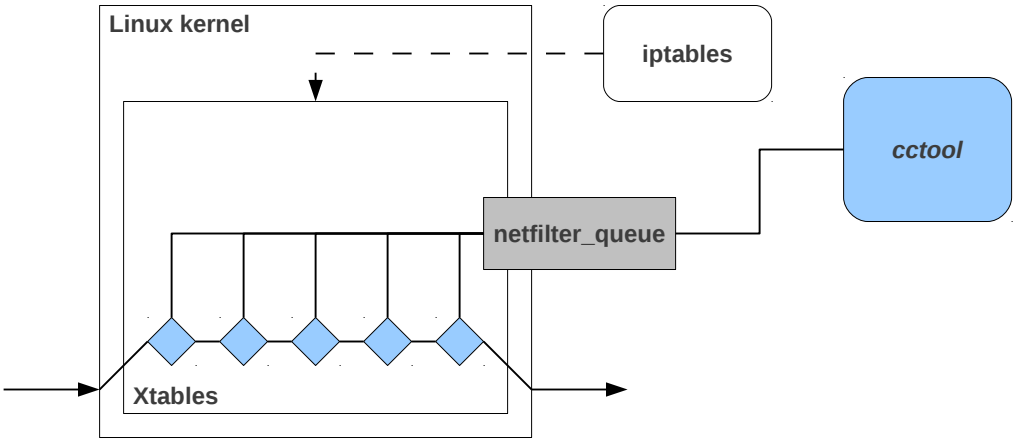

Figure 4.4: Interaction between Linux kernel components and *cctool*.

4.4 Experiments

Experiments conducted as part of this study focused on multiple aspects of covert channel communications and active warden functionality. Cover correctness and warden correctness tests were aimed to verify whether cover

traffic modified by Alice or Wendy is still semantically correct. Live traffic covert channel and warden scenarios were designed to check whether covert channel communications perform as expected, or are blocked as expected, by *cctool* operation. Finally, warden performance tests were intended to establish the impact that an active warden might have on network operations.

4.4.1 Cover correctness test

This scenario tests whether the cover traffic modified by Alice performs well enough to reach its destination and be understood by the receiver.

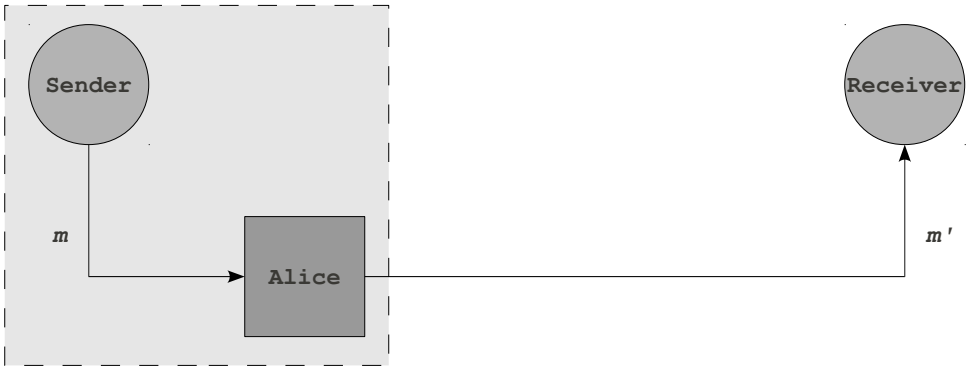

Figure 4.5: Cover correctness test.

In this scenario, the cover traffic can be provided by any IPv6-enabled application. For example, in order to test ICMPv6 traffic, ping6 tool can be used to generate standard ICMPv6 ECHO requests. Alice will modify the generated traffic to embed her covert message. Since there is no receiver (Bob), the traffic will continue to its destination in its modified form. If the modified traffic behaves correctly, the cover communication will func-

tion as usual. In the example using ping6 application, the destination node should receive modified ICMPv6 packets and be able to interpret them correctly despite their modification. If this is the case, the node will follow the normal ICMPv6 processing rules and respond with ICMPv6 ECHO_REPLY messages.

If the sender receives the ECHO_REPLY messages in response, the conclusion is that the covert message embedding by Alice does not disrupt the cover traffic enough to impede its normal functionality.

An equivalent tests can be performed using different applications to generate different form of cover traffic. Using a web browser will result in the cover traffic being TCP over IPv6. Similarly as before, a successful establishment of the connection and retrieval of a web page signifies that Alice's modification are not disruptive to the cover traffic.

A limitation of this test scenario is that not all covert channels described previously can be tested. Some of the channels depend on protocol messages or headers that are only present in specific situations that do not prompt a response from the receiver. For example, a covert channel utilizing a field that is only present in Packet Too Big ICMPv6 message, can only be tested using the required message. However, receiving a Packet Too Big message does not result in a response, making it impossible to verify whether the receiver has successfully read the message.

Example parameters of the test: m = ICMPv6 ECHO_REQUEST message(s), hop limit covert channel used, live traffic modification using netfilter

queue no 4, the covert message is contained in `message.txt` file.

Alice:

```
# configures iptables to pass outgoing traffic to
# queue #4
ip6tables -t mangle -A OUTPUT -j NFQUEUE --queue-num 4

# instructs cctool to attach to queue #4 and send
# the message contained in message.txt file using
# the hop limit (HL) covert channel
./cctool -cNFQ -aHL -Mmessage.txt -oSEND -q4
```

Sender:

```
ping6 www.example.org
```

If the standard ping reply messages are displayed, the test was successful.

4.4.2 Live traffic test

This scenario tests covert message embedding into live traffic. The traffic is then re-inserted into the network and the message should be read by the receiver.

As before, the cover traffic can be provided by any IPv6-enabled application. In the example below, we will use curl, a command line HTTP tool.

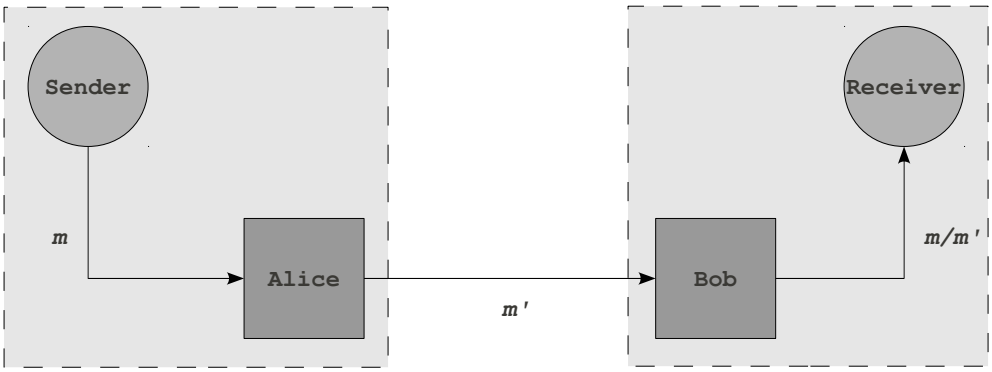

Figure 4.6: Live traffic test.

Alice will modify the passing traffic embedding her covert message. The traffic will then pass by Bob, who will extract the message. Finally, the traffic should reach the receiver. If the modifications performed by Alice were either sufficiently benign or were undone by Bob, the receiver should be able to correctly interpret the communication and provide requested content.

In this case, Bob should be able to display the covert message while the receiver should be able to provide the requested web page.

An advantage of this test over the covert correctness test described previously is that since both sender and receiver are located on a controlled network, it is possible to test all covert channels. However, unlike the previous test, this scenario does not provide information about the behavior of independent Internet nodes.

Example parameters of this test: m = TCP over IPv6 traffic, hop limit covert channel used, live traffic modification using netfilter queue no 1, the covert message is provided in the command line.

Alice:

```
# configures iptables to pass outgoing traffic to
# queue #0
ip6tables -t mangle -A OUTPUT -j NFQUEUE --queue-num 0

# instructs cctool to attach to queue #0, send message
# "Test0123" using the traffic class (TC) covert channel
./cctool -cNFQ -aTC -mTest0123 -oSEND -q0
```

Sender:

```
curl http://www.example.org/
```

Bob:

```
# configures iptables to pass incoming traffic to
# queue #1
ip6tables -t mangle -A INPUT -j NFQUEUE --queue-num 1

# instructs cctool to attach to queue #1, and
# listen for messages on the traffic class (TC)
# covert channel
./cctool -cNFQ -aTC -oRECV -q1
```

If Bob correctly displays the test message and the requested web page content is shown, the test was successful.

4.4.3 Warden correctness test

This test scenario mimics the cover correctness test above, but instead of verifying whether covert channel embedding preserves cover integrity, it verifies whether the cover survives modification by an active warden. The traffic monitored by the active warden contains to covert channel embeddings to verify that a normal, unmodified network traffic can perform correctly in the presence of an active warden.

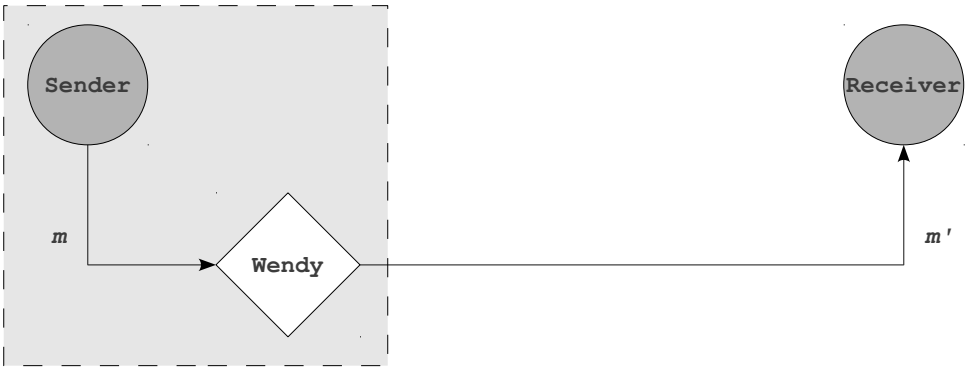

Figure 4.7: Warden correctness test.

Another difference present in this scenario is that while Alice usually employs a single covert channel to send her data, Wendy performs modifications blocking all channels.

Example parameters of the test: m = ICMPv6 ECHO_REQUEST message(s), live traffic modification using netfilter queue no 3.

Wendy:

```
# configures iptables to pass outgoing traffic to
```

```
# queue #3
ip6tables -t mangle -A OUTPUT -j NFQUEUE --queue-num 3

# instructs cctool to attach to queue #3 and ward all traffic
./cctool -cNFQ -oWARD -q3
```

Sender:

```
ping6 www.example.org
```

If the standard ping reply messages are displayed, the test was successful.

4.4.4 Warden covert channel test

This test is conducted similarly to the last one, but this time the traffic modified by the warden actually contains covert communication inserted by Alice.

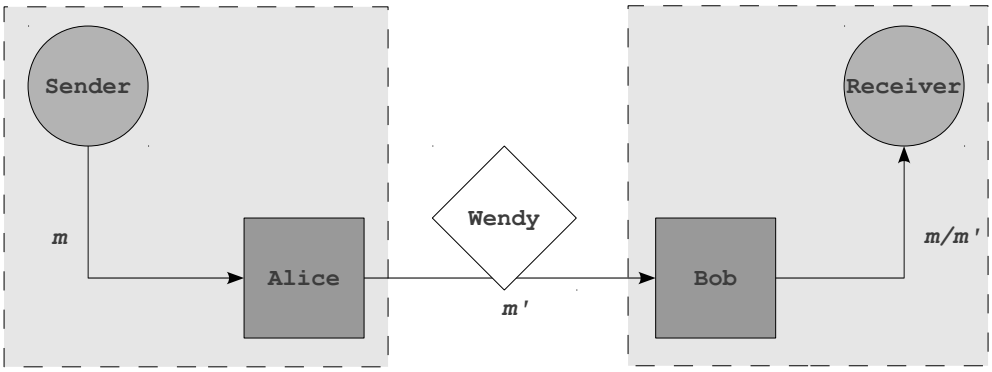

Figure 4.8: Covert channel warden test.

The goal of this test is to verify an active warden's effectiveness against actual covert channel attacks. The traffic reaching its destination should be stripped of covert messages injected by Alice.

Alice:

```
# configures iptables to pass outgoing traffic to
# queue #0
ip6tables -t mangle -A OUTPUT -j NFQUEUE --queue-num 0

# instructs cctool to attach to queue #0, send message
# "HelloWorld" using the traffic class (TC) covert channel
./cctool -cNFQ -aTC -mHelloWorld -oSEND -q0
```

Bob:

```
# configures iptables to pass incoming traffic to
# queue #1
ip6tables -t mangle -A INPUT -j NFQUEUE --queue-num 1

# instructs cctool to attach to queue #1, and
# listen for messages on the traffic class (TC)
# covert channel
./cctool -cNFQ -aTC -oRECV -q1
```

Wendy:

```
# configures iptables to pass forwarded traffic to
# queue #8
ip6tables -t mangle -A FORWARD -j NFQUEUE --queue-num 8

# instructs cctool to attach to queue #0, ward traffic
./cctool -cNFQ -oWARD -q0
```

Sender:

```
curl http://www.example.org/
```

As a result, Alice's covert message should be blocked before reaching Bob and the cover communication (web page retrieval) should be only disrupted as described in Chapter 3.

4.4.5 Warden performance test

Unlike the test scenarios described previously, this scenario is concerned with measuring active warden's effect on network performance.

To estimate the performance impact, the same network application is run with and without the presence of an active warden. The influence of the following parameters is tested:

- Size of modified packets — since an IPv6 active warden inspects only packet headers, the performance impact of a warden is expected to diminish as packet sizes grow; in other words, the impact depends on

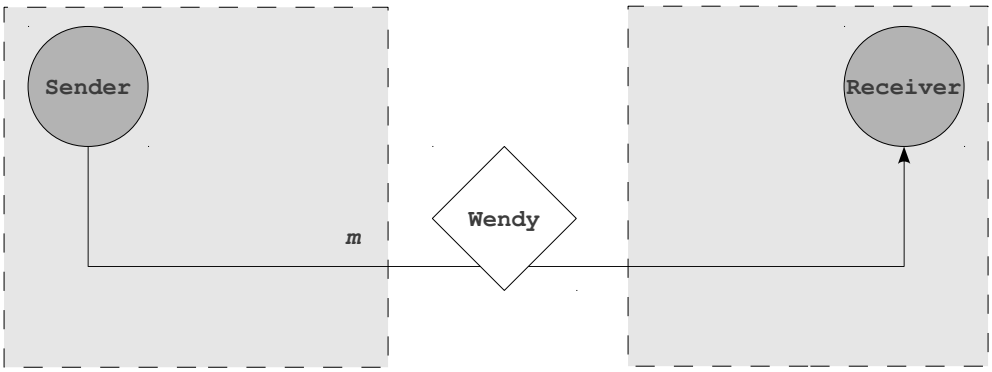

Figure 4.9: Active warden performance test.

the number of packets – or more strictly, headers – and not on the amount of data carried by the inspected packets.

- Presence of specific IPv6 extension headers — while most of IPv6 packets do not contain extension headers, the headers can be present and an IPv6 active warden will inspect them if that is the case. In consequence, the effect that the warden has on network performance is expected to increase when the extra headers are present.

- Size of network guarded by the active warden — since a network-aware warden depends on its network knowledge, it is interesting to see whether the size of the guarded network, and by extension the size of the required network information, has an impact on the warden's performance.

Wendy:

```
# configures iptables to pass incoming/outgoing traffic to
```

```
# queue #8
ip6tables -t mangle -A INPUT -j NFQUEUE --queue-num 8
ip6tables -t mangle -A OUTPUT -j NFQUEUE --queue-num 8

# instructs cctool to attach to queue #0, ward traffic
./cctool -cNFQ -oWARD -q0
```

4.5 Results

The results of the experiment scenarios presented in Section 4.4 are shown below.

4.5.1 Cover correctness test

The test was conducted against 10 Internet sites providing IPv6 connectivity using both `ping6` and HTTP requests.

The results are shown in Table 4.1. The tested covert channels generally performed as expected, with the exceptions noted below.

channel ID	**expected result**	**ICMPv6**	**HTTP over IPv6**
1	+	10 / 10	10 / 10
2	+	10 / 10	10 / 10
3	−	2 / 10	0 / 10
4	−	0 / 10	0 / 10

channel ID	**expected result**	**ICMPv6**	**HTTP over IPv6**
5	+	10 / 10	10 / 10
6	−	0 / 10	0 / 10
HBH.7	+	10 / 10	10 / 10
HBH.8	+	10 / 10	10 / 10
HBH.9	−	0 / 10	0 / 10
HBH.10	+	9 / 10	9 / 10
RH.11	+	10 / 10	10 / 10
RH.12	+	10 / 10	10 / 10
FH.13	+	10 / 10	10 / 10
FH.14	+	10 / 10	10 / 10
FH.15	+	9 / 10	9 / 10
FH.16	+	9 / 10	9 / 10
DH.17	+	10 / 10	10 / 10
DH.18	+	10 / 10	10 / 10
ICMP.30	+	5 / 10	N/A
ICMP.31	+	10 / 10	N/A

Table 4.1: Results of cover correctness test.

Channel 3 (*Payload Length*) was not expected to work in this scenario since it injects data into the IPv6 payload and likely disrupts the cover communication. Still, two tested sites returned ICMPv6 replies despite the mod-

ification. Channel HBH.10 (*Router Alert*) was expected to perform successfully, however one of the test sites did not respond. A possible reason is that the site filters packets containing Router Alert options.

Similarly one site did not respond as expected to packets containing channels FH.15 and FH.16. It is possible that the site uses a different packet re-assembly algorithm than the one assumed by *cctool* implementation.

Channels ICMP.30 and ICMP.31 could not be tested using HTTP requests as they require the presence of ICMPv6 headers for their operation. Unexpectedly, channel ICMP.30 worked correctly only in half of the tests. Since it uses the *Code* field of ECHO_REQUEST messages, it is possible that some implementations do not ignore the field's value as directed by the specification.

4.5.2 Live traffic test

Live traffic tests were conducted between two IPv6 endpoints connected to the Internet via independent IPv6 tunnels. The tests were conducted using ICMPv6 protocol as the cover traffic except when the channels required specially crafted packets, e.g. Mobility Header channels. All covert channels performed as expected.

4.5.3 Warden correctness test

The warden correctness test was again conducted against 10 Internet sites that provide IPv6 connectivity using both ICMPv6 and HTTP over IPv6 protocols. Similarly to the cover correctness tests described previously, not all covert channel countermeasures could be tested, as some channels (and their countermeasures) require the presence of IPv6 features that cannot be included in communication with independent Internet nodes.

Both `ping6` and HTTP performed as expected.

4.5.4 Warden covert channel test

Testing of active warden against actual covert channels attacks was conducted between two IPv6 endpoints on a controlled network. The warden was positioned so it can monitor all communications between the sender and the receiver.

All covert channels were affected as predicted in Chapter 3.

4.5.5 Warden performance test

The performance of *cctool* warden mode was evaluated by measuring the network throughput for different packet sizes, different extension headers present, and different network sizes, performing 10 measurements each time.

Table 4.2 shows the measured throughput loss caused by the presence of an active warden for packets of 64-octet length, of 1024-octet length, and of

1456-octet length traveling between the end points.

packet size	**throughput w/o warden**	**throughput with warden**	**performance loss**
64 octets	62.76 Mbps	51.21 Mbps	18.4%
1024 octets	91.1 Mbps	80.1 Mbps	12.1%
1456 octets	94.2 Mbps	85.9 Mpbs	8.8%

Table 4.2: Warden's impact on network throughput for different packet sizes.

Table 4.3 shows the impact that an active warden has on network throughput when different IPv6 extension headers are present.

extension header	**performance loss**
none	14.1%
Hop-by-Hop Options Header	14.2%
Routing Header	16.8%
Fragment Header	39.1%
Destination Options Header	15.0%
Authentication Header	—%
Encapsulating Security Payload	—%

Table 4.3: Warden's impact on network throughput for different extension headers present.

The last test measured the differences in network throughput when the packets traversed networks composed of 10 and 1000 hosts. While the presence of an active warden impacted the performance as described previously, the impact did not depend on the size of warden's network information. For both small and large network, the performance loss was identical.

The conclusions from the tests are:

1. The relative slowdown introduced by the active warden decreases as the packet size grows. As mentioned before, this is likely caused by the fact that the warden inspects only packets' headers, not packets' payload, hence its impact diminishes as packets size increases.

2. The presence of IPv6 extension headers obviously affects the warden's performance since if present the headers have to be examined. As shown above, some headers have a smaller relative impact, e.g. the Hop-by-Hop Options Header, likely explained by the fact that these are the headers also examined by the intermediate nodes during the normal network processing, making the warden's overhead relatively smaller. Moderate impact of IPsec headers is probably due to the fact that costly IPsec processing is performed by the endpoints, again making the delays introduced by the warden relatively smaller. On the other hand, the biggest difference was seen in the case of the Fragment Header and it is caused by the time consuming re-assembly procedure.

3. The size of simulated network topology does not influence the warden's

performance. This is not a surprising outcome because the data structures used by the warden to store network information exhibit constant lookup times. The increased size of the network does force the warden to store more information, but the increased storage demand does not translate into lower throughput.

Chapter 5

Conclusions and Future Work

The purpose of this study was to discover whether possibilities for network storage covert channels exist within the IPv6 protocol structure and, if their existence was shown, whether techniques based on active wardens would prove effective against them. In order to achieve this objective, an investigation of the IPv6 protocol was conducted, and following the discovery of 41 potential covert channels, existing active warden methodology was analyzed. Finding that specification-based countermeasures cannot defeat some of the more sophisticated attacks, an extension to the current active warden methodology was proposed.

To summarize the implications of the findings, this chapter is organized as follows. Section 5.1 draws closing statements about the significance of the study. Section 5.2 details the main contributions. Finally, Section 5.3 suggests possibilities for future research.

5.1 Conclusions

The importance of the IPv6 protocol as the mainstay of the next generation of the Internet is steadily growing as the supply of available IPv4 addresses continues to shrink. With IPv6 adoption as the dominant Internet protocol coming ever closer, the danger posed by covert channels existing within it grows larger and larger. This study examining the IPv6 protocol discovered a potential for existence of multiple covert channels and identified 41 possible network covert channel attacks threatening the information security of any IPv6-based network.

The attacks are made possible by the fact that the correct values of IPv6 protocol fields are not always known and contain certain amount of entropy. As a result, a covert channel attacker can take advantage of the available entropy and inject foreign data into protocol fields to be carried to the covert channel receiver. The IPv6 protocol fields used by the attacks were analyzed to ascertain the available entropy and classified according to the levels of *syntax* and *semantics preservation* of the covert channels involved.

Following the discovery of the covert channel attacks, this study focused on the investigation of possible countermeasures. First, classic methodologies such as traffic normalizers and active wardens based on the IPv6 specification were considered. These approaches attempt to mitigate covert channel attacks by either reducing available covert channel entropy using the protocol specification as a reference, or conversely, take advantage of the uncertainty

to inject their own data into the IPv6 traffic therefore overwriting possible covert communications. Adapting these techniques to the IPv6 protocol yielded a solution that was able to defeat some but not all of the described channels.

To mitigate the remaining covert channels, this study proposed an extension to the active warden technology. Since many of the resistant channels were utilizing protocol fields that carry network related data, which cannot be easily normalized nor scrambled based on the specification alone, the extension suggested adding network knowledge in order to aid the wardens in decreasing available entropy and mitigating covert channels. The network-based mitigation was able to defeat previously resistant channels and improve the results in several channels that were only partially defeated by the simpler countermeasures. This approach is made possible by the fact that the IPv6 protocol is a structured medium and its headers are well-defined semantically. Moreover, the information IPv6 headers carry naturally concerns the networks they utilize. In consequence, an active warden that possesses adequate knowledge about its network can decrease the uncertainty as to the correct protocol field values and eliminate covert channels that might attempt to exploit them.

A similar approach is certainly possible for combating covert channels in other layers of the protocol stack as long as similar conditions hold. First, protocol fields potentially abused by covert channel attackers have to be well-defined both syntactically and semantically, and second, they have to carry

information about the network, or at least an environment that the warden is knowledgeable about. These conditions generally exclude protocol payloads as they might not carry a well-defined structure and their semantics are often unknown to the active warden.

Finally, *cctool* was developed to experimentally verify the existence of the described covert channels, as well as the performance of the proposed active wardens. Tests conducted on live IPv6 Internet confirmed the feasibility of the covert channel attacks, and testing the performance of the active warden implementation proved that an active warden is capable of defeating the covert channel attacks with an acceptable network performance impact.

As a result of this study, the existence of network storage covert channels in the IPv6 protocol is demonstrated and effective countermeasures are presented, showing the potential of applying network knowledge in IPv6 covert channel mitigation.

5.2 Summary of Contributions

The main goal of this study was to investigate possibilities for covert channel attacks against IPv6-based networks and, if necessary, to research possible countermeasures against these attacks. The research conducted as part of the study resulted in the following contributions:

1. Discovery of 41 network storage covert channels in the IPv6 protocol and its companion protocols like ICMPv6. These covert channels can

be used to conduct attacks on any network that deploys the IPv6 protocol, presenting a data ex-filtration and remote control threats to the network's security.

2. Analysis of the affected protocol fields using concepts of syntax and semantics preservation originating from the domain of protocol steganography. The concepts themselves were extended to better describe covert channel attacks performed via a protocol with many intermediate nodes whose understanding of syntax and semantics varies. The covert channels were classified according to their syntax and semantics preservation levels.

3. Adaptation of existing, specification-based traffic normalization and active warden technologies to combat IPv6 covert channels. The effectiveness of the updated techniques was evaluated against the described covert channels.

4. Extension of the active warden concept to allow wardens to fight more sophisticated channels or defeat previously defeated channels with less detrimental side-effects. The key expansion methodology was to equip wardens with the knowledge of their surrounding network allowing them to decrease the entropy present in IPv6 protocol fields.

5. Creation of a covert channel network tool, named *cctool*, that was used to both conduct covert channel experiments on the live network as well

as off-line processing of captured traffic. The tool can also perform as an active warden and was used to determine the impact that an active warden operations might have on the network performance.

5.3 Future Work

The study of IPv6 covert channels and network-aware active wardens is a new area and it presents several possibilities for future research. For each topic explored during this investigation, there exist potential avenues for further inquiry.

The extensible nature of the IPv6 protocol designed to allow future protocol extensions without obsoleting existing network hardware is a powerful feature, but it also makes it easy for attackers to inject arbitrary data into the network traffic with assurance that the traffic will perform as expected. It brings the question whether the protocol designers should consider the potential presence of covert channels and attempt to minimize it during the design process. Another example of the protocol design impacting covert channel possibilities is presented by IPsec. If IPsec packet integrity algorithms were created to make the integrity check possible without full knowledge of the IPsec security association, IPsec covert channels could be defeated without the disclosure of security information.

Network-aware active wardens follow an established methodology of acquiring and perfecting the understanding of the cover and its properties to

allow detecting and destroying any data that could be hidden within. In this study, network knowledge provided the necessary edge in fighting network covert channels. Two obvious questions remain: Is a similar methodology applicable to other protocols? Is there other information that could be applied to defeat IPv6 covert channels?

It seems perfectly plausible that the network knowledge would be applicable for combating covert channels in other protocols, especially the protocols that by design carry information about the network they operate within. However, even higher-level protocols that are less concerned with the underlying network infrastructure might form interesting topologies and by design or by accident carry information about them. For example HTTP protocol has long since evolved from a simple client-server topology to a more complex environment consisting of proxies, front proxies, load balancers, etc.

A network-aware active warden can perform network-based traffic normalization if the observed traffic carries information about the network that the warden is familiar with. While the origin and the destination of a packet do influence what network information might be carried, the information itself is the deciding factor. As a result, a warden knowledgeable about a mid-point network might be able to normalize certain packet fields if the information included is relevant to the warden's network. In consequence, it is possible that many wardens might contribute to a given packet's normalization as it traverses their respective networks. The effectiveness of the normalizations they perform depends on whether any of the wardens know

about the network relevant to the network information the packet carries. For example, in case of *Hop Limit* normalizations, it is enough that one of the wardens knows enough about the topology of the packet's destination to set appropriate field values. It is not required that a single warden has knowledge of the entire packet's path. In consequence, it is possible that a set of independent wardens might be able to approximate the performance of a perfect warden if the sum of their network knowledge approximates the knowledge of a perfect warden.

cctool software could be expanded in several different directions as well. The most obvious possibility is to include other protocols beside IPv6. TCP and UDP provide natural goals for such expansion. It appears that the race between ever more sophisticated covert channel attacks and increasingly robust covert channel countermeasures is to continue and *cctool* can provide a useful tool in the study of channels and their countermeasures beyond just the IPv6 protocol.

Bibliography

Bibliography

[1] Christopher Abad. IP Checksum Covert Channels and Selected Hash Collision. http://gray-world.net/cn/papers/ipccc.pdf, 2001.

[2] J. Abley, P. Savola, and G. Neville-Neil. Deprecation of Type 0 Routing Headers in IPv6. Retrieved on December 31, 2010 from the World Wide Web: http://www.ietf.org/rfc/rfc5095.txt, December 2007. RFC 5095.

[3] Kamran Ahsan. Covert channel analysis and data hiding in tcp/ip. Master's thesis, University of Toronto, 2002.

[4] Kamran Ahsan and Deepa Kundur. Practical data hiding in TCP/IP. In *Proceedings of the ACM Workshop on Multimedia Security at ACM Multimedia*, December 2002.

[5] Ross Anderson. Stretching the limits of steganography. In Ross Anderson, editor, *Information Hiding: Proceedings of the First International Workshop*, pages 39–48, Cambridge, U.K., May 30-June 01, 1996. Springer.

[6] Ross J. Anderson and Fabien A.P. Petitcolas. On the limits of steganography. In *IEEE Journal of Selected Areas in Communications: Special Issue on Copyright and Privacy Protection*, pages 474–481, May 1998.

[7] J. Arkko, J. Kempf, B. Zill, and P. Nikander. SEcure Neighbor Discovery (SEND). Retrieved on December 31, 2010 from the World Wide Web: http://www.ietf.org/rfc/rfc3971.txt, March 2005. RFC 3971.

[8] Matthias Bauer. New covert channels in HTTP - adding unwitting web browsers to anonymity sets. In Pierangela Samarati and Paul Syverson, editors, *Proceedings of the 2003 ACM Workshop on Privacy in the Electronic Society*, pages 72–78, Washington, DC, USA, October 30, 2003. ACM Press. ISBN 1-58113-776-1.

[9] Philippe Biondi. Scapy. Retrieved on July 7, 2010 from the World Wide Web: http://www.secdev.org/projects/scapy/, 2010.

[10] D. Borman, S. Deering, and R. Hinden. IPv6 Jumbograms. Retrieved on December 31, 2010 from the World Wide Web: http://www.ietf.org/rfc/rfc2675.txt, August 1999. RFC 2675.

[11] R. Braden, D. Borman, and C. Partridge. Computing the Internet Checksum. Retrieved on December 31, 2010 from the World Wide Web: http://www.ietf.org/rfc/rfc1071.txt, September 1988. RFC 1071.

[12] Serdar Cabuk, Carla E. Brodley, and Clay Shields. IP covert timing channels: Design and detection. In *Proceedings of the 11th ACM Confer-*

ence on Computer and Communications Security, pages 178–187, Washington DC, USA, 2004. ACM Press.

[13] J. Case, M. Fedor, M. Schoffstall, and J. Davin. A simple network management protocol (SNMP). Retrieved on January 13, 2005 from the World Wide Web: http://www.ietf.org/rfc/rfc1157.txt, May 1990. RFC 1157.

[14] Sergey Chavlyuk. itun - simple icmp tunnel. Retrieved on July 8, 2010 from the World Wide Web: http://sourceforge.net/projects/itun/, 2005.

[15] ChinaView. China, EU to build wide-band network. Retrieved on January 12, 2006 from the World Wide Web: http://news.xinhuanet.com/english/2006-01/12/content_4045153.htm, 2006.

[16] Cisco. Cisco IOS NetFlow. Retrieved on November 17, 2005 from the World Wide Web: http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html, 2005.

[17] Wesley K. Clark and Peter L. Levin. Securing the information highway. *Foreign Affairs*, 88(6):2–10, 2009.

[18] A. Conta. Extensions to IPv6 Neighbor Discovery for Inverse Discovery Specification. Retrieved on December 31, 2010 from the World Wide Web: http://www.ietf.org/rfc/rfc3122.txt, June 2001. RFC 3122.

[19] A. Conta, S. Deering, and M. Gupta. Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. Retrieved on December 31, 2010 from the World Wide Web: http://www.ietf.org/rfc/rfc4443.txt, March 2006. RFC 4443.

[20] Scott Craver. On public-key steganography in the presence of an active warden. In David Aucsmith, editor, *Information Hiding: Proceedings of the Second International Workshop*, pages 355–368, Portland, Oregon, U.S.A., April 14-17, 1998. Springer.

[21] M. Crawford. Router Renumbering for IPv6. Retrieved on December 31, 2010 from the World Wide Web: http://www.ietf.org/rfc/rfc2894.txt, August 2000. RFC 2894.

[22] M. Crawford and B. Haberman. IPv6 Node Information Queries. Retrieved on December 31, 2010 from the World Wide Web: http://www.ietf.org/rfc/rfc4620.txt, August 2006. RFC 4620.

[23] daemon9 (route@infonexus.com). LOKI2 (the implementation). Phrack Magazine, 51, article 6, September 1997. http://www.phrack.org/show.php?p=51&a=6.

[24] daemon9 (route@infonexus.com) and alhambra (alhambra@infornexus.com). Project LOKI. Phrack Magazine, 49, article 6, August 1996. http://www.phrack.org/show.php?p=49&a=6.

[25] S. Deering, W. Fenner, and B. Haberman. Multicast Listener Discovery (MLD) for IPv6. Retrieved on December 31, 2010 from the World Wide Web: http://www.ietf.org/rfc/rfc2710.txt, October 1998. RFC 2710.

[26] S. Deering and R. Hinden. Internet Protocol, version 6 (IPv6) specification. Retrieved on December 31, 2010 from the World Wide Web: http://www.ietf.org/rfc/rfc2460.txt, December 1998. RFC 2460.

[27] Luca Deri and Stefano Suin. Improving network security using Ntop. In *Third International Workshop on the Recent Adcances in Intrusion Detection, RAID 2000*, Toulouse, France, October 2000.

[28] Tom Dunigan. Internet steganography. Technical report, Oak Ridge National Laboratory (Contract No. DE-AC05-96OR22464), Oak Ridge, Tennessee, October 1998. [ORNL/TM-limited distribution].

[29] Gina Fisk, Mike Fisk, Christos Papadopoulos, and Joshua Neil. Eliminating steganography in Internet traffic with active wardens. In Job Oostveen, editor, *Information Hiding: Preproceedings of the Fifth International Workshop*, pages 29–46, Noordwijkerhout, The Netherlands, October 7-9, 2002. Springer.

[30] S. Frankel and H. Herbert. The AES-XCBC-MAC-96 algorithm and its use with IPsec. Retrieved on December 31, 2010 from the World Wide Web: http://www.ietf.org/rfc/rfc3566.txt, September 2003. RFC 3566.

[31] Chris Gibson, Kris Katterjohn, Mixter ¡mixter@gmail.com¿, and Fyodor ¡fyodor@insecure.org¿. ncat. Retrieved on July 10, 2010 from the World Wide Web: http://nmap.org/ncat/, 2010.

[32] John Giffin, Rachel Greenstadt, Peter Litwack, and Richard Tibbetts. Covert messaging through tcp timestamps. In *Second Workshop on Privacy Enhancing Technologies*, volume 2482 of *Lectures Notes in Computer Science*, pages 194–208, San Francisco, CA, USA, April 14-15, 2002 2003. Springer-Verlag Heidelberg.

[33] Global Summit IPv6. Retrieved on May 17, 2005 from the World Wide Web: http://www.ipv6-es.com/05/in/i-intro.php, 2005.

[34] Thomas Graf. Messaging over ipv6 destination options. http://net.suug.ch/articles/2003/07/06/ip6msg.html, July 6, 2003.

[35] B. Haberman and J. Martin. Multicast Router Discovery. Retrieved on December 31, 2010 from the World Wide Web: http://www.ietf.org/rfc/rfc4286.txt, December 2005. RFC 4286.

[36] Theodore Handel and Maxwell Sandford. Hiding data in the OSI network model. In Ross Anderson, editor, *Information Hiding: Proceedings of the First International Workshop*, volume 1174, pages 23–38, Cambridge, U.K., May 30-June 01, 1996. Springer.

[37] Mark Handley and Vern Paxson. Network intrusion detection: Evasion, traffic normalization, and end-to-end protocol semantics. In *Proceed-*

ings of the 10^{th} *USENIX Security Symposium*, Washington, DC, USA, August 13-17, 2001. USENIX Association.

[38] Hobbit ¡hobbit@atstake.com¿. netcat. Retrieved on July 10, 2010 from the World Wide Web: http://nc110.sourceforge.net/, 1996.

[39] HP. Network node manager advanced edition. Retrieved on November 17, 2005 from the World Wide Web: http://www.managementsoftware.hp.com/products/nnm/index.html, 2005.

[40] Geoff Huston. IPv4 Address Report. Retrieved on March 1, 2011 from World Wide Web: http://www.potaroo.net/tools/ipv4/, March 2011.

[41] IANA. IPv6 Addresses for the Root Servers. Retrieved on May 1st, 2010 from the World Wide Web: http://www.iana.org/reports/2008/root-aaaa-announcement.html, January 2008.

[42] Internet Assigned Numbers Authority (IANA). Protocol numbers. Retrieved on December 31, 2010 from the World Wide Web: http://www.iana.org/assignments/protocol-numbers, November 2010.

[43] Internet Assigned Numbers Authority (IANA). IP Version 6 Address Space. http://www.iana.org/assignments/ipv6-address-space, October 2010.

[44] Internet Assigned Numbers Authority (IANA). IP Version 6 Parameters. http://www.iana.org/assignments/ipv6-parameters, May 2010.

[45] IBM. Tivoli NetView. Retrieved on November 17, 2005 from the World Wide Web: http://www-306.ibm.com/software/tivoli/products/netview/, 2005.

[46] IPv6 Forum Korea. Retrieved on October 13, 2005 from the World Wide Web: http://www.ipv6.or.kr/, 2005.

[47] D. Johnson, C. Perkins, and J. Arkko. Mobility Support in IPv6. Retrieved on December 31, 2010 from the World Wide Web: http://www.ietf.org/rfc/rfc3775.txt, June 2004. RFC 3775.

[48] Ka0ticSH. Diggin em walls (part 3) - advanced/other techniques for bypassing firewalls. New Order, April 11, 2002. http://neworder.box.sk/newsread.php?newsid=3957.

[49] Dan Kaminsky. MD5 to be considered harmful someday. http://www.doxpara.com/md5_someday.pdf, December 6, 2004.

[50] Stefan Katzenbeisser and Fabien A.P. Petitcolas. *Information Hiding: Techniques for Steganography and Digital Watermarking*. Artech House, Norwood, MA, 2000.

[51] S. Kent. IP Authentication Header. Retrieved on December 31, 2010 from the World Wide Web: http://www.ietf.org/rfc/rfc4302.txt, December 2005. RFC 4302.

[52] S. Kent. IP Encapsulating Security Payload (ESP). Retrieved on December 31, 2010 from the World Wide Web: http://www.ietf.org/rfc/rfc4303.txt, December 2005. RFC 4303.

[53] S. Kent and K. Seo. Security architecture for the Internet Protocol. Retrieved on December 31, 2010 from the World Wide Web: http://www.ietf.org/rfc/rfc4301.txt, December 2005. RFC 4301.

[54] R. Koodli. Mobile IPv6 Fast Handovers. Retrieved on December 31, 2010 from the World Wide Web: http://www.ietf.org/rfc/rfc5568.txt, July 2009. RFC 5568.

[55] Butler W. Lampson. A note on the confinement problem. *Communications of the ACM*, 16(10):613–615, 1973.

[56] Norka Lucena. *Application-level Protocol Steganography*. PhD thesis, Syracuse University, December 2008.

[57] Norka B. Lucena, James Pease, Payman Yadollahpour, and Steve J. Chapin. Syntax and semantics-preserving application-layer protocol steganography. Lecture Notes in Computer Science, pages 164–179, Toronto, Canada, May 23-25, 2004. Springer-Verlag Heidelberg.

[58] Luxembourg IPv6 Summit 2005. Retrieved on April 6, 2006 from the World Wide Web: http://wiki.uni.lu/ipv6/Luxembourg+IPv6+Summit+2005.html, 2005.

[59] C. Madson and R. Glenn. The use of HMAC-MD5-96 within ESP and AH. Retrieved on December 31, 2010 from the World Wide Web: http://www.ietf.org/rfc/rfc2403.txt, November 1998. RFC 2403.

[60] C. Madson and R. Glenn. The use of HMAC-SHA-1-96 within ESP and AH. Retrieved on December 31, 2010 from the World Wide Web: http://www.ietf.org/rfc/rfc2404.txt, November 1998. RFC 2403.

[61] G. Robert Malan, David Watson, Farnam Jahanian, and Paul Howell. Transport and application protocol scrubbing. In *Proceedings of the IEEE INFOCOM 2002 Conference*, pages 1381–1390, Tel-Aviv, Israel, March 26-30, 2000.

[62] V. Manral. Cryptographic algorithm implementation requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH). Retrieved on December 31, 2010 from the World Wide Web: http://www.ietf.org/rfc/rfc4835.txt, April 2007. RFC 4835.

[63] Wojciech Mazurczyk and Krzysztof Szczypiorski. Steganography in handling oversized ip packets. In *Proceedings of 2009 International Conference on Multimedia Information Networking and Security*, pages 559–564, Wuhan, China, November 18-20, 2009.

[64] Steven J. Murdoch and Stephen Lewis. Embedding covert channels into TCP/IP. In *Information Hiding: Proceedings of the Seventh International Workshop*, Barcelona, Spain, June 06-08, 2005. Springer.

[65] T. Narten, E. Nordmark, W. Simpson, and H. Soliman. Neighbor Discovery for IP version 6 (IPv6). Retrieved on December 31, 2010 from the World Wide Web: http://www.ietf.org/rfc/rfc4861.txt, September 2007. RFC 4861.

[66] Binh Nguyen. SendIP. Retrieved on July 7, 2010 from the World Wide Web: http://freshmeat.net/projects/sendip/, 2003.

[67] K. Nichols, S. Blake, F. Baker, and D. Black. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. Retrieved on December 31, 2010 from the World Wide Web: http://www.ietf.org/rfc/rfc2474.txt, December 1998. RFC 2474.

[68] Phish ¡phish@mindless.com¿. sneaky-sneaky. Retrieved on July 8, 2010 from the World Wide Web: http://www.packetstormsecurity.org/UNIX/penetration/rootkits/index8.html, 2002.

[69] Press Trust of India. TRAI wants govt to kickstart shift to ipv6 through e-gov. http://www.hindustantimes.com/news/181_1578124,00020020.htm, December 20, 2005.

[70] J. Rajahalme, A. Conta, B. Carpenter, and S. Deering. IPv6 Flow Label Specification. Retrieved on December 31, 2010 from the World Wide Web: http://www.ietf.org/rfc/rfc3697.txt, March 2004. RFC 3697.

[71] K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. Retrieved on December 31, 2010 from the World Wide Web: http://www.ietf.org/rfc/rfc3168.txt, September 2001. RFC 3168.

[72] Gerhard Rieger ¡rieger@dest unreach.org¿. socat. Retrieved on July 7, 2010 from the World Wide Web: http://www.dest-unreach.org/socat/, 2010.

[73] Craig H. Rowland. Covert channels in the TCP/IP protocol suite. Psionics Technologies, November 14, 1996. http://www.firstmonday.dk/issues/issue2_5/rowland/.

[74] Joanna Rutkowska. The implementation of passive covert channels in the linux kernel. In 21^{st} *Chaos Communication Congress, Berliner Congress Center*, Berlin, Germany, December 27-29, 2004. www.ccc.de/congress/2004/fahrplan/ files/223-passive-covert-channels-linux.pdf.

[75] Salvatore Sanfilippo. hping. Retrieved on July 7, 2010 from the World Wide Web: http://www.hping.org/, 2006.

[76] Sergio D. Servetto and Martin Vetterli. Codes for the fold-sum channel. In *Proceedings of the* 35^{35} *Annual Conference on Information Science and Systems (CISS)*, Baltimore, MD, USA, March 2001.

[77] Sergio D. Servetto and Martin Vetterli. Communication using phantoms: Covert channels in the internet. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, Washington, DC, USA, June 2001.

[78] Umesh Shankar and Vern Paxson. Active mapping: Resisting NIDS evasion without altering traffic. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, pages 44–61, Washington, DC, USA, May 11-14, 2003. IEEE Computer Society.

[79] Gustavus J. Simmons. The prisoners' problem and the subliminal channel. In David Chaum, editor, *Advances in Cryptology, Proceedings of CRYPTO '83*, pages 51–67. Plenum Press, 1984.

[80] Ed Skoudis. *Counter Hack: A Step-by-Step Guide to Computer Attacks and Effective Defenses*. Series in Computer networking and Distributed Systems. Prentice Hall, Upper Saddle River, NJ, 2002.

[81] Lucie Smith and Ian Lipner. Free pool of IPv4 address space depleted. Retrieved on March 01, 2011 from the World Wide Web: http://www.nro.net/news/ipv4-free-pool-depleted, February 2011.

[82] Taeshik Sohn, Jung-Taek Seo, and Jongsub Moon. A study on the covert channel detection of tcp/ip header using support vector machine. In *Proceedings of the 5th International Conference on Information and Com-*

munications Security (ICICS) 2003, pages 313–324, Huhehaote, China, October 10-13, 2003.

[83] Daniel Stodle. ptunnel - Ping Tunnel. Retrieved on July 08, 2010 from the World Wide Web: http://www.cs.uit.no/ daniels/PingTunnel/, 2009.

[84] Sun. Solstice site manager. Retrieved on November 17, 2005 from the World Wide Web: http://www.sun.com/software/solstice/sm/index.xml, 2005.

[85] Krzysztof Szczypiorski. Hiccups: Hidden communication system for corrupted networks. In *Proceedings of the Tenth International Multi-Conference on Advanced Computer Systems ACS'2003*, pages 31–40, Miedzyzdroje, Poland, October 22-24, 2003.

[86] The IPv6 Portal. Retrieved on April 14, 2006 from the World Wide Web: http://www.ist-ipv6.org/, 2005.

[87] Chau-Ren Tsai, Virgil D. Gligor, and C. S. Chandersekaran. A formal method for the identification of covert storage channels in source code. *IEEE Transactions on Software Engineering*, 16(6):569–580, 1990.

[88] United States IPv6 Summit. Retrieved on November 05, 2005 from the World Wide Web: www.usipv6.com/, 2005.

[89] Verizon. Verizon Begins Testing IPv6 on FiOS Services. Retrieved on May 3, 2010 from the World Wide Web:

http://money.cnn.com/news/newsfeeds/articles/prnewswire/NY81426.htm/, April 2010.

[90] R. Vida and L. Costa. Multicast Listener Discovery Version 2 (MLDv2) for IPv6. Retrieved on December 31, 2010 from the World Wide Web: http://www.ietf.org/rfc/rfc3810.txt, June 2004. RFC 3810.

[91] Xiaoyun Wang, Dengguo Feng, Xuejia Lai, and Hongbo Yu. Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMID. http://eprint.iacr.org/2004/199/, August 17, 2004.

Appendices

Appendix A

Specification-based Traffic Normalization

Flow Label

A specification-based traffic normalization cannot normalize the *Flow Label* field since its value is effectively random. Thus, a traffic normalizer can only reset the field value disabling the functionality entirely. An MRF-based warden can overwrite the field with another value, making sure that the mapping is consistent within a flow. In consequence the available covert channel bandwidth is reduced to 1 bpp as Alice and Bob can only communicate via label changes and not label values.

Source Address Channel

Against a blind adversary that inserts data directly into the *Source Address* field, the probability of discarding a modified packet $P_{Interception}$ equals 7/8. That is because the only type of addresses that a warden allows in *Source Address* field are global unicast addresses and according to the current IANA allocation [43], they have to belong to 2000::/3 IPv6 prefix. Since the prefix contains 1/8 of the total IPv6 address space, the probability of interception is:

$$P_{Interception} = 1 - \frac{1}{8} \tag{A.1}$$

Adversaries aware of active warden's presence can adjust their strategy and ensure that all data injected into *Source Address* field falls within the required prefix. This strategy would lower the channel bandwidth from 32 bits per packet to 29 bits per packet.

MTU Channel (ICMP.27)

If a blind adversary inserts data directly into the *MTU* field, the injections that result in *MTU* value of less than 1280 will be overwritten. This means that 1280 values out of possible 4294967296 will be affected, resulting in discarding of 0.000000298 of modified packets assuming uniform distribution of inserted data.

$$P_{Interception} = \frac{1024}{4294967296} \tag{A.2}$$

Attackers that want to avoid even such a low probability of detection, can simply avoid the low 1280 values.

Pointer Channel (ICMP.29)

Parameter Problem messages are sent to indicate a problem with a received IPv6 packet. Since they are used to report IP-layer problems, there are only few legal *Pointer* values. The exact number depends on the packet included in the message payload, for example in a IPv6 packet containing only IPv6 header and an upper-layer payload, the only feasible *Pointer* values are 1, resulting from a *Flow Label* problem, and 6, indicating a problem with the *Next Header* field. In this scenario, virtually all packets modified by the attacker will be discarded.

Appendix B

Network-based Traffic Normalization

Traffic Class

A covert channel attacker can manipulate the *Traffic Class* field to send covert messages. A specification-based warden can only zero the field preventing the attack but at the cost of disabling the traffic class functionality. A network-aware active warden can enforce the traffic class values coherent with the current values valid within its network thus reducing the available covert bandwidth.

The bandwidth available when a network-aware active warden is present is

$$C_{Bandwidth} = log_2(t!) \tag{B.1}$$

where t is the number of currently valid traffic classes, assuming that both Alice and Bob are aware of the current traffic class values and were able to negotiate assigned meanings. If Bob does not possess this information, Alice can only communicate at 1 bpp via changing the field value.

If the network defines multiple equivalent traffic classes, the warden can randomly reassign a packet's traffic class value to an equivalent class, and further disrupt the communication to 0 bpp.

Flow Label

Improving the defenses presented in Appendix A, a network-aware active warden can perform additional channel mitigation. If the warden is aware of current valid flow label values, it can ensure that observed value within a flow "sticks" to the flow, i.e. the flow label value does not fluctuate within any given network flow. Since flow label values can expire, the new value should not appear until the flow stops and the current value expires. The minimal expiration timeout is specified at 120 seconds, giving Alice and Bob the bandwidth of 1 bit per 2 minutes.

A clever covert channel attacker might attempt to maintain 1 bpp bandwidth by manipulating not only the *Flow Label* value but also all other parameters that identify the packet as belonging to a flow (e.g. *Source Ad-*

dress, *Destination Address*, etc) effectively reassigning the packet from one flow to another. While possible, this approach presents Bob with a significantly harder problem while receiving the covert communication.

First, the more thorough modification decreases the semantics-preservation level of the covert communication from *final destination* to *destination* thus forcing Bob to perform traffic restoration or causing overt traffic disruption. In some cases, for example when IPsec is present, this makes the communication impossible at all. Second, since the modified packet will be routed differently than other packets of its flow, Bob's positioning has to be much more precise allowing him to intercept both flows. Thirdly, Alice and Bob require an additional communication channel allowing Bob to distinguish between legitimate packets of the second flow and the reassigned packets from the first flow.

Source Address

In addition to effects described in Appendix A, a network-aware active warden can further lower the channel bandwidth depending on a number of nodes present on the network. An attacker using an unguarded channel can achieve 32 bits per packet bandwidth, the presence of a traffic-normalizing warden lowers the bandwidth to 29 bits per packet. In order to defeat a network-aware warden, attackers have to communicate using addresses of actual network nodes. The amount of information that a choice of an ad-

dress can convey depends on the number of available nodes n. If $C_{Bandwidth}$ is the channel bandwidth measured in bits per packet, then

$$C_{Bandwidth} = log_2(n) \tag{B.2}$$

For example, for an active warden guarding a network consisting of 256 nodes, the channel bandwidth would be reduced from 32 (or 29) bits per packet to 8 bits per packet.

Routing Header Addresses

A warden-aware adversary that attempts to circumvent the actions taken by an active warden has an alternative to manipulate the order of legitimate router addresses in the Routing Header.

Let $C_{Bandwidth}$ be the channel bandwidth measured in bits per packet, n be the number of addresses present in a Routing Header. The bandwidth of a Routing Header covert channel based on the order of the contained addresses is given by the equation,

$$C_{Bandwidth} = 128 \cdot n \tag{B.3}$$

considering that each address has a length of 16 octets (128 bits).

However, if the attacker is forced to use only real router addresses, such bandwidth also depends on the number of routers, r, within the protected

system. That is,

$$C_{Bandwidth} = log_2(r^n) = n \cdot log_2(r) \tag{B.4}$$

The ratio between B.3 and B.4,

$$\frac{128}{log_2(r)} \tag{B.5}$$

represents bandwidth loss the adversary will suffer when her actions are limited by the active warden.

The warden can further reduce the bandwidth if she assumes that any given router address can only appear once in the Routing Header address list. In this case, the available covert bandwidth is further reduced to

$$C_{Bandwidth} = log_2(n!) \tag{B.6}$$

where the maximum value of n is r.

Home Agent Address Discovery Reply Addresses

Similarly as in the case of the addresses included in the Type 0 Routing Header, the bandwidth available in an unguarded network is

$$C_{Bandwidth} = 128 \cdot n \tag{B.7}$$

where n is the number of inserted fake home agent addresses.

In the presence of a network-aware active warden, the attacker is forced to use only addresses of the legitimate home agent nodes. Since the ordering of the addresses in the protocol message does not matter, the entropy present is

$$C_{Bandwidth} = log_2(h!) \tag{B.8}$$

The additional measure that the warden can take is to take advantage of the fact that the ordering is arbitrary and rearrange the addresses thus further reducing potential covert communication bandwidth to 0.

Appendix C

Protocol Field Entropy Reduction

field	**countermeasure/knowledge**	**entropy/bandwidth**
Traffic Class	—	8 bpp
Traffic Class	all current traffic classes	$log_2(T!)$ bpp
Traffic Class	multiple equivalent traffic classes	0 bpp
Flow Label	—	20 bpp
Flow Label	current flow parameters	1 bit per 2 mins
Hop Limit	—	< 8 bpp
Hop Limit	network node map	0 bpp
Source Address	—	128 bpp

field	**countermeasure/knowledge**	**entropy/bandwidth**
Source Address	all node addresses	$log_2(S)$ bpp
Destination Address	—	128 bpp
Destination Address	all node addresses	$log_2(D)$ bpp
RH Addresses	—	128 bits per included address
RH Addresses	all router nodes	$log_2(R!)$ bpp
AH/ESP SPI	—	32 bpp
AH/ESP SPI	IPsec SAs, sequences and algorithms	$< log_2(SA)$ bpp
AH/ESP SPI	full IPsec disclosure	0 bpp
AH/ESP Sequence	—	32 bpp
AH/ESP Sequence	IPsec SAs, sequences and algorithms	variable, depends on current sequence no.
AH/ESP Sequence	full IPsec disclosure	0 bpp
AH ICV	—	variable
AH ICV	IPsec SAs, sequences and algorithms	0 bpp
ESP ICV	—	variable
ESP ICV	full IPsec disclosure	0 bpp
ESP Payload	—	variable

field	**countermeasure/knowledge**	**entropy/bandwidth**
ESP Payload	full IPsec disclosure	0 bpp
MTU	—	32 bpp
MTU	all links' MTUs	0 bpp
Pointer	—	32 bpp
Pointer	destination node capabilities	variable, depends on data
HAADR Addresses	—	128 bits per included address
HAADR Addresses	all home agents	$log_2(H!)$ bpp
HAADR Addresses	randomized-order list	0 bpp

Table C.1: Effects of network knowledge on available protocol field entropy/bandwidth. T - number of current traffic classes, S - number of eligible source nodes, D - number of eligible destination nodes, R - number of eligible router nodes, H - number of home agent nodes.

Table C.1 describes the effects of information about the network on the entropy available in IPv6 protocol fields. The table lists only IPv6 protocol fields whose normalization is performed by the network-aware active wardens (i.e. *Reserved* fields and other similar fields that can be normalized using a specification-based warden are not included in the table). For each listed field, the first entry describes the baseline scenario when the network

is not guarded by any active warden. The subsequent entries enumerate increasing levels of network knowledge and the resulting drop in the field's entropy and available covert channel bandwidth. The presented scenarios assume that Bob is positioned to perform traffic restoration.

Appendix D

cctool Examples

Off-line traffic test

The off-line test is an equivalent of live traffic tests, but instead of capturing and inserting actual traffic, it operates on `libpcap` capture files. In this scenario, the previous test is repeated, but without live network modification.

Sender:

```
# instructs tcpdump to capture traffic on interface eth0
# and save it in traffic.cap
tcpdump -i eth0 -s 0 -w traffic.cap


curl http://www.example.org/
```

Alice:

```
# instructs cctool to attach to process the capture
# file, embedding the message using the traffic class
# (TC) covert channel
./cctool -cPCAP -aTC -Mtraffic.cap -Tcovert.cap \
-mTest0123 -oSEND
```

Bob:

```
# instructs cctool to process the covert traffic file
# and extract messages using the traffic class (TC)
# covert channel
./cctool -cPCAP -aTC -Mcovert.cap -Tfinal.cap -oRECV
```

Vita

Grzegorz Lewandowski was born in Szczecin, Poland, on August 13, 1973, the son of Irena Lewandowska and Henryk Lewandowski. After graduating from Adam Asnyk High School in Szczecin, in 1992, he entered the Technical University of Szczecin, pursuing a double major of Electrical Engineering and Computer Science. In 1994, he transferred to Franco-Polish School of New Information and Communication Technologies (EFP) in Poznan. During his studies at the EFP, he joined the Northeast Parallel Architectures Center (NPAC), Syracuse, as a visiting scholar. He entered The Graduate School at Syracuse University in January 1997, receiving a Master of Science degree in Computer Science in December 1997. He continued his work at NPAC and later at Syracuse University's Center for Advanced Systems and Engineering (CASE). Gregg published papers in protocol steganography, building automation and web application security and received his Ph.D. in Computer Engineering from Syracuse University in 2011. He also co-founded CollabWorx, a technology company providing real-time collaboration software where he holds position of Chief Software Engineer.