Network Forensic Readiness: A Bottom-up Approach for IPv6 Networks

Roman Ammann

A thesis submitted to Auckland University of Technology
in partial fulfilment of the requirements for the degree of
Masters of Forensic Information Technology

2012

School of Computing and Mathematical Sciences

Primary Supervisor: Associate Professor Nurul I. Sarkar

Declaration

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.

. . . .
Roman Ammann

Acknowledgment

This research was completed at the School of Computing and Mathematical Sciences in Faculty of Design and Creative Technologies of the AUT University in New Zealand. While completing this research project I received support from many people. I would like to thank all those who supported me, those who challenged me or gave me feedback and those who just endured listening to me complaining about the research. I highly appreciate your support.

Further, I would like to thank Associate Professor Nurul I. Sarkar, my primary supervisor, and the network research group at AUT for their input at all stages of the thesis and constant critical questioning of the form and content.

I would also like to express my gratitude to Diana Kassabova for proof reading my thesis and indicating some vague sections. Her work helped to improve the readability of the thesis immensely.

I would also like to acknowledge my fellow student Wai Loong Tham for reading several chapters of my thesis and providing me valuable feedback.

Also, a big thank to Appserv Limited, my employer, for lending me hardware to run the virtualised network lab. The uncomplicated and hassle-free access to their server definitely simplified the task of setting up and running the network lab.

Finally, I would like to thank my family, my girlfriend and my friends who accepted my staying away from events and get-togethers and gave me time and room to complete this thesis.

Abstract

A computer network is considered forensically ready, when crucial evidence for a forensic investigation is proactively collected and easily available. While the benefits of a forensically ready network are well understood, the exact information required to be collected to achieve forensic readiness is largely unknown. This thesis focuses on identifying and locating the information that is essential for successful forensic investigations in an IPv6 network. Without the knowledge of what information should be retained, the approach to achieving forensic readiness is likely to be unstructured and crucial information for an investigation might be missed.

This study conducted an empirical investigation to identify and extract forensic information from network protocol standards and related literature. Malicious and genuine network scenarios were run and retraced in a test bed to elicit the information that is significant for a forensic investigation. The network scenarios were grouped by network layer and the layers were processed bottom-up to resolve dependencies of the higher layers on the lower layers. A subset of network scenarios was exclusively used to ascertain the effectiveness of the identified information (hold-out approach).

This thesis identifies the information in an IPv6 network that is relevant for a successful forensic investigation. Further, the thesis also proposes an optimisation phase as an extension of the National Institute of Standards and Technology (NIST) forensic life-cycle. This phase allows to improve the forensic readiness further through the identification of missing information after conducting a forensic investigation in the network. Finally, design and deployment strategies for implementing a forensically ready network are outlined and recommendations are made for mastering key issues related to forensic readiness.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AAAA	DNS resource record type used to store a single IPv6 address
ACK	Acknowledgement
AH	Authentication Header
ARP	Address Resolution Protocol
AXFR	DNS opcode mnemonic for a zone transfer
BPDU	Bridge Protocol Data Unit
CNAME	DNS resource record type used to specify an alias for a domain name
CVE	Common Vulnerabilities and Exposures
DAD	Duplicated Address Detection
DHCP	Dynamic Host Configuration Protocol (v4: version 4, v6: version 6)
DNS	Domain Name System
DoS	Denial of Service
DUID	DHCP Unique Identifier
FIN	Finish (often refers to TCP segments with the finish flag set)
GRE	Generic Routing Encapsulation
HTTP	Hypertext Transfer Protocol
IAID	Identity Association ID
ICMP	Internet Control Message Protocol (v4: version 4, v6: version 6)
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol (v4: version 4, v6 version 6)
IPSec	Internet Protocol Security
LSA	Link-State Advertisement

MAC	Media Access Control
MTU	Maximum Transfer Unit
NA	Neighbour Advertisement
ND	Neighbour Discovery
NFAT	Network Forensic Analysis Tools
NIST	National Institute of Standards and Technology
NS	Neighbour Solicitation
NUD	Neighbour Unreachability Detection
OSPF	Open Shortest Path First
PSH	Push (often refers to TCP segments with the push flag set)
RA	Router Advertisement
RFC	Request for Comments
RIP	Routing Information Protocol
RIPng	Routing Information Protocol Next Generation
RPF	Reverse Path Forwarding
RR	Resource Record
RS	Router Solicitation
RST	Reset (often refers to TCP segments with the reset flag set)
RSTP	Rapid Spanning Tree Protocol
SLAAC	Stateless Address Auto-Configuration
SOA	Start of Authority
SYN	Synchronise (often refers to TCP segments with the synchronise flag set)
TCN	Topology Change Notification
TCP	Transport Control Protocol
UDP	User Datagram Protocol
VLAN	Virtual Local Area Network

Glossary

Access Port A bridge or switch port that is connected to a host.

Access Link A network connection between the host and the bride or switch it is connected to.

Access Network A network used to connect hosts to the network.

Bridge/Switch A device that forwards Ethernet frames based on the destination MAC address. The two terms are used interchangeable.

Datagram A User Datagram Protocol Data Unit.

Frame An Ethernet Protocol Data Unit.

Host Any node that is not a router or a bridge.

ICMP Internet Control Message Protocol version 6. The terms ICMPv4 and ICMPv6 are only used when it is necessary to avoid ambiguity.

Interface A node's attachment to a link.

IP Internet Protocol version 6. The terms IPv4 and IPv6 are only used when it is necessary to avoid ambiguity.

Link A communication facility or medium over which nodes can communicate.

Neighbours Nodes attached to the same link.

Node A device that implements IPv6.

Off-Link A node is off-link if it is not a neighbour.

On-Link A node is on-link if it is a neighbour.

Packet An IP Protocol Data Unit.

Router A device that forwards IPv6 packets not explicitly addressed to itself.

Segment A Transport Control Protocol Data Unit.

Notation

Forensically relevant information and network scenarios are referenced by IDs (e.g. I.DB.03 or S.NM.07). The IDs are in the format:

<Kind>.<Layer><Protocol/Intent>.<Number>

The following abbreviations are used in the information IDs:

Kind	Layer	Protocol	Number
I → Information	**D** → Data-Link	**E** → Ethernet	incrementing
		B → Bridge	
		V → VLAN	
	N → Network	**I** → IPv6	
		C → ICMPv6	
		6 → 6in4	
		G → GRE	
		R → RIPng	
		O → OSPF	
	T → Transport	**T** → TCP	
		U → UDP	
	A → Application	**H** → DHCPv6	
		N → DNS	

The following abbreviations are used in the scenario IDs:

Kind	Layer	Intent	Number
S → Scenario	see above	**G** → Genuine	incrementing
		M → Malicious	

For example, I.DB.03 stands for the information number 03, related to bridging on the data-link layer and S.NM.07 refers to the malicious network scenario number 07 on the network layer.

Chapter 1

Introduction

1.1 Background and Motivation

Computer networks form the backbone for services such as email, Facebook, Skype and Google. These services are so common today that almost nobody wants to do without them (90.7 % of Internet users in Australia rate it as "very important" or "important" [1]). One could even say that many users depend on them heavily. This makes it very important to operate these services in a robust and secure manner.

The first steps in securing a service are to understand the threats the service is exposed to and to understand who and why somebody would attack the service. All this leads to the conclusion that in order to make a service secure it is necessary to lower the incentive to attack the service, making it unattractive for an attacker.

The common way to lower the incentive for attacking a service is to make an attack as difficult as possible. This is done by implementing defence measures and by making the service more robust. However, the better and more complete the defence is, the more complicated and resource-intensive it is to operate. At some point, it is just not economical anymore to further improve the defence because it would be cheaper to repair the damage of a successful attack than to implement an additional protective measure.

Another way to lower the incentive for attacking a service is to increase the likelihood of severe consequences for the attacker. Consequences in this context usually mean legal prosecution and penalties.

Network forensics is concerned with the analysis of security incidents and, at best, determining the source of an attack or even the identity of the attacker. Yet, shortly after an incident has occurred the focus is not on collecting forensically sound evidence but on restoring the service as quickly as possible.

Forensic readiness is the concept of proactively collecting forensically sound evidence. In the case of an incident, the focus would still be on restoring services and the incident could later be properly analysed with the evidence already collected before and during the incident. The evidence is collected in a forensically sound manner and, therefore, the chances for a successful prosecution of the attacker is higher and the cost of an incident analysis is lower [2].

1.2 Objective of this Thesis

Archiving network forensic readiness involves implementing proactive evidence collection in a forensically sound manner for all network activities. This thesis identifies forensic information in an IPv6 network that is to be retained to achieve network forensic readiness. Data-link layer protocols up to application layer protocols are analysed and network scenarios on these network layers are run and retraced to identify the relevant forensic information.

Network or security practitioners implementing means to proactively collect the identified information prepare their network for forensic investigations and, consequently, achieve network forensic readiness. Without such a recommended set of information to retain, an approach to achieve forensic readiness is likely to be unstructured and crucial information for an investigation might be missed.

1.3 Structure of this Thesis

The overall structure of this thesis is shown in Figure 1.1. It is divided into three parts.

Chapters 2 to 4 introduce the topic and provide the necessary background knowledge. Chapter 2 discusses network forensic readiness. It reviews current literature to establish a sound understanding of the concepts and the current state of the field. Chapter 3 reviews the network protocols covered in this thesis. It identifies the relevant protocol standards and examines known attacks against them. Chapter 4 describes and justifies the research methodology and outlines the expected outcomes.

Chapters 5 to 8 are devoted to the experimental phase of the thesis. Each chapter covers one network layer, presents the relevant network scenarios and discusses some selected scenarios in detail. Further, each of this chapters

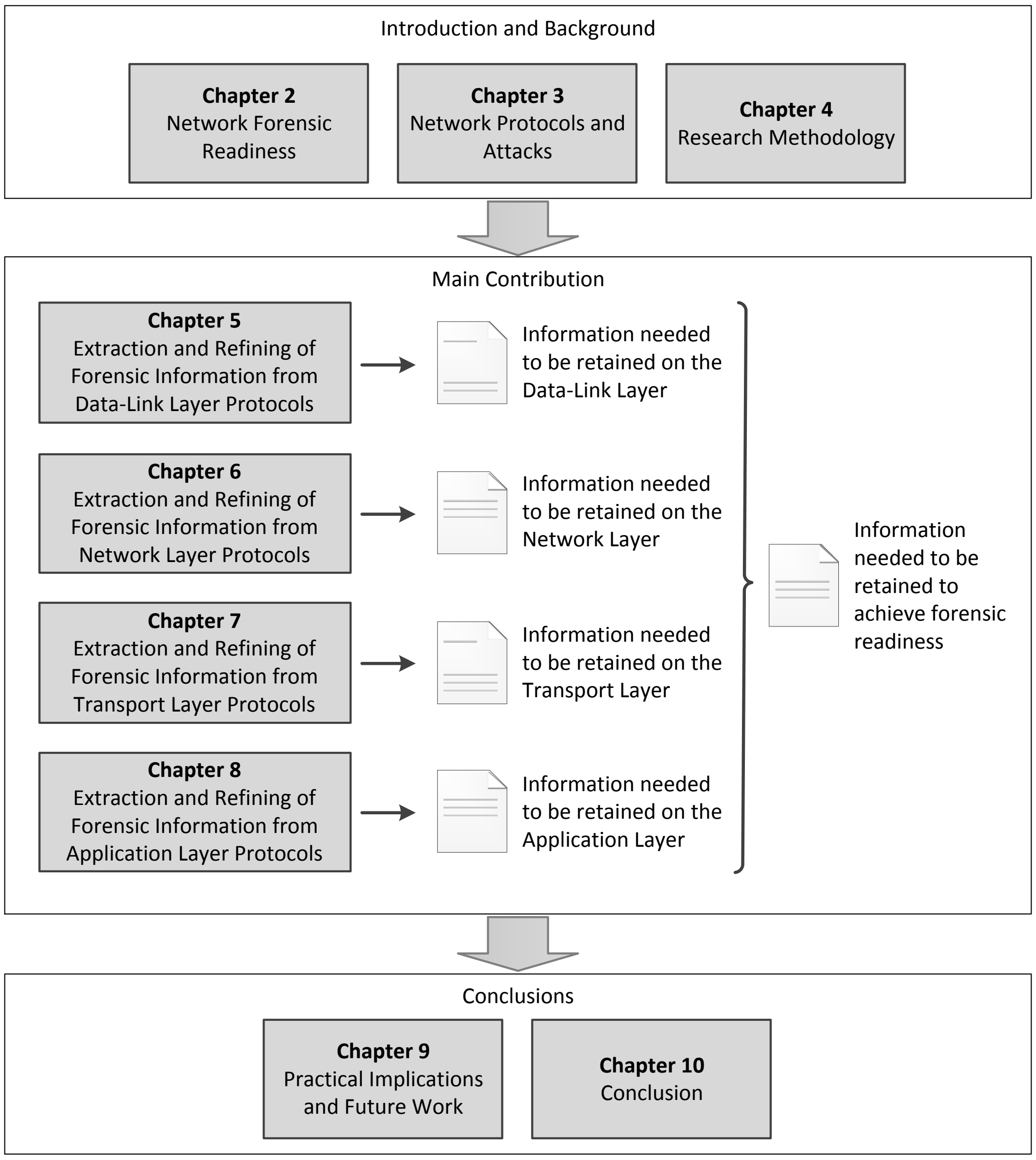
Introduction and Background
Chapter 2
Network Forensic Readiness
Chapter 3
Network Protocols and Attacks
Chapter 4
Research Methodology
Main Contribution
Chapter 5
Extraction and Refining of Forensic Information from Data-Link Layer Protocols
Information needed to be retained on the Data-Link Layer
Chapter 6
Extraction and Refining of Forensic Information from Network Layer Protocols
Information needed to be retained on the Network Layer
Chapter 7
Extraction and Refining of Forensic Information from Transport Layer Protocols
Information needed to be retained on the Transport Layer
Chapter 8
Extraction and Refining of Forensic Information from Application Layer Protocols
Information needed to be retained on the Application Layer
Information needed to be retained to achieve forensic readiness
Conclusions
Chapter 9
Practical Implications and Future Work
Chapter 10
Conclusion

Figure 1.1: The structure of this thesis

presents the information that is recommended to be retained for the particular network layer.

Chapter 9 discusses the results and Chapter 10 concludes the thesis. Chapter 9 presents the major findings from Chapters 5 to 8 and the original contributions of this thesis. In addition, it discusses the limitations of the research and its practical application. The chapter closes with elaborating on a number of possible further research directions. Chapter 10 summarises and concludes the thesis.

Chapter 2

Network Forensic Readiness

2.1 Introduction

Chapter 1 outlined the background and the motivation for this research. This chapter reviews contemporary literature regarding network forensic readiness with the aim to acquire a sound understanding of the current state of the research field and the current challenges. This is important because this thesis aims to extend the field with a set of information that should be retained in order to achieve forensic readiness.

Section 2.2 discusses network forensics and its definition. Section 2.3 presents the concept of network forensic readiness and the arguments for why it is a worthwhile objective. Section 2.4 is based on the first two sections and introduces the concept of network forensic readiness. The section reviews a framework for forensic readiness that approaches the objective from a management point of view. Section 2.5 summarises the relevant literature about IPv6 forensics and introduces ideas borrowed from IPv6 security related literature. Section 2.6 analyses and discusses the problem concerning the amount of potential forensic evidence and its possible solutions.

2.2 Network Forensics

"Network forensics is the act of capturing, recording, and analyzing network audit trails in order to discover the source of security breaches or other information assurance problems" [3, p. 2]. Security expert Marcus Ranum gets the credit for minting the term network forensics [4].

Garfinkel [5] identifies two main concepts related to capturing events relevant for network forensics. The Catch-it-as-you-can approach tries to retain all information from all sources and analyses subsets of it later, while the Stop-look-and-listen approach tries to analyse the events in memory and retains just the relevant information.

The definition of network forensics and what elements are part of it varies from source to source. Corey et al. [6] define network forensics as capturing all network traffic. Almulhem and Traoré [4] add recording network events to the definition without going into details about what these events exactly are. Ren and Jin [7] even go further and argue that computers are part of the network and, therefore, define computer forensics as a subset of network forensics.

This research adapts the definition of Ren and Jin [7] and includes end systems in the scope because they are the reason for the existence of computer networks and are a vital part of them. They also hold a wealth of information crucial to achieving forensic readiness and should, therefore, not be excluded.

2.3 Forensic Readiness

According to Tan, forensic readiness has two objectives: "Maximizing an environment's ability to collect credible digital evidence, and; Minimizing the cost of forensics in an incident response." [8, p. 1]. He urges a proactive approach to evidence collection and gives a broad overview of the elements of forensic readiness. An important point he makes is that multi-tier logging adds credibility to the collected evidence. Multi-tier logging is collecting the same information from different sources. For example, a client requests a website from a web server. One source of information about this activity is the web server's log. Another source is the network traffic that contains the client's request and the server's reply. Having information from both sources makes the evidence more credible.

Rowlingson [9] proposes a ten-step process to achieve forensic readiness throughout an organisation. He suggests a top-down approach and starts by defining business scenarios that require digital evidence before identifying potential sources. He motivates this research by stating "relevant evidence is unlikely to exist by default" [9, p. 3]. However, he does not go down to the protocol level details with his approach and, therefore, does not specify which information exactly should be retained to achieve forensic readiness.

2.4 Network Forensic Readiness

Network forensics and the concept of forensic readiness are relatively new areas of research in network security. Combining the two gives a promising edge for tackling current network forensic challenges.

Endicott-Popovsky and Frincke [10] urge the implementation of forensic ready networks to reduce the time and money necessary to successfully prosecute attackers. They created a framework for implementing network forensic readiness based on the strategic goal of a company to be able to hold intruders accountable. Endicott-Popovsky et al. [2] argue that network forensic readiness is the only way to break the escalation cycle of the arms race between attackers and victims. Forensic readiness enables a company to collect sound forensic evidence which is a key element in successfully prosecuting an attacker. Both papers advocate the need for forensic readiness. They approach network forensic readiness by setting the strategic goal of having the ability to pursue legal action against intruders. However, they do not elaborate on the implementation phase of a forensically ready network, neither do they recommend concrete mechanisms for achieving forensic readiness.

2.5 IPv6 Forensics

Nikkel [11] identifies the need for resolving incidents involving IPv6 networks. He points out the major changes in IPv6 compared to IPv4 (address format, DNS entries), shows newly available helper protocols (neighbour discovery, router advertisement) and mentions publicly available databases and how they are useful in an investigation involving an IPv6 network. Nevertheless, his approach relies on available information and not on proactively collected evidence. Also, he does not elaborate on the protocol level details necessary to retrace incidents in an IPv6 network.

The field of IPv6 security is better documented and gives some pointers towards IPv6 forensics. Hogg and Vyncke [12] provide a good introduction to IPv6 security with some minor pointers towards forensics. The NIST standard SP800-119 [13] is a guideline for a secure IPv6 deployment. It also gives a few recommendations regarding forensics (e.g. preference for DHCPv6 over Stateless Address Auto-Configuration (SLAAC) and disabling the privacy extension [14] for a better audit trail). Further, known attacks against the IPv6 protocol [15, 16] allow inferring what kind of situations a forensically ready network needs to be able to handle.

Table 2.1 presents the key researchers in the fields of network forensics, IPv6 forensics and forensic readiness with their main contribution to the respective field.

Table 2.1: Key researchers and their main contributions to IPv6 forensics, network forensics and forensic readiness.

Key Researcher	**Main Contribution**	**Year**	**Key concepts/description**
Casey	Network forensics fundamentals	2011	Provided a good discussion on network forensics, network level evidence and their potential values.
Kaushik, Joshi	Network forensic framework	2010	Developed a framework based on detecting ICMP attacks.
Pilli, Joshi, Niyogi	Network forensic framework	2010-2011	Provided an overview of existing frameworks; implemented a framework based on TCP/IP attacks.
Nikkel	IPv6 network forensics	2007	Provided an overview of IPv6 network forensics; pointed out some key issues and challenges.
Endicott-Popovsky, Frincke, Taylor	Approach to archive forensic readiness	2007	Developed an approach based on the operational goal to be able to prosecute attacker.
Rowlingson	Approach to archive forensic readiness	2004	Developed a structured approach to archive forensic readiness.
Tan	Concept of forensic readiness	2001	Provided a key terms and definitions; technical and legal concerns about evidence collection.

2.6 The Vast Quantity of Forensic Evidence

Evidence in a network comes from different sources. Some of the sources are: intrusion detection system (IDS) and firewall logs, logs generated by network service and applications, traffic captures of packet sniffers and network forensic analysis tools (NFATs) and network artefacts on hard-drive images [17].

The potentially large number of collected network events can be reduced in two ways. Firstly, the number of events can be lowered by only collecting data about events that are crucial for a later investigation. Secondly, the amount of space required to store a single event can be minimised by recording only the representative attributes for each event [18].

2.6.1 Minimising the Number of Events

Almulhem and Traoré [4] propose a network traffic collection system based on three distinct modules. The first module identifies and marks suspicious network traffic. The second one is host-based and captures the marked traffic and sends it to the third module, the central logging module. Almulhem and Traoré concentrate on the capturing mechanism and the advantages of capturing network traffic on the end system but not how the marking module decides which traffic is suspicious. While they use network traffic as the sole source of information to retrace network events, having a capture module on the end system might be a viable approach to achieve network forensic readiness. It allows not just the recording of network traffic but also the logging of events and state tables related to that traffic.

2.6.2 Minimising the Amount of Space per Event

Several approaches exist for minimising the storage capacity needed to store captured network events.

Cooke et al. [19] propose a resource-aware multi-format security data storage. They suggest to record security data at several different granularities in parallel and to summarise them to different abstractions. The summarisation is a trade-off between storage costs and information details. For example, network traffic is captured at a packet level. The packets are summarised into flows (network and transport layer parameters and traffic counters of a connection between two systems). They could also be summarised into events, for example, client A requested website www.google.com from server B. Because the summaries need less storage space, it is possible to keep them for longer. Detailed packet captures with a wealth of information but high storage demands are kept just for a short time while summarised abstractions such as flows or events are kept for longer.

Maier et al. [20] implemented a ring-buffer system that records only the beginning of long communication sessions. They showed that the system is able to capture and retain large volumes of network traffic for several days with a reasonable amount of storage capacity. They also argued that most of the data needed for the event analysis is found at the beginning of traffic sessions. This approach is very promising for high bandwidth links. The placement of such a system is a challenge because centralised collection often lacks visibility. Even if the system is set up to capture all network traffic in the network core, it sees just a subset of all connections [21]. For example, traffic between hosts in the same subnet can never be captured by a system placed in the network core. In addition, it just sees the data-link layer details for the segment to which it is connected. They use a capture-all-sort-later approach that focuses on network traffic. This research, on the other hand, focuses on identifying relevant events and uses network traffic as one potential source.

Kaushik et al. [22] and Pilli et al. [23] analyse ICMP attacks and TCP/IP attacks to extract significant protocol features to identify suspicious network traffic. This traffic is captured and written to a database. There it is analysed based on statistical thresholds to recognise attacks. They use known protocol level attacks to identify significant protocol features. While it is important to be able to detect and investigate attacks, it is also crucial to be able to retrace genuine activities. For instance, even though the traffic up to a certain network layer is genuine, the payload that it transports for the next higher layer might be malicious. In that case, the evidence from the lower layers is helpful to show that there was a connection between the two systems and the actual attack data shows the malignity of the connection. Furthermore, even if it is not an attack, it might be network traffic that is not complying with the network usage policy. In both cases, it might be important for an investigation to prove a connection between two systems or at least the feasibility of such a connection. When focusing on malicious activities, the importance of being able to retrace genuine behaviour in the network might be neglected. This research adopts the approach of using malicious traffic to train and optimise the set of forensically significant information but also extends it by adding genuine traffic to the training data. Also, a wider range of protocols is covered.

2.7 Summary

Table 2.2 shows the main steps to achieving network forensic steps, the challenges for each step and some approaches to handle these challenges. The conducted literature review shows that network forensic readiness is a worthwhile goal to achieve. It also shows that the possible amount of information needed to achieve forensic readiness is overwhelming and that a recommendation on what exact information should be retained to achieve forensic readiness is lacking. Such a recommendation would bring the field a step forward.

The next chapter reviews the relevant standards of the network protocols analysed in this thesis and the known attacks against them.

Table 2.2: Steps to achieving network forensic readiness, possible approaches and challenges.

	Identification	**Locating**	**Collecting**	**Storing**
Description	Identify the information that is relevant for a forensic investigation	Locate where the identified information is stored and can be accessed	Collect the identified information in a forensically sound manner	Store the collected information so that it is easily accessible during forensic investigations
Challenges	Not missing any crucial forensic information	Identifying the most reliable source of a specific piece of information and locating where the information can be easily accessed	The amount of different information that needs to be collected from a variate of network devices stored in diverse formats Implementing reliable and forensically sound means to collect a variate of forensic evidence in different formats and with access methods	The different format of the collected evidence, the large volume of potential evidence
Approach	Identifying business processes that require forensic evidence [9], Identifying assets during risk assessment that would warrant digital forensic protection [2], Identifying relevant forensic information by analysing known network attacks [22, 23, 24], Public available IPv6 information [11]	-	Distributed network traffic collection [4, 25]	Compressing and summarising network traffic over time [19], a ring-buffer system storing only the beginning of communication sessions [20], Compression [25]

Chapter 3

Network Protocols and Attacks

3.1 Introduction

Chapter 2 outlined the issues of network forensic readiness. This chapter introduces relevant network protocol standards and identifies potential attacks against them. The chapter also defines the scope of the thesis and briefly introduces the involved network protocols. It also illustrates the situations that a forensically ready network needs to be able to handle and the kind of network scenarios that need to be retraceable.

Section 3.2 reviews possible types of attacks on the data-link layer against the protocols Gigabit Ethernet, Bridging, Virtual Local Area Network (VLAN) and Rapid Spanning Tree Protocol (RSTP). Section 3.3 reviews the network layer protocols Internet Protocol (IP) version 6, Internet Control Message Protocol (ICMP) version 6, Routing Information Protocol Next Generation (RIPng), Open Shortest Path First (OSPF) version 3, 6in4 and Generic Routing Encapsulation (GRE). Section 3.4 discusses the transport layer protocols User Datagram Protocol (UDP) and Transport Control Protocol (TCP). Section 3.5 reviews the Domain Name System (DNS) protocol and the Dynamic Host Configuration Protocol (DHCP version 6 located on the application layer.

3.2 Data-Link Layer

3.2.1 Gigabit Ethernet

Gigabit Ethernet, standardised in IEEE 802.3ab [26, 27], defines the physical layer for 1000 Mbps Ethernet. It is widely used in switched campus networks.

Attacks against the physical layer are not very common but they are not impossible. Unplugging a network port is essentially a denial of service (DoS) attack. Another DoS attack would be to jam the signal. Signal modification, eavesdropping and signal injection are also possible on the physical layer, even though they are often easier to implement on the data-link layer.

Another form of attack on the physical layer might be simply stealing or destroying a device.

3.2.2 Bridging

Ethernet Media Access Control (MAC) address bridging is defined in IEEE 802.1D [28]. It divides each access link into a single data-link layer collision-domain, learns the MAC addresses in each domain and forwards frames between the domains based on their destination MAC address. The bridge stores the learned MAC addresses in the Filter Database and consults it for forwarding decisions.

The Generic Attribute Registration Protocol and the Generic Attribute Registration Protocol Multicast Registration Protocol, also part of the standard, are ignored in this research because they are not widely used.

The MAC flooding attack tries to overflow the Filter Database of a switch with random source MAC addresses. The goal is that old genuine entries in the Filter Database are replaced with new random entries. This forces the switch to forward newly arrived frames to all ports because it does not have an entry for the destination MAC address of the frame in the Filter Database [29].

MAC address spoofing, also known as port stealing or MAC cloning, tries to take over a MAC address of a victim by sending traffic to the switch with the source MAC address of the victim. When receiving a frame with the victim's source MAC address on another port, the switch updates its Filter Database and from then on forwards traffic that is addressed to the victim to the port of the attacker instead [30].

3.2.3 Rapid Spanning Tree Protocol (RSTP)

RSTP, standardised in IEEE 802.1D [28], defines how to achieve a loop-free network topology on the data-link layer by temporarily disabling specific bridge ports.

The protocol can be attacked in several ways. Root Claim is an attack where an attacker claims to be the root bridge by sending spoofed frames with the lowest possible bridge ID. Being the root bridge can give the attacker an advanced position, for example, for traffic sniffing if the attacker is connected to more than one switch [31]. Furthermore, this can lead to significant performance penalties if, for example, the root bridge is suddenly located in the access layer and the link between the core switches changes its state from forwarding to blocking [32].

Eternal Root Election is a way to keep the switching infrastructure in the state of electing a root bridge. The attacker keeps decrementing the bridge ID in the bogus bridge frames, starting with one lower than the current root bridge. The attack just starts over when the lowest possible bridge ID is reached [32].

Receiving a Topology Change Notifications (TCN) can cause the root bridge to send Bridge Protocol Data Units (BPDUs) with the topology change flag set. Other bridges receiving these BPDUs clear their Filter Database because of the new topology (most implementation just age them out faster by reducing the ageing time). By sending a constant stream of TCNs, an attacker can force bridges in the network to continuously clear their Filter Databases, which can cause loops and keeps the network in an unstable state [32].

3.2.4 Virtual Local Area Network (VLAN)

VLANs, defined in IEEE 802.1Q [33], are used to virtually divide the network into separate logical networks. Affiliation to a specific VLAN is marked with an addition field in the Ethernet header. This is also known as a VLAN tag.

The Multiple Spanning Tree Protocol, defined in the same standard, is ignored in this research because with RSTP there is already a spanning tree protocol being analysed and the basic mechanism is similar to RSTP.

VLAN Hopping is an attack where the adversary is able to send traffic from one VLAN into another one (without going through a router). The frames sent by the attacker to the switch have two VLAN tags. The outer tag uses the same VLAN to which the access port is assigned. Normally traffic on such a port would not be tagged. The switch receiving the frame strips the outer tag but ignores the inner tag. When the access VLAN of the attacker and the native VLAN on the trunk match, the switch does not tag the frame when it is sent over a trunk link to the next switch. The receiving switch then interprets the inner tag and forwards the frame according to the VLAN ID in the tag. The attack only allows the sending of traffic from one VLAN into another, return traffic is not possible. Furthermore, the access VLAN of the attacker and the native VLAN of the trunk have to match in order for the attack to be successful [32, 34].

Attacks based on Autotrunking Mode are ignored in this research because they are based on the proprietary Dynamic Trunking Protocol.

3.3 Network Layer

3.3.1 Internet Protocol version 6 (IPv6)

IPv6, defined in RFC 2460 [35], is used for end-to-end connectivity between nodes. The protocol has undergone several changes and modifications. RFC 5095 [36] deprecates the type 0 routing header, RFC 5722 [37] specifies the handling of overlapping IPv6 fragments and RFC 6437 [38] defines IPv6 flow labels. RFC 4291 [14] standardises the IPv6 addressing architecture and RFC 5952 [39] states recommendations for the text representation of IPv6 addresses. RFC 4294 [40] defines the IPv6 node requirements and RFC 4862 [41] defines how a host can auto-configure an IPv6 interface. Table 3.1 shows an overview of the standards most relevant to this research.

IP Multicast [42], Multicast Listener Discover [43], multihoming [44], mobility [45], Internet Protocol Security (IPSec) [46, 47, 48] and Secure Neighbour Discovery [49] are not analysed in this research. Dual-stack issues, problems related to the simultaneous operation of an IPv4 and IPv6 network, network address translation, and Jumbograms [50] are ignored as well. For a sender it is possible to use any IP address as a source address (IP address spoofing). This is, by itself, not an attack but allows disguising the identity of an attacker [51].

While RFC 5095 [36] deprecates the use of the type 0 routing header, some operating systems still interpret it. An attacker could create an IP packet with a type 0 routing header and sends it to a vulnerable node. The node interprets the header and forwards it accordingly. An attacker could use this attack, for example, to evade a firewall that would otherwise filter the attacker's traffic [12]. The extension header could also be misused to create a packet that is sent back and forth between two nodes. If the two nodes are on opposite sites of a tunnel, the attack consumes even more resources [52].

An attacker could use packet fragmentation for several malicious purposes. One attack would be to overload a victim with too many small fragments [53]. Another attack could create overlapping fragments to exploit the fragment assembly algorithm of the receiving node [53]. Furthermore, an attacker could send multiple fragments to a node but drop the last one. This blocks resources on the node until it runs into a timeout and drops all fragments [12]. Fragments could also be used to disguise malicious parts of an attack. A packet could be created with so many extension headers that the header of the next higher layer is in the second fragment. This might be used to evade filtering by a firewall or detection by an IDS.

The Router Alert Option header could create additional load on a router because it requests the router to take a closer look at the content of the packet [12]. Unknown option headers or specially crafted option headers could be used to attack the protocol parser of IPv6 nodes [16].

3.3.2 Internet Control Message Protocol version 6 (ICMPv6)

ICMPv6, as opposed to ICMPv4, is essential for the functioning of IPv6 [54]. It is used for sending informal and error messages [55], for resolving data-link layer address of neighbour nodes [56] and for Stateless Address Auto-Configuration (SLAAC) [41]. Table 3.1 shows an overview of the standards most relevant to this research.

ICMP error and informal messages can be misused in several ways. The smallest Maximum Transfer Unit (MTU) in IPv6 is 1280 bytes [35]. This gives a lot of room for misusing ICMP messages for creating covert channels [12]. Another way to misuse the error messages is to send an ICMP Packet Too Big messages to a victim in order to force it to send smaller packets than necessary. Depending on the client's IPv6 stack implementation, it is even possible to go lower than the smallest MTU.

ICMP is used to discover the data-link layer address of nodes in the same data-link layer. The process is called Neighbour Discovery (ND) and does essentially what the Address Resolution Protocol (ARP) did in IPv4. Unfortunately, ND inherited also the vulnerabilities from ARP. For example, an attacker can reply with spoofed Neighbour Advertisement (NA) messages when a victim tries to resolve the data-link layer address of a node [51]. Consequently, the victim addresses traffic to this node with the data-link layer address received from the attacker. The attacker can also send spoofed Neighbour Solicitation (NS) messages. IPv6 nodes update their Neighbour Cache when they receive NS messages [57]. The Neighbour Unreachability Detection (NUD) mechanism is used to age out entries in the Neighbour Cache. An attacker can reply to NS messages with spoofed NA messages to keep entries in the victim's Neighbour Cache active. This might be used as a DoS attack [57].

An attacker can also spoof ICMP Redirect messages for any given destination. The attacker spoofs an ICMP Redirect message with the data-link layer address of the first-hop router to appear legitimate and replies to further NUD messages sent by the victim to keep the redirection active [12, 57].

SLAAC simplifies the task of configuring nodes but also creates several new vulnerabilities. A client requesting the current network prefix sends a Router Solicitation (RS) message and an attacker can reply with a spoofed Router Advertisement (RA). The client uses whatever address the attacker chooses to put into the RA message. If the client already has received a valid RA message from the genuine default router, the attacker starts resending the same RA message but sets the lifetime to zero. Afterwards the attacker sends its own RA message [15]. Further, an attacker can simply launch a DoS attack by sending many RA messages with different source addresses (CVE-2010-4669) [58]. Another attack is to hinder a node from using its new IPv6 address. A node has to perform Duplicated Address Detection (DAD) before using the newly created IP address. The client, therefore, sends ND messages for the new address and assumes that it is safe to use the address, if there is no reply within a certain time. An attack can stop a client from using any address created by SLAAC simply by replying to all ND messages with a corresponding NA message [13].

An attacker might also “kill” the default router by a DoS attack against it or by spoofing RA messages with a zero lifetime. If there is no default router in the local network, the hosts assume that they are able to reach every destination directly. Therefore, they send NS messages for any destination address they try to reach. An attacker then replies with NA messages to receive traffic for destinations at will [57].

Another DoS attack is to claim arbitrary prefixes as on-link. Thus, victims on the same data-link do not send the traffic for the prefix to the default router but create NS messages instead. When no device responds to the NS messages, the clients time out eventually. The lifetime of the RA message could even be infinite [57].

Parameter spoofing is when an attacker sends RA messages with bogus parameters, for example, a very small current hop limit so that packets get dropped before reaching their destination or with the O or M flag set to attack the address configuration [57].

Sending a large number of packets to non-existing host in a subnet is a DoS attack against the last-hop router. It has to resolve the data-link layer addresses for all those IP addresses. Therefore, it might be possible that the router does not reply to genuine NS messages in the same segments as it might be too busy resolving data-link layer addresses for the bogus IP packets sent by the attacker. This DoS attack is a special case because it is possible to execute it remotely [57] while for the other attacks the attacker and the victim need to be connected to the same data-link layer.

Table 3.1: IPv6 and ICMPv6 protocol standards most relevant for this research.

Standard	Content
RFC 2460	IPv6 specification
RFC 3756	Neighbour discovery trust models and threats
RFC 4291	IPv6 addressing architecture
RFC 4294	IPv6 node requirements
RFC 4443	ICMPv6 specification
RFC 4861	Neighbour discovery
RFC 4862	Stateless address auto-configuration
RFC 4890	Filtering ICMPv6 messages in firewalls
RFC 5095	Deprecation of type 0 routing header
RFC 5722	Handling of overlapping IPv6 fragments
RFC 5952	Recommendation for IPv6 address text representation
RFC 6437	IPv6 flow label specification

3.3.3 Routing Information Protocol next-generation (RIPng)

RIPng, defined in RFC 2080 [59], is a distance-vector routing protocol. It is merely an extension of RIPv2 to accommodate the longer IPv6 addresses. While not widely used anymore, it is the only non-proprietary distance-vector protocol and is, therefore, included in this research.

The RIPng header does not contain an authentication field. The protocol fully relies on IPSec Authentication Header (AH) to ensure the integrity and authentication of the exchanged routing messages. Because there is no established standard to create Security Associations between multiple parties (the Internet Key Exchange protocol is only designed to work between two parties [60]), IPSec does not work well as a protection mechanism in a broadcast segment with multiple routers. It would be necessary to create a logical tunnel between each of the routers to protect RIPng, an approach that scales very badly.

Without the protection of IPSec, however, the protocol is easily attackable. Routes can be injected and withdrawn at will, black hole routes can be established, and DoS attacks can be run against the routers or another victims. For example, a request for the whole routing table is sent to the RIPng multicast address ff02::9 and the source address in the request is spoofed with the victim's address.

3.3.4 Open Shortest Path First version 3 (OSPFv3)

OSPFv3, specified in RFC 5340 [61], is a link-state routing protocol commonly found in corporate networks. The authentication fields available in the previous version were removed and instead it is suggested to use IPSec to protect the protocol [62].

In general, OSPFv3 suffers the same issues as RIPng regarding using IPSec to protect the routing protocol. RFC 4552 [62] suggests using manual keying to run OSPFv3 over a broadcast medium. All the systems would share an inbound and an outbound Security Association. This is not a scalable solution and does not protect against replay attacks. Furthermore, a key created by an administrator is likely to be too weak to be used for a long time [13].

3.3.5 6in4

Until IPv6 is widely implemented, IPv6 networks need to traverse IPv4 networks to reach other IPv6 networks. Tunnelling is an important and often-used concept to traverse IPv4 networks with IPv6 traffic. There are several mechanisms available to tunnel IPv6 network traffic through an IPv4 network.

RFC 4213 [63] defines 6in4, a static site-to-site tunnelling mechanism. It allows expanding IPv6 network by tunnelling traffic through IPv4 networks. The protocol simply creates an IPv4 packet addressed to the other tunnel endpoint and places the IPv6 packet into the payload. The receiving tunnel endpoint strips the IPv4 header and forwards the packet according to the IPv6 destination address.

Generally, any packet that would force the tunnel endpoint to fragment a packet could be used as a DoS attack. Tunnels can also be misused in amplifier attacks. An IP packet could be crafted that it is sent back and forth between two tunnel endpoints (e.g. with an ICMP Source Routing header) [51]. Sending spoofed ICMPv4 errors to the tunnel endpoint could get it to send ICMPv6 errors back to the original sender. This opens an attack vector from an IPv4 network to an IPv6 host. Furthermore, an attacker could send packets with a spoofed tunnel header to a tunnel endpoint to inject packets into the IPv6 network.

3.3.6 General Routing Encapsulation (GRE)

GRE, another static site-to-site tunnelling mechanism, is defined in RFC 2784 [64]. GRE is more flexible than 6in4 and allows tunnelling protocols other than IPv6. RFC 2890 [65] extends the GRE header with a key and a sequence field that allows identifying a traffic flow and reordering out-of-sequence packets respectively. The key is not intended as an authentication mechanism.

The types of attacks against GRE are essentially the same as those against 6in4 tunnels. The sequence number might open a new attack vector against the tunnel endpoint and its reordering algorithm.

3.4 Transport Layer

3.4.1 User Datagram Protocol (UDP)

UDP is defined in RFC 768 [66]. It is used by DHCPv6 and DNS to transport the application layer data.

To the knowledge of the author, there are no known security issues related to UDP. The only issue is that the stateless nature of the protocol makes it harder for firewalls and IDS to keep track of connections. They usually use timeout-based mechanism to determine whether a UDP session has ended.

3.4.2 Transport Control Protocol (TCP)

TCP, defined in RFC 793 [67], is one of the most commonly used transport layer nowadys. RFC 6298 [68] clarifies the computing of the retransmission timer. RFC 1323 [69] adds performance features to the protocol such as the window scale option, round-trip time measurement and protection against wrapped sequence numbers. RFC 5681 [70] adds the congestion control algorithms Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery to the protocol.

There exist a variety of different types of attacks against TCP [71]. A segment with special combinations of header fields and retransmission time sampling is used to fingerprint the TCP implementation of a host in order to identify the operating system. Other attacks use the PSH Flag, an excessive number of SYN segment or just establish TCP sessions to abandon them afterwards in order to block resources on the remote host. Some attacks try to reset an established TCP session between two hosts by guessing and spoofing the necessary header fields. Further attacks exploit the sequence number concept of TCP to inject data, create overlapping frames, or intentionally drop a frame so the receiver wastes resources while waiting for the final frame to arrive. Kumar et al. [72] describe a TCP attack where they combine optimistic acknowledging with a timeout to create a low-rate flooding DoS attack. They steer the TCP sender into creating a DoS attack.

3.5 Application Layer

3.5.1 Domain Name System (DNS)

DNS allows resolving hostnames into IP addresses and vice versa. It is defined in several RFCs. RFC 1034 [73] focuses on the concepts and how the protocol works, while RFC 1035 [74] specifies the details for the implementation. Further clarifications of the DNS specification are provided in RFC 2181 [75] and RFC 4343 [76]. RFC 1123 [77] section 6 describes the requirement for an Internet host regarding DNS. DNS Security Extensions [78], Dynamic Update DNS [79, 80] and internationalized domain names [81, 82, 83, 84] are ignored in this research.

A new AAAA Resource Record (RR) was created to support IPv6 addresses in DNS [85]. Because only a new RR type is required, all DNS attacks mentioned in RFC 3833 [86] and RFC 5358 [87] against resolving IPv4 addresses work also against resolving IPv6 addresses. RFC 4472 [88] covers general issues and considerations regarding the operation of a DNS server for IPv6 and RFC 3901 [89] regarding mixed transport (IPv4 and IPv6) in dual-stack environments.

RFC 4074 [90] describes some attacks specific to dual-stack hosts (host with an IPv4 and IPv6 network stack). CVE-2003-1132 [91] reports an attack where an implementation error is misused for a DoS attack. It depends on an authoritative name server to respond with an NXDOMAIN error (RCODE 3) instead of no error (RCODE 0) in the case of a non-existing AAAA record. In that case, an attacker could send queries to a caching name server for AAAA records. The cache name server would keep the reply in its negative cache and negate queries for the same name from any other client, even if they send a request for another RR type than AAAA.

In addition, the handling of DNS responses larger the 512 bytes might be a problem for firewalls or IDS [13]. Also, cache poisoning is still possible but reasonably harder to achieve with the recommendations of RFC 5452 [92].

3.5.2 Dynamic Host Configuration Protocol (DHCP)

DHCPv6 is defined in RFC 3315 [93]. It describes a stateful alternative to IPv6 SLAAC. DHCPv6 is a new protocol and is not backwards compatible with DHCPv4. It supports two modes, one where it works as a DHCPv4 server and gives out leases for addresses and another one, where it complements SLAAC by offering additional configuration options (e.g. the address of the DNS server - RFC 3646 [94]). In the second mode, defined in RFC 3736 [95], the server does not keep track of the IP addresses and works stateless. The standard supports authentication of the DHCP messages based on a shared key and a hash message authentication code.

Consuming all available leases from the DHCP server with spoofed client requests is called a DHCP consumption or DHCP starvation attack [96]. IPv6 subnets are typically very large (1.8×10^{19} potential addresses in /64 network) and it is difficult for an attacker to consume all DHCP leases. However, the attack might create significant load on the DHCP server and might stop it from serving genuine DHCP clients.

Another attack is to run a rogue DHCPv6 server and send forged advertise and reply messages to victims requesting IPv6 addresses. The attacker could use it to send the client wrong DNS information for man-in-the-middle attacks or just DoS by assigning them a non-existing DNS server [12].

As DHCP servers typically assign IP addresses sequentially, they simplify scanning for potential targets [12]. Also, all DHCP servers can be addressed with two IPv6 multicast addresses ff02::1:2 (link-local) and ff05::1:3 (site-local).

3.6 Summary

This chapter described the network protocol standards relevant for this research and the known attacks against these protocols. The understanding of the protocols is essential for comprehending network scenarios and attacks based on these protocols. The comprehension of the network scenarios and attacks is crucial for investigating the scenarios and for identifying relevant information for a successful forensic investigation.

The research methodology adopted in this thesis is presented in the next chapter.

Chapter 4

Research Methodology

4.1 Introduction

Chapter 3 reviewed network layer protocols and briefly described known attacks against these protocols. This chapter sets the objective of this research by stating the research questions and explaining the methodology chosen to answer them. In addition, it justifies the chosen approach and the design decisions.

Section 4.2 states the research question. Section 4.3 presents the research design and Section 4.4 justifies the chosen approach.

4.2 Research Questions

Endicott-Popovsky and Frincke [10] as well as Rowlingson [9] urge the implementation of forensic ready networks. Rowlingson [9] recommends a top-down approach by setting the strategic goal of having the ability to pursue legal action against intruders. He suggests focusing on specific business cases that require forensic evidence and collecting the necessary evidence proactively. As many business cases depend on interconnected systems, the ability to retrace the communication between these systems is an important part of retracing the whole case. This research attempts to identify the information that should be retained proactively to successful conduct investigations at a network level and attempts to answer the following two research questions:

1. What key information (e.g. states, mappings, events) is required to be retained in an IPv6 network to successfully investigate incidents?
2. How is this information stored in an IPv6 network?

The research focuses on IPv6 to limit its scope and to be up-to-date with the next step of the development of the IP protocol.

The following section describes the approach chosen to answer these research questions.

4.3 Research Design

This research uses an experimental approach to answer the research questions. Genuine and malicious network scenarios are run in a test bed (see B.2) and forensic investigations for these scenarios are conducted to identify and locate significant forensic information.

The relevant network protocols are grouped by network layer. The layers are processed bottom-up in the following order: data-link layer, network layer, transport layer, and application layer. The research is conducted in four iterations and each iteration focuses on one network layer. An iteration consists of the three phases: extraction, training, and verification.

4.3.1 Extraction Phase

In the extraction phase an empirical investigation of network protocol standards and related literature is conducted to identify and locate forensic information. The information is categorised based on the potential usefulness for an investigation and the degree of difficulty to collect it. The categories and criteria based on which they are assigned on are provided in Appendix A.1. A first set of information is created containing information that is valuable or critical for an investigation and that is easy to collect. This set of information is optimised in the training phase.

4.3.2 Training Phase

Based on standards, relevant literature and vulnerability databases a set of network scenarios is compiled. A scenario describes a specific sequence of events in chronological order. The set contains genuine scenarios that conform to the behaviour described in the protocol standards and malicious scenarios which represent attacks against the protocol.

The hypothesis is that the set of information created in the extraction phase suffices to retrace all network scenarios. The hypothesis is tested with experiments. An experiment consists of running a network scenario and retracing it with the set of information. If the retracing fails, the set of information is expanded with further relevant information and the experiment is repeated. 70 % of the identified network scenarios are run in experiments to optimise

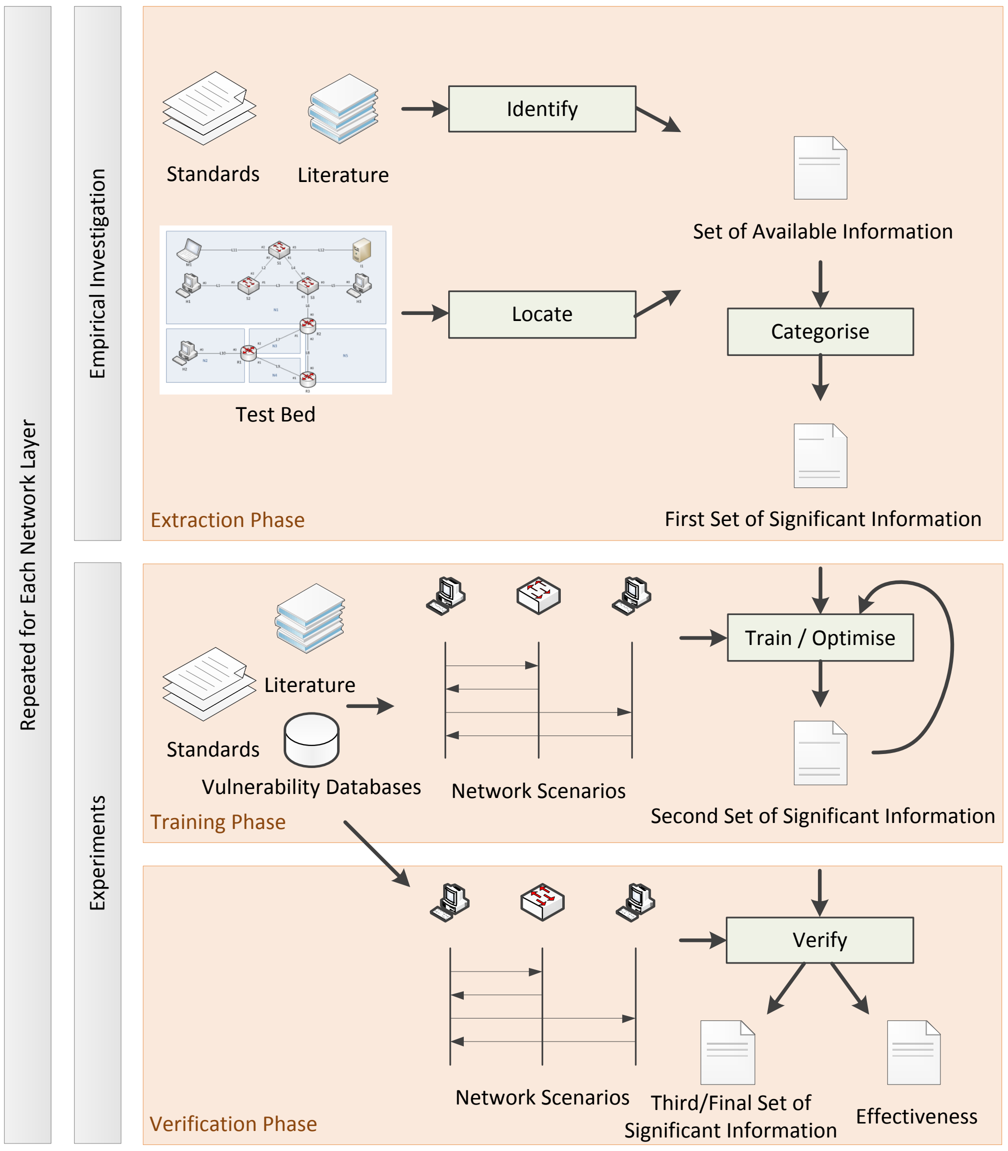

Figure 4.1: The workflow of one iteration of this research.

the initial set of information. The optimised set of information is verified in the next phase.

4.3.3 Verification Phase

The set of information is verified with the holdout approach [97] where 30 % of the network scenarios are held back for the verification phase and are not used during the training phase. The same experiments used for training are conducted with the network scenarios reserved for verification. However, the set of information is not further expanded and only the retractability of the scenarios is ascertained. The efficiency of the set of information is measured by dividing the number of verification scenarios that are retraceable by the total number of verification scenarios.

4.3.4 Results

When the testing and the verification for all network layers is completed, the set of significant information covers all four network layers and the effectiveness of each layer is ascertained. This final set of significant information is the set of information that is recommended to be retained in order to achieve network forensic readiness.

A practitioner, for example, a network or security engineer, can use the set of information as follows. The practitioner determines the active protocols in the subject network and consults the set to determine the information worthwhile retaining and how the information is stored. Afterwards, appropriate means to collect the recommended information are implemented. This leads to a forensically ready network where the information necessary to investigate network incidents is readily available.

4.4 Justification

4.4.1 Layer-based Bottom-up Approach

Carrier [98] shows the advantages of using layers in handling the complexity of network protocols, while Nikkel [17] and Casey [99] apply the approach successfully.

This research considers the network layers in a bottom-up manner in order to identify valuable information and to verify its usefulness for an investigation. Since higher layer network protocols use services and functionalities of lower layer protocols, a top-down dependency exists. A bottom-up approach allows resolving this dependency. Therefore, it makes sense to build forensic readiness on the lowest layer first and then expand it to the upper layers. This means that it should be possible to retrace the services used by higher layer network protocols before the forensic readiness is expanded to the network layer using these services.

4.4.2 Empirical Investigation

To identify potential information an empirical investigation of protocol standards and related literature is chosen. The protocol standard formally defines the network protocol behaviour and the exact states each participating entity can assume. Therefore, it is a natural source of information to retrace network protocol behaviour. The related literature supports the understanding of the network protocol.

An alternative approach would be to analyse the implementation of a specific network protocol. However, this approach would impede the investigation as the source code of a protocol implementation is more difficult to read and understand than the natural language of the protocol standard. Furthermore, it is likely that the source code contains parts that are not required by the protocol itself but are necessary to adapt the protocol to the specific operating system. In addition, the result would be specific to the analysed implementation and might not be transferable to other operating systems or even other versions of the same operating system. Finally, gaining access to the source code of some closed source operating system might be not possible or very difficult. Overall, an empirical investigation seems to be the simpler approach that should reveal results that are more portable.

4.4.3 Inductive Reasoning

The chosen approach is based on inductive reasoning. “Inductive reasoning starts from specific observations . . . and then develops a general conclusion from them” [100]. Based on the experimental investigation of a group of network scenarios it is concluded what information is necessary to successfully investigate any network scenario. Because the set of network scenarios cannot be determined in their entirety, every new scenario might prove the identified set of significant information insufficient.

Nonetheless, Walliman [100] states that inductive reasoning is still valid if the number of observations, on which the conclusion is based on, is large enough. The author is confident that, by using protocol standards and vulnerability databases as sources for training and verification scenarios, the number of scenarios is large enough to draw meaningful conclusions.

4.4.4 Experimental Approach

An alternative, positivist approach might be to create every possible network packet and attempt to retrace them. This is feasible as the frame length is limited and so is the number of different packets. Nevertheless, since scenarios are based on a sequence of packets and they trigger state and mapping changes, all permutations of all possible packets would have to be taken into account. As the number of packets following each other is not limited, the number of permutations is not limited either. Consequently, this approach is not feasible and the implemented experimental approach is more reasonable.

4.4.5 Verification with Holdout

The verification of the set of forensically significant information with the holdout approach, borrowed from data mining, seems to be the only possible way to verify the set. As the whole approach is based on inductive reasoning, and every new scenario could prove the set of significant information insufficient, the verification gives the recommend set of forensically significant information credibility.

4.5 Summary

This chapter presented the research questions, the research methodology used to answer them and the expected outcomes of this research. Further, the chosen research paradigm was discussed and the rationalisation for several design choices was given. This gives this research a solid base, a clear objective and a sound methodology.

The next chapter starts with the analysis of the data-link layer.

Chapter 5

Extraction and Refining of Forensic Information from Data-Link Layer Protocols

5.1 Introduction

The research methodology adapted in this thesis was presented in Chapter 4. This chapter presents the extraction and refining of forensic information from data-link layer protocols and discusses network scenarios that are especially interesting from a network forensics point of view.

An experimental approach was taken where network scenarios based on the data-link layer protocols were run in lab environment and retraced to identify relevant forensic information. This chapter covers the protocols Ethernet (IEEE Std 802.3 [26] and IEEE Std 802.3ab [27]), Bridging (IEEE Std 802.1D [28]) and VLANs (IEEE Std 802.1Q [33]).

Section 5.2 presents the network scenarios used to train the set of information and to verify the recommended set of information. Some relevant scenarios are discussed in details. Section 5.3 assesses the quality of the extracted and refined information and Section 5.4 presents the set of significant information recommended to be retained on the data-link layer in order to achieve a forensically ready network.

5.2 Network Scenarios

To extract the information from the data-link layer protocols the scenarios presented in Table 5.1 are used. Each scenario represents a sequence of events on the data-link layer with a malicious or genuine intent. The scenarios are randomly assigned to be used for the training of the set of information or the verification of the recommended set of information.

Table 5.1: Data-link layer scenarios with their intent (M=malicious, G=genuine) and purpose (T=training, V=verification).

Scenario	**Description**	**Intent**	**Purpose**
S.DG.01	Sending a Frame	D	T
S.DG.02	Establishing Spanning Tree	D	T
S.DG.03	Send Frame over Trunk	D	T
S.DM.01	Device is Removed or Destroyed	D	T
S.DM.02	Link Unplugged	D	V
S.DM.03	MAC Flooding	D	T
S.DM.04	MAC Spoofing	D	T
S.DM.05	RSTP Root Claim	D	V
S.DM.06	RSTP Eternal Root Election	D	V
S.DM.07	RSTP Topology Change Flooding	D	T
S.DM.08	VLAN Hopping	D	T
S.DM.09	Occupying a Link	D	V
S.DM.10	Administrative Access	D	T

The following six subsections discuss the retracing of selected scenarios that were either not retraceable in the training or verification phase, or they are otherwise interesting for a forensic investigation. The detailed description and discussion of all data-link layer scenarios can be found in Appendix B.3.

5.2.1 Sending a Frame

In scenario S.DG.01, a host S sends a frame to another host R on the data-link layer. Several things can be proven about that scenario. Firstly, that S has sent a frame and the exact point in time the frame was sent; secondly, the content (header and payload) of the sent frame; thirdly, the way the frame travelled through the network and, lastly, the exact point in time when R received the frame.

Without actually capturing the frames directly on the port where S is connected to the network, it is not possible to prove that S sent a specific frame at a specific time. However, it is possible to show that S sent frames within a certain timeframe. When the Filter Database (I.DB.05) is known and the timestamps showing when entries were created and removed are also available, it is possible to show when a certain MAC address sent traffic for the first time and on which access port the traffic was received. In addition, it is possible to determine the last time when a certain MAC address sent traffic by subtracting the Filter Database Ageing Time (I.DB.01) from the time when the entry related to that MAC address was removed from the Filter Database. Even though it is not possible to show the content of the communication between S and R, it is possible to show that there was communication between the two hosts.

The path the frame took through the network is retraceable by using the state of the Spanning Tree (I.DB.10) at the time the frame was sent. Another option to determine the path is to reconstruct it from the Filter Database (I.DB.05) of all network switches. Starting with the switch where S is connected to the network, the outgoing port is determined based on the Filter Database and the Interface Address (I.DE.03) of R. Based on the Physical Layout (I.DE.08) the switch behind this port is identified. The process is repeated on the next switch and any subsequent switch until the switch connected to R is reached. This process does not work when traffic is sent to a previously unknown MAC address because the Filter Database does not contain entries for the MAC address and the frame is flooded (sent out on all ports but the one the frame was received on). This means that there is some redundancy in the path reconstruction with the Filter Database and the Spanning Tree.

The exact point in time when R received the frame, the fact that R actually received the frame and the frame's content can only be retraced by a packet sniffer operating on R.

This scenario is not fully retraceable with the set of significant information. However, the Frame Payload (I.DE.02) is not added to the set of significant information because it is likely that the packet triggers events when it is received and passed to the next higher layer on R. This would render the information redundant.

5.2.2 MAC Flooding

MAC Flooding (S.DM.03) can be defined as sending an excessive number of frames with different source MAC addresses with the objective to overflow the Filter Database (I.DB.05) of a switch. Recognising new MAC addresses and adding them to the Filter Database is a normal operation for a switch. Therefore, the attack is simply defined by the frequency at which new MAC addresses are seen.

With the definition of what frequency is considered malicious and the monitoring of the Filter Database (I.DB.05), it is possible to detect and retrace MAC flooding attacks. The detection is based on the number of changes in the database for a certain port. The access port where the changes happen identifies the port where the attacker is connected to the network.

Another way to detect a MAC flooding attack would be to trigger a message when a switch needs to remove elements from the Filter Database because its memory is exhausted. However, this would not allow retracing the port on which the malicious frames were received.

Some commercial switches support sending a log message if the number of MAC address learned on a specific port exceeds a defined limit (e.g. Cisco Port Security [101]). This would allow do detect and retrace the attack to the access port without the Filter Database.

5.2.3 MAC Address Spoofing

MAC address spoofing (S.DM.04) with the intent to receive traffic addressed for another host can be retraced with the Filter Database (I.DB.05). The attack presents itself through a MAC address that keeps switching between two switch ports. The switching rate depends on how often the victim sends traffic.

Simply disconnecting a host from one port and reconnecting it to another port would lead to a similar change in the Filter Database. However, in that case, the Link Operational Status (I.DE.06) of the old port changes from up to down before the MAC address appears in the Filter Database under the new port. If the old interface stays in the state up and the MAC address switches between two ports, a MAC address spoofing attack is in progress.

To retrace which access port was used for the attack, the chronological order of the changes in the Filter Database and the Link Operation Status are important. Simplified, the port that learned the MAC address first is the port with the genuine client and the port that learned the MAC address later is the port where the attacker is connected to the network. In addition, the port that was first operational is more likely to be the one used by the genuine client. Both statements are false if the attacker disconnects the victim quickly before starting the attack. In addition, the attacker's port might be identifiable because the Filter Address contains multiple MAC addresses for a single port. This is the case when the attacker keeps sending traffic with the original MAC address while running the attack.

Finally, baseline data might support the investigation. If the network contains similar computers with similar operating systems, the time between the change of the Link Operational Status of a port and the new entry in the Filter Database for the same port should be constant within a certain tolerance. It is likely that the attacker needs a bit more time to finish booting up its operating system before starting the attack or the system is already booted and the attack starts immediately.

5.2.4 RSTP Topology Change Flooding

An RSTP TCN flooding attack (S.DM.07) can be detected on any bridge in the network by monitoring the frequency of TCNs (I.DB.12) received. To retrace the source of the attack it is necessary to monitor all bridges. With the help of the Spanning Tree (I.DB.10), it is possible to determine the branch or leaf of the spanning tree furthest away from the root bridge with an unusual high number of TCN messages. This is the bridge where the attacker is connected to the network. The actual access port on the switch cannot be determined if the bridge does not log the interface on which the TCN messages are received.

Another way to retrace the attack is to create a log message whenever a TCN message is received on an access port. Since only hosts are connected to access ports, TCN messages should not be received on them. Some commercial switches support similar features (e.g. Cisco Bridge Protocol Data Unit Filter/Guard [101]).

The set of significant information cannot be extended sufficiently in order to retrace this scenario to the access port. However, an improved logging of the received TCN messages would considerably simplify retracing this scenario.

5.2.5 VLAN Hopping

VLAN hopping (S.DM.08) with double tagged frames is difficult to detect and retrace. Double tagged frames can be detected and retraced with the Frame Header (I.DE.01). However, the double tagged frame only exists on the link between the attacker and the first switch. Therefore, a forensically ready network needs to be able to capture traffic in a way that preserves the VLAN tags as received on the access switch.

If a frame is received on an access port (Interface Role - I.DV.01) with one or multiple VLAN tags in the Frame Header (I.DE.01), the switch should drop the frame and create a log message. The message should include the interface on which the frame was received, the VLAN tag(s) the frame contains and the reason for dropping the frame. Such a log message would allow detecting and retracing the attack but would require a significant extension of the security/verification features of the switch.

The set of significant information cannot be extended sufficiently in order to allow the retracing of this scenario and future work would need to concentrate on resolving this issue.

5.2.6 Occupying a Link

Occupying a link (S.DM.09) by flooding the network with traffic is a DoS attack. With the knowledge of the Physical Layout (I.DE.08) of the network, the Traffic Counter (I.DE.11) and the current Spanning Tree Port Status (I.DB.10) it is possible to locate the access port of the attacker. The interfaces with a large amount of incoming traffic point towards the attacker. Retracing the access port is possible by following the interfaces switch by switch until the access port is found.

To show the plausibility of the attack, the Link Speed (I.DE.07) is also relevant. For example, if an attacker connects only with a 10 Mbps connection to the network it is not feasible that the attacker occupies a link with 1 Gbps capacity.

While this scenario is retraceable with the current set of significant information, the set is extended with the Link Speed (I.DE.07) to be able to show the feasibility of the attack.

5.3 Verification

The quality of the set of significant information after the training phase was assured with the verification scenarios (see Table 5.1). All four verification scenarios could be retraced with the set of significant information after the training phase.

Even though scenario S.DM.09 is retraceable, the Link Speed (I.DE.07) was added to the final set of significant information to achieve better credibility of the collected evidence.

5.4 Results

The following three subsections present the results from the extraction and refining process of forensic information from data-link layer protocols. Each subsection shows the forensically relevant information for one data-link layer protocol. The information should be retrained and changes in the information should be tracked. Information in the context of this research is not just the current value of a specific piece of information, for example, the value of a state variable, but also the time-period when the value was current. Retaining this information makes it more likely to be able to retrace network incidents on the data-link layer.

5.4.1 Ethernet - IEEE Std 802.3 and IEEE Std 802.ab

Table 5.2 shows the information that should be preserved for the Ethernet protocol standards IEEE Std 802.3 [26] and IEEE Std 802.3ab [27]. Not appearing in that table are the Frame Payload (I.DE.02) and the Link Master/Slave state (I.DE.05). It is expected that the Frame Payload is going to trigger events on the receiving host at a higher network layer and does not need to the retained separately and the Link Master/Slave state seems not relevant for any scenario studied in this thesis.

Table 5.2: Information recommended to be retained for the Ethernet protocol (IEEE Std 802.3 and IEEE Std 802.3ab).

Information	Description
I.DE.01	Frame Header
I.DE.03	Interface Address
I.DE.04	Duplex
I.DE.06	Link Operational Status
I.DE.07	Speed
I.DE.08	Physical Layout
I.DE.09	Traffic Filter
I.DE.10	Implementation Version
I.DE.11	Traffic Counter

5.4.2 Bridge - IEEE Std 802.1D

Table 5.3 shows the information that should be preserved for the Bridge protocol IEEE Std 802.1D [28].

The training phase showed how important the Filter Database (I.DB.05) is to retrace the access port in several scenarios. Even when the device spoofs the source MAC address, the spoofed address ends up in the Filter Database and can be used to retrace the access port where the spoofed frame entered the network.

Table 5.3: Information recommended to be retained for the Bridge protocol (IEEE Std 802.1D).

Information	**Description**
I.DB.01	Ageing Time
I.DB.02	Bridge Address
I.DB.03	Bridge Identifier
I.DB.04	Bridge Priority
I.DB.05	Filter Database
I.DB.06	Port Identifier
I.DB.07	Port Number
I.DB.08	Port Path Cost
I.DB.09	Port Priority
I.DB.10	Port State
I.DB.11	Root Path Cost
I.DB.12	TCN Received
I.DB.13	Port Role
I.DB.14	Root Bridge
I.DB.15	Implementation Version
I.DB.16	Bridge Log

5.4.3 VLAN - IEEE Std 802.1Q

Table 5.4 shows the information that should be preserved for VLANs as defined in IEEE Std 802.1Q [33].

As VLAN logically segment networks, their configuration is crucial in retracing network scenarios. Even when two hosts are connected to the same switch, if they are in different VLANs, the network traffic exchanged between them needs to flow through a router. Only with the VLAN configuration it is possible to retrace the path used by the traffic between these two hosts. Further, depending on the specific Spanning Tree protocol used, each VLAN might run and calculate its own Spanning Tree. Therefore, the path a frame takes through a network might differ depending on the VLAN.

Table 5.4: Information recommended to be retained for the VLAN protocol (IEEE Std 802.1Q).

Information	Description
I.DV.01	Interface Role
I.DV.02	Interface VLAN ID
I.DV.03	Implementation Version
I.DV.04	Traffic Filter

5.5 Summary

This chapter discussed several network scenarios of interest to a forensic investigator and identified the information necessary to retrace network scenarios on the data-link layer. With this information retained, a successful investigation of a network incident is more likely.

The chapter identified the Physical Layout (I.DE.08) of the network as an important information to retain. While it is an obvious element needed to retrace network scenario it can easily be missed when focusing on network protocols and device configurations. Further, the Filter Database (I.DB.05) has proven to be crucial for retracing the access port where a specific frame entered the network. It is even useful when an attacker spoofs the data-link layer source address of a frame because the Filter Database allows detecting the spoofing and determining on which port the spoofed frame entered the network.

The next chapter repeats the process with network layer protocol standards.

Chapter 6

Extraction and Refining of Forensic Information from Network Layer Protocols

6.1 Introduction

The extraction and refining of forensic information on the data-link layer was discussed in Chapter 5. This chapter focuses on network layer protocols and discusses network scenarios that are especially interesting from a network forensics point of view.

An experimental approach was taken where network scenarios based on the network layer protocols were run in lab environment and retraced to identify relevant forensic information. This chapter covers the protocols IPv6, ICMPv6, 6in4, GRE, RIPng, and OSPF.

Section 6.2 presents the network scenarios used to train the set of information and to verify the recommended set of information. Some relevant scenarios are discussed in details. Section 6.3 assesses the quality of the extracted and refined information and Section 6.4 presents the set of significant information recommended to be retained on the network layer in order to achieve a forensically ready network.

6.2 Network Scenarios

To extract the information from the network layer protocols the scenarios presented in Table 6.1 are used. Each scenario represents a sequence of events on the network layer with a malicious or genuine intent. The scenarios are randomly assigned to be used for the training of the set of information or the verification of the recommended set of information.

Table 6.1: Network layer scenarios with their intent (M=malicious, G=genuine) and purpose (T=training, V=verification).

Scenario	Description	Intent	Purpose
S.NG.01	Enabling and Configuring IP	N	T
S.NG.02	Sending a Packet to an On-Link Host	N	T
S.NG.03	Sending a Packet to an Off-Link Host	N	T
S.NG.04	Sending Packet to a Non-Existing Host	N	T
S.NG.05	Sending a Too Large Packet	N	T
S.NG.06	Sending a Packet with a Small Hop Limit	N	V
S.NG.07	Sending a Packet with Unknown Parameter	N	V
S.NG.08	Sending an ICMP Echo Request	N	V
S.NG.09	Sending Traffic through a 6in4 Tunnel	N	T
S.NG.10	Sending Traffic through a GRE Tunnel	N	T
S.NG.11	Establishing Routing with RIPng	N	V
S.NG.12	Establishing Routing with OSPFv3	N	T
S.NM.01	Sending a Packet with a Spoofed Source Address	N	T
S.NM.02	Circumventing Traffic Filtering with the Source Routing Header	N	T
S.NM.03	Overlapping IP Fragments	N	T
S.NM.04	DoS IP Fragment Reassembler	N	T
S.NM.05	Unknown Extension Header	N	V
S.NM.06	DoS attack with excessive Hop-by-Hop Options	N	T
S.NM.07	IP Options as a Covert Channel	N	T
S.NM.08	Specially created Header Option	N	T
S.NM.09	Administrative Access	N	T
S.NM.10	DoS with the Router Alert Option	N	T
S.NM.11	DoS Last Hop Router	N	V
S.NM.12	Spoof NA messages	N	T
S.NM.13	Spoof NS messages	N	T
S.NM.14	DoS Neighbour Cache	N	T
S.NM.15	Spoof RA messages	N	T
S.NM.16	Clean the Default Router List	N	T
S.NM.17	Flood the Default Router List	N	T

Continued on next page

Scenario	Description	Intent	Purpose
S.NM.18	Lower Hop-Count	N	T
S.NM.19	DoS Auto-Configuration	N	V
S.NM.20	Traffic Redirect with ICMP Redirect Messages	N	V
S.NM.21	DoS with ICMP Packet Too Big messages	N	V
S.NM.22	DoS with ICMP Error Messages	N	V
S.NM.23	Covert Channel with ICMP messages	N	T
S.NM.24	DoS with ICMP message flooding	N	T
S.NM.25	Spoof Tunnel Packet Source	N	T
S.NM.26	IPv4 Reassembly Attack against the Tunnel Endpoint	N	T
S.NM.27	Extend Attack Reach with Encapsulated Packets	N	T
S.NM.28	Circumvent Traffic Filtering	N	T
S.NM.29	DoS the Tunnel with Malicious Payload Addressing	N	V
S.NM.30	Lower Tunnel MTU	N	V
S.NM.31	DoS on GRE tunnels with Sequence Numbers	N	V
S.NM.32	Redirect Traffic with more Specific Routes	N	V
S.NM.33	Redirect Traffic with Lower Metric	N	T
S.NM.34	DoS RIPng Router with Requests	N	T
S.NM.35	DoS RIPng Routers with Triggered Updates	N	T
S.NM.36	Redirect Traffic with more Specific Routes	N	T
S.NM.37	DoS OSPF Router with Topology Changes	N	V
S.NM.38	DoS OSPF Router by Flooding Hello Packets	N	T
S.NM.39	DoS OSPF Router with Spoofed Hello Packets	N	T

The following ten subsections discuss the retracing of selected scenarios that were either not retraceable in the training or verification phase, or they are otherwise interesting for a forensic investigation. The detailed description and discussion of all scenarios can be found in Appendix B.4.

6.2.1 Enabling and Configuring IP

A host has three methods to retrieve an IP address. The first one is with SLAAC as described in scenario S.NG.01, the second one is with DHCP as described in scenario S.AG.05 and the last one is to manually pick one. The Interface Address (I.NI.08) covers all three methods.

However, if the Interface Address is not available, determining the IP addresses (link-local and global) that a host uses can be a strenuous task. The following options might support the task depending on the available information.

When a node sends an IP packet to a neighbour (on-link node), it has to resolve the data-link layer address of the receiver. To do so, the node sends an NS message (I.NC.03) asking for the data-link layer address corresponding to the receiver's IP address. The neighbour replies with an NA message (I.NC.04) announcing its data-link layer address. Tracing the NA messages allows building pairs of data-link layer and the corresponding network layer addresses specific to a node. One data-link layer address can be related to multiple network-layer addresses. The pairs can further be combined with the Filter Database (I.DB.05) with the data-link layer address as the linking element. This leads to triples in the form: access-port, data-link layer address, network layer addresses. Table 6.2 shows two examples of such a triple. Such a database could be created for each access network. Each piece of information should be unique in the database and if one element is known, the other two can be determined. If the data-link layer address is known, the access-port and the network layer addresses can be queried. If the access-port is known, the data-link layer and network layer addresses can be determined and if one network-layer address is known, the access port, data-link layer address and the other network-layer addresses can be concluded.

Table 6.2: Example triples of Access Port, Data-Link Layer (MAC) Address and Network Layer (IP) Address

Access Port	**MAC Address**	**IP Address(es)**
Switch 1, Port 2	00:00:00:03:16:00	fd00::2:200:ff:fe03:1600/64 fe80::200:ff:fe03:1600/64
Switch 2, Port 5	00:00:00:03:14:00	fd00::2:200:ff:fe03:1400/64 fe80::200:ff:fe03:1400/64

If the packet is captured in a subnet other than where it was created, the first step in finding the access port used to send the packet is to determine the access network of the sender. This is done with the source IP address in the Packet Header (I.NI.01) and the Routing Table (I.NI.09). Once the first-hop router of the sender is found, the Neighbour Cache (I.NC.06) is consulted to determine the data-link layer address of the sender. If the node is not found in the cache, an NS/NA message exchange needs to be triggered, for example with an ICMP Echo Request message to the IP address under investigation. When the data-link layer address is determined, the Filter Database (I.DB.05) is used to query the access port of the sender. This manual retracing procedure requires that the node under investigation replies to NS messages. With the database described in the previous paragraph, this is not an issue.

Another method is based on the interface identifier used in SLAAC. SLAAC combines the prefix announced by the router (RA messages) with the interface identifier [41] of the node. The interface identifier in the case of an Ethernet interface is based on the data-link layer (MAC) address. The 48-bit MAC address is extended to 64 bit interface identifier by adding `ff:fe` in the middle [14]. The final IP address is:

```
prefix | first half MAC | ff:fe | second half MAC
```

This works only if the host does SLAAC and does not use the privacy extension defined in RFC 4942 [102]. In that case, there is a strong link between the data-link layer and the network layer address of a node.

6.2.2 Sending a Packet to an On-Link Host

In scenario S.NG.02, a node S sends an IP packet to another node R and both nodes are on the same data-link layer. When S starts, it consults its Destination Cache (I.NC.07) to determine how to reach R. Because it is the first time S sends a packet to R, the Destination Cache of S does not have an entry for R. Therefore, S consults the Routing Table (I.NI.09) to determine how to reach R. It shows that R is on the same data-link layer and that it is reachable through a specific interface. S consults then its Neighbour Cache (I.NC.06) to determine the data-link layer address of R. Again, because it is the first time S sends a packet to R, the Neighbour Cache does not have an entry for R. Consequently, S creates and sends an NS message (I.NC.03) to determine the data-link layer address of R. R receives the NS message and replies with a corresponding NA message (I.NC.04). S receives the NA message, updates its Neighbour and its Destination Cache, and sends the packet to R. Sending the packet on the data-link layer is described in scenario S.DG.01 (see Subsection 5.2.1 for a discussion of the scenario).

Without actually capturing the frames directly on the ports where S and R are connected to the network, it is not possible to prove that a specific packet was sent from S to R at a specific time. However, it is possible to show that there was communication between S and R based on the changes in the Neighbour and Destination Cache of both nodes. With the knowledge of the cache timeout, it is also possible to estimate the timestamp of the last communication between the two systems.

This scenario is not fully retraceable with the current set of significant information. However, the Packet Payload (I.NI.14) is not added to the set of significant information because it is likely that the packet triggers events when it is received and passed to the next higher layer on R. This would render the information redundant.

6.2.3 Sending a Packet to an Off-Link Host

Scenario S.NG.03 is very similar to the previous one. The only difference is that the receiver R is now off-link and the packet is sent to the default router instead of sending it directly to R. Consequently, when S looks up the Routing Table (I.NI.09), it determines that in order to reach R it needs to send the packet to the default router. S does not need to determine the data-link layer address of the default router because it is already in the Neighbour Cache (I.NC.06). It is added and updated every time S receives an RA message with the source data-link layer option from the router. S sends the packet to the default router. The default router receives the packet, decrements the hop limit field in the packet header, looks up the destination IP address in the Destination Cache (I.NC.07) or, if not found, in the Routing Table (I.NI.09) and forwards the packet to the next router. This procedure is repeating until the router connected to the access network of R is reached. This router determines that the destination address is in a directly connected network. Therefore, it retrieves the data-link layer address of R from the Neighbour Cache (I.NC.06) or creates and sends an NS message if the address is not found. Finally, it sends the packet to R.

The scenario is identical to the previous one but adds the element of packet routing. Therefore, the Routing Table (I.NI.09) and/or the Destination Cache (I.NC.07) of the involved routers are crucial for retracing the path of the packet through the network. The Routing Table and the Destination Cache can be verified with the Physical Layout (I.DE.08) of the network.

As in the previous scenario, it is not possible to retrace the exact content of a packet that was sent or received. However, the Packet Payload (I.NI.14) is not added to the set of significant information because it is likely that the packet triggers events when it is received and passed to the next higher layer on R. This would render the information redundant.

6.2.4 Source Address Spoofing

In scenario S.NM.01, a node M sends an IP packet to an off-link node R with a spoofed source address. This scenario is relatively difficult to retrace.

If M uses its own data-link layer source address (the other case is covered in S.DM.04, discussed in Section 5.2.3), the spoofing can be detected by a mismatch of the data-link layer and the network layer source address. Devices theoretically in a position to detect such a mismatch would be switches on the path to the default router and the default router itself. However, neither the switches nor the default router have all the information necessary to do so readily available.

The switches could use the Filter Database (I.DB.05) combined with a mechanism that traces NS/NA messages or DAD messages (if SLAAC or DHCP is used) to map network layer addresses to the entries in the Filter Database to create valid triples of access port, source data-link layer address, and source network layer addresses. Some commercial switches support similar features (e.g. Cisco IPv6 ND Inspection [103]).

The other device able to detect a mismatch is the default router. The router can compare the data-link layer address of the incoming packet with the data-link layer address stored in the Neighbour Cache (I.NC.06) for the network layer source address of the incoming packet. However, the Neighbour Cache might not contain the relevant entry and the router would need to send first an NS message to populate the Neighbour Cache. The additional delay might be undesirable and the attacker M could spoof the answer (see S.NM.13).

If M choses to use a source IP address outside the network range of the access network, the default router is able to detect the malicious activity with a traffic filter on the incoming interface. That interface needs to log packets with source IP addresses outside the network range assigned to the interface. This feature is also known as filtering based on the Reverse Path Forwarding (RPF).

New software needs to be developed to detect the network layer address spoofing. Subsection 9.4.6 provides a discussion of such software.

6.2.5 Retracing the Access Port of Scenarios with malicious Packet Headers

In many cases, it is possible to retrace scenarios with malicious packet headers. The relevant fields in the additional packet headers (I.NI.02 - I.NI.05) show the malignity of the traffic and the Packet Header (I.NI.01) can be used to determine the access port of the attacker.

If the traffic is captured in the access network of the attacker and it contains the Packet Header and the Frame Header (I.DE.01), retracing of the access port is relatively easy. The source data-link layer address (MAC address) in the Frame Header can be used to lookup the packet forwarding in the Filter Database (I.DB.05) of the switch. Each entry leads to another switch or, on the last switch, to the access port of the attacker.

If the Frame Header is not available or is not relevant because the traffic was not captured in the access network of the attacker, the source IP address can be used to determine the access port of the attacker. This is discussed in detail in Subsection 6.2.1. Subsection 5.2.3 discusses the case when the attacker spoofs the MAC address and Subsection 6.2.4 discusses the case when the attacker spoofs the IP address.

6.2.6 DoS Auto-Configuration

A DoS attack against a client using SLAAC (S.NM.19) by replying to all NS messages sent for DAD is harder to detect than to retrace. An unusual high number of NS/NA messages might indicate that the attack is taking place. However, if the attack is only directed against a single node, the number of NS/NA messages is minimal. In that case, additional software that is able to verify NS/NA messages entering a switch port (see also Subsection 9.4.6 for a discussion of such software) is needed.

The access port can be identified when the malicious NA message is identified. However, without the software mentioned above, it is not possible to prove that the NA message is malicious.

6.2.7 Malicious Tunnel Packets

Several scenarios (S.NM.25 - S.NM.29) exploit IPv4 source address spoofing to send malicious packets to a tunnel endpoint (GRE or 6in4). Since the only verification these tunnel protocols support is the verification of the source IPv4 address in the tunnel packet, detecting and retracing IPv4 address spoofing is crucial for retracing these attacks.

Retracing of IPv4 address spoofing is similar to IPv6 and brings the same challenges. The process in described in Subsection 6.2.4 can be adapted for IPv4 by replacing ND messages with ARP messages. In addition, filtering based on RPF allows detecting source IP address spoofing on first-hop routers when an address outside of the access network range is used. However, IPv4 is out of the scope of this research and, therefore, not further analysed.

6.2.8 Malicious Routing Packets

Several scenarios inject false routing information or overwhelm a router with a DoS attack. While the consequences of such attack can be detected in Routing Table (I.NI.09) to retrace the attack to an access port the Routing Protocol Packet (I.NR.13/I.NO.10) including the Packet Header (I.NI.01), the Frame Header (I.DE.01) is needed.

All attacks are easy to detect if the attacker is connected to an access network and the router is aware of which interfaces are connected to access networks. In that case, the router blocks the packet and generates a log message when it receives a routing protocol packet on such an interface. The log message should contain information about the interface on which the packet was received, the reason for the message and the complete packet including the Routing Protocol Packet, the Packet Header, and the Frame Header. Such a log message would simplify the retracing of these attacks significantly.

6.2.9 Scenarios that trigger ICMP Error Messages

Scenarios that create ICMP error messages (e.g. S.NG.06) are easier to retrace because the ICMP error message contains at least part of the original packet that triggered the error notification. This might be enough to identify the attacker based on the source IP address or it might be possible to locate the original packet in a network traffic capture that contains not just the complete Packet Header (I.NI.01) but also the Frame Header (I.DE.01).

Having both the ICMP error message payload and the original Packet Header, and being able to show their chronological order and the relation they have with each other makes the evidence more reliable.

6.2.10 Traffic Redirection with ICMP Redirect Messages

Traffic redirection with malicious ICMP Redirect messages (S.NM.20) can be detected by monitoring them. In a typical access network with one default router, it is not expected to see any ICMP Redirect messages (I.NC.05) at all. Even if the attacker intends only to disrupt the network traffic and uses spoofed network layer source addresses in the ICMP Redirect message, the data-link layer source address of the frame transporting the ICMP Redirect message can still be found in the Filter Database (I.DB.05).

A node receiving an ICMP message that is likely to be malicious should create a log message containing the reason for the message and the complete ICMP message including Packet Header and Frame Header. Such a log message would simplify a forensic investigation of such a scenario considerably.

6.3 Verification

The quality of the set of significant information after the training phase was assured with the set of verification network scenarios (see Table 6.1). All but two verification scenarios could be retraced with the set of significant information after the training phase. The efficiency of the set is 87 % for network layer scenarios, 13 out of 15 verification scenarios are retraceable with the set of significant information.

A DoS attack against a client using SLAAC (S.NM.19) is not detectable because it is not possible to link NA messages to an access port and, therefore, it cannot be proven that a specific NA message is malicious. Lowering the MTU of a Tunnel (S.NM.30) is also not retraceable because IPv4 address spoofing is not retraceable with the current set of significant information.

6.4 Results

The following six subsections present the results from the process of extraction and refining of forensic information from network layer protocols. Each subsection shows the forensically relevant information for one network layer protocol. The information should be retrained and changes in the information should be tracked. Information in the context of this research is not just the current value of a specific piece of information, for example, the value of a state variable, but also the time-period when the value was current. Retaining this information makes it more likely to be able to retrace network incidents on the network layer.

6.4.1 IPv6

Table 6.3 shows the information that should be preserved for the IPv6 protocol as defined in RFC 2460 [35], RFC 4291 [14], RFC 5095 [36], RFC 5722 [37], RFC 5952 [39] and RFC 6437 [38]. Not appearing in the table is the Packet Payload (I.NI.14). It is expected that the Packet Payload is going to trigger events on the receiving node on higher network layer and does not need to be retained separately.

The training and verification phase showed that many malicious scenarios (e.g. S.NM.11) are only retraceable if it is possible to retrace an access port based on a source IP address. The biggest hurdle to do so is the possibility of the sender spoofing the source IP address. Therefore, to achieve forensic readiness it is important to be able to detect and retrace source IP address spoofing.

The training also revealed out the necessity of retaining the Frame Header (I.DE.01) in order to be able to retrace the access port for some attacks. Therefore, the Frame Header was added to the set of significant information during the network layer analysis.

Table 6.3: Information recommended to be retained for IPv6.

Information	Description
I.NI.01	Packet Header
I.NI.02	Hop-by-Hop Options Header
I.NI.03	Routing Header
I.NI.04	Fragment Header
I.NI.05	Destination Options Header
I.NI.08	Interface Addresses
I.NI.09	Routing Table
I.NI.10	Node Type
I.NI.11	Traffic Filter
I.NI.12	Traffic Mangler
I.NI.13	Implementation Version
I.NI.15	Packet Unknown Header
I.NI.16	Packet Unknown Option
I.NI.17	Protocol Parameter

6.4.2 ICMPv6

Table 6.4 shows the information that should be preserved for the ICMPv6 protocol as defined in RFC 4443 [55], RFC 4861 [56], RFC 4862 [41] and RFC 4884 [104]. Not appearing in the table are the Send RA Flag (I.NC.12) and the Interface Reachable Time I.NC.25. The Send RA Flag is redundant because the RA message (I.NC.02) is part of the set and the Interface Reachable Time can be determined from changes in the Neighbour Cache (I.NC.06).

The training phase showed the diversity of ICMP messages and related attacks. It suggests a differentiated handling of the ICMP messages in the protocol stack. For example, an ICMP Redirect message is more likely to be misused and should, therefore, be logged by the protocol handler in detail to support forensic investigations.

Table 6.4: Information recommended to be retained for ICMPv6.

Information	Description
I.NC.01	Router Solicitation Message
I.NC.02	Router Advertise Message
I.NC.03	Neighbour Solicitation Message
I.NC.04	Neighbour Advertise Message
I.NC.05	Redirect Message
I.NC.06	Neighbour Cache
I.NC.07	Destination Cache
I.NC.08	Prefix List
I.NC.09	Default Router List
I.NC.10	Routing Enabled
I.NC.12	RA Maximum Interval
I.NC.13	RA Minimum Interval
I.NC.14	RA Managed Flag
I.NC.15	RA Other Configuration Flag
I.NC.16	RA Link MTU
I.NC.17	RA Reachable Time
I.NC.18	RA Retransmission Timer
I.NC.19	RA Current Hop Limit
I.NC.20	RA Default Lifetime
I.NC.21	RA Prefix List

Continued on next page

Information	Description
I.NC.22	Interface Link MTU
I.NC.23	Interface Cur Hop Limit
I.NC.24	Interface Base Reachable Time
I.NC.26	Interface Retransmission Timer
I.NC.27	Number of NS messages for DAD
I.NC.28	Destination Unreachable Message
I.NC.29	Packet Too Big Message
I.NC.30	Time Exceeded Message
I.NC.31	Parameter Problem Message
I.NC.32	Echo Request Message
I.NC.33	Echo Reply Message
I.NC.34	ICMP Error Message Rate Limit
I.NC.35	Traffic Filter
I.NC.36	Traffic Mangler
I.NC.37	Implementation Version

6.4.3 6in4

Table 6.5 shows the information that should be preserved for the 6in4 protocol as defined in RFC 4213 [63]. The table contains of all information that was extracted for the protocol.

Most malicious scenarios related to tunnelling are based on IPv4 source address spoofing. In consequence, IPv4 address spoofing must also be retraceable to achieve forensic readiness of an IPv6 network with active tunnels.

Table 6.5: Information recommended to be retained for the 6in4 protocol.

Information	**Description**
I.N6.01	Tunnel Endpoint Address
I.N6.02	Tunnel Endpoint Type
I.N6.03	Tunnel Type
I.N6.04	Tunnel MTU Size
I.N6.05	Traffic Encapsulated
I.N6.06	Traffic Decapsulated
I.N6.07	Tunnel MTU Mechanism
I.N6.08	Tunnel Hop Limit
I.N6.09	Strict Reverse Path Forwarding Check
I.N6.10	Ingress Filtering Tunnelled Traffic
I.N6.11	Link-Local Addresses
I.N6.12	Tunnel IP Header
I.N6.13	Implementation Version
I.N6.14	ICMPv4 Packet
I.N6.15	IPv4 Routing Table

6.4.4 GRE

Table 6.6 shows the information that should be preserved for the GRE protocol as defined in RFC 2784 [64] and RFC 2890 [65]. Not appearing in the table are the Out of Order Timer (I.NG.03) and the Buffer Limit (I.NG.04), since neither one is implemented in the kernel on the lab systems.

The issue with IPv4 address spoofing is the same for GRE tunnels as for 6in4 tunnels. In addition, events that lead to a packet drop (e.g. packet out of sequence) should be logged in detail including lower layer protocols to facilitate forensic investigations.

Table 6.6: Information recommended to be retained for the GRE protocol.

Information	**Description**
I.NG.01	Payload Type
I.NG.02	GRE Header
I.NG.05	Tunnel Endpoint Address
I.NG.06	Tunnel MTU Size
I.NG.07	Traffic Encapsulated
I.NG.08	Traffic Decapsulated
I.NG.09	Tunnel Features
I.NG.10	Implementation Version

6.4.5 RIPng

Table 6.7 shows the information that should be preserved for the RIPng protocol as defined in RFC 2080 [59]. The table contains of all information that was extracted for the protocol.

The training and verification phase showed that the routers could easily determine the interfaces on which routing protocol messages are expected. In the case where a message is received on an interface where no other router is expected, the messages should be logged in full detail including lower layer protocols. This would facilitate a forensic investigation.

Table 6.7: Information recommended to be retained for the RIPng protocol.

Information	Description
I.NR.01	Network Cost
I.NR.02	RIP Routers
I.NR.03	Directly Connected Networks
I.NR.04	Routing Table
I.NR.05	External Routes
I.NR.06	Socket
I.NR.07	Passive Interface
I.NR.08	Propagation Timer
I.NR.09	Link Local Addresses
I.NR.10	Split Horizon Setting
I.NR.11	Valid Neighbour List
I.NR.12	Filter List
I.NR.13	RIP Packet
I.NO.14	Implementation Version

6.4.6 OSPFv3

Table 6.8 shows the information that should be preserved for the OSPFv3 protocol as defined in RFC 5340 [61] and RFC 4552 [62]. The table contains of all information that was extracted for the protocol.

The requirement of the protocol to form a neighbour relationship before exchanging routing information is an advantage for a forensic investigation. Introducing new routers in a network is a rare event. Therefore, logging the forming of new neighbour relationships is essentially logging the beginning of new attacks. This considerably simplifies the retracing of attacks against OSPF.

Table 6.8: Information recommended to be retained for the OSPF protocol.

Information	**Description**
I.NO.01	OSPF Routers
I.NO.02	Instance
I.NO.03	Interfaces
I.NO.04	Areas
I.NO.05	Neighbour Table
I.NO.06	Link State Database
I.NO.07	Routing Table
I.NO.08	Virtual Links
I.NO.09	Interface State
I.NO.10	OSPF Packet
I.NO.11	Prefixes
I.NO.12	Area Prefix Filter
I.NO.13	Implementation Version

6.5 Summary

This chapter discussed several network scenarios of interest to a forensic investigator and identified the information necessary to retrace network scenarios on the network layer. With this information retained, a successful investigation of a network incident is more likely.

This chapter also shows the importance of being able to retrace source IP address spoofing and discusses briefly options to detect IP address spoofing. Many scenarios are only detectable and retraceable if it is possible to detect and retrace IP address spoofing. It also discusses how to retrace an access port of an IP packet based on the IP Header (I.NI.01) and the source IP address. In addition, it shows the importance of capturing and retaining ICMP messages. They are crucial for the correct behaviour of the IP protocol but are also easily misused for attacks against the protocol.

The next chapter repeats the process with protocol standards on the transport layer.

Chapter 7

Extraction and Refining of Forensic Information from Transport Layer Protocols

7.1 Introduction

The extraction and refining of forensic information on the network layer was discussed in Chapter 6. This chapter focuses on transport layer protocols and discusses network scenarios that are especially interesting from a network forensics point of view.

An experimental approach was taken where network scenarios based on the transport layer protocols were run in lab environment and retraced to identify relevant forensic information. This chapter covers the transport layer protocols TCP and UDP.

Section 7.2 presents the network scenarios used to train the set of information and to verify the recommended set of information. Some relevant scenarios are discussed in details. Section 7.3 assesses the quality of the extracted and refined information and Section 7.4 presents the set of significant information recommended to be retained on the transport layer in order to achieve a forensically ready network.

7.2 Network Scenarios

To extract the information from the transport layer protocols the scenarios presented in Table 7.1 are used. Each scenario represents a sequence of events on the transport layer with a malicious or genuine intent. The scenarios are randomly assigned to be used for the training of the set of information or the verification of the recommended set of information.

Table 7.1: Transport layer scenarios with their intent (M=malicious, G=genuine) and purpose (T=training, V=verification).

Scenario	Description	Intent	Purpose
S.TG.01	Send a UDP Segment	T	T
S.TG.02	Send a UDP Segment to a Closed Port	T	T
S.TG.03	Establish a UDP Stream	T	T
S.TG.04	Establish a TCP Session	T	T
S.TG.05	Initiate a TCP Session to a Closed Port	T	T
S.TM.01	UDP Flooding Attack	T	T
S.TM.02	Endless Stream	T	T
S.TM.03	Crash UDP Parser	T	T
S.TM.04	Hide sending Process	T	T
S.TM.05	Fingerprinting with Header Fields	T	T
S.TM.06	Fingerprinting by Retransmission Timeout Sampling	T	T
S.TM.07	DoS with Push Flag	T	V
S.TM.08	SYN Flooding Attack	T	T
S.TM.09	Connection Forgery Attack	T	V
S.TM.10	Connection Flooding Attack “Naptha”	T	T
S.TM.11	Overlong TCP Option	T	T
S.TM.12	Blind In-Window Attack (Timestamp)	T	V
S.TM.13	FIN-WAIT-2 Flooding Attack	T	V
S.TM.14	Resource Exhaustion Attack “Netkill”	T	T
S.TM.15	TCP Reassembly Buffer Attack	T	T
S.TM.16	Overlapping TCP segments	T	V
S.TM.17	ACK Division Attack	T	T
S.TM.18	Duplicated ACK Forgery	T	V
S.TM.19	Optimistic ACKing	T	T
S.TM.20	Blind Throughput-Reduction Attack	T	V
S.TM.21	Blind Flooding Attack	T	T
S.TM.22	Blind Connection Reset with RST Flag	T	V
S.TM.23	Blind Data-Injection Attack	T	T
S.TM.24	TCP Port Scan	T	V
S.TM.25	Blind Performance-Degrading Attack with ICMP	T	T
S.TM.26	Blind Connection-Reset Attack with ICMP	T	V
S.TM.27	TCP-based Traceroute	T	T

The following twelve subsections discuss the retracing of selected scenarios that were either not retraceable in the training or verification phase, or they are otherwise interesting for a forensic investigation. The detailed description and discussion of all scenarios can be found in Appendix B.5.

7.2.1 Send a UDP Segment

In scenario S.TG.01, a host S sends a UDP datagram from port 2000 to port 1000 of another host R.

Without actually capturing the frames directly on the ports where S and R are connected to the network, it is not possible to prove that a specific datagram was sent from S to R at a specific time. However, it is possible to show that there was communication between S and R on a specific UDP port at a specific time. The client programme on S creates a socket to send the datagram that is shown in the Active Sockets (I.TU.03). Further, the traffic filter between the two hosts keeps track of the UDP connection in the Traffic Filter Sessions (I.TU.05) and R opens first a listening socket and creates an established socket with S when the first datagram is received. Both sockets are shown in the Active Sockets (I.TU.03). The timestamp showing when the socket changes from a listening socket to an established socket refers to the time when R received the UDP packet. This should correlate with the creation of a new Traffic Filter Session (I.TU.05).

This scenario shows that it is possible to ascertain that there was communication between certain hosts even when the network traffic of the communication is not available.

This scenario is not fully retraceable with the current set of significant information. However, the Datagram Payload (I.TU.02) is not added to the set of significant information because it is likely that the datagrams trigger events when they are received and passed to the next higher layer on R. This renders the information redundant.

7.2.2 Send UDP Segment to closed Port

Scenario S.TG.02 is almost identical to the previous scenario. The only difference is that R is not listening on the addressed port and, therefore, the datagram triggers an ICMP Destination Unreachable (Port Unreachable) message.

The Active Sockets (I.TU.03) show the socket used to send the datagram as before and the Traffic Filter Sessions (I.TU.05) show the connection. However, R does not show any Active Sockets (I.TU.03) in this scenario. In addition, the

ICMP Destination Unreachable message (I.NC.28) is a good indication that this scenario is in progress. The inspection of the payload of the ICMP message allows retracing the involved ports and the involved systems (IP addresses). The access interfaces of the involved nodes can be retraced based on their IP addresses (see Subsection 6.2.1).

This scenario shows the usefulness of retaining ICMP messages to retrace UDP based scenarios. The ICMP message allows retracing the whole scenario and the actual UDP traffic is not necessarily needed.

7.2.3 Establish a UDP Stream

This scenario (S.TG.03) is almost identical to scenario S.TG.01. The only difference is that S and R exchange multiple datagrams and establish a two-way communication.

The difference in retracing this scenario is that the traffic filter moves the state of the traced UDP connection from being flagged as "unreplied" after the first packet, to "no flag" after the first reply, and eventually to being flagged as "assured" after receiving the second packet in either direction. If the connection is flagged as "assured", this is an indication that there were at least three packets exchanged. Unfortunately, the session tracking does not show traffic counters (bytes or packets). The reason for this is that the feature was not enabled when the kernel of the traffic filter was compiled. Traffic counters would allow retracing the amount of data transported using a specific session. While not needed for this scenario, such information could be helpful to show that a certain message or content was sent through the UDP stream.

This scenario shows the usefulness of the session tracking of additional network devices such as firewalls or IDS. This could be used to verify scenarios or as an addition source to prove the existence of a network connection between hosts.

7.2.4 Establish a TCP Session

In scenario S.TG.04, a host S establishes a TCP session with another host R. They exchange a few bytes and close the session afterwards. Without actually capturing the segments in front of S and R it is not possible to retrace the TCP stream. The client programme on S creates a socket to send the stream. The socket is shown in the Active Sockets (I.TT.03). The traffic filter keeps track of the TCP session in the Traffic Filter Sessions (I.TU.05) and R opens first a listening socket and creates later an established socket with S. Both sockets are shown in the Active Sockets (I.TT.03). The timestamp showing when the socket changes from a listening socket to an established socket refers to the time when the TCP three-way handshake was completed. This should correlate with the Traffic Filter Session (I.TU.05) on the traffic filter moving through the different TCP states. As with the UDP stream, a traffic counter (packets and bytes) would be useful as it could show that a certain amount of data was transferred over the TCP session.

This scenario is not fully retraceable with the current set of significant information. However, the Segment Payload (I.TT.02) is not added to the set of significant information because it is likely that the TCP stream triggers events on R when it is received and passed to the next higher layer on R. This renders the information redundant.

7.2.5 Initiate a TCP Session to Closed Port

Scenario S.TG.05 is almost identical to scenario discussed in the previous subsection. The only difference is that the receiver R is not listening on port 1000 and, therefore, replies with an ICMP Destination Unreachable (Port Unreachable) message.

The Active Sockets (I.TU.03) on the sender S show the socket used to establish the session in the state "SYN_SENT" and also the Traffic Filter Session (I.TU.05) moves through the states "NEW (SYN_SENT)" and "DESTROYED" when the ICMP message was seen. R does not show any Active Sockets (I.TU.03) in this scenario. The ICMP Destination Unreachable message (I.NC.28) indicates that this scenario is in progress. Further, the payload of the ICMP message allows retracing the involved ports and the involved systems (IP addresses).

7.2.6 Malicious UDP Header

The length field in the UDP header cannot be shorter than the minimum UDP header length of eight bytes and the checksum field cannot be zero because UDP is supposed to calculate the checksum when it runs over IPv6 [35]. Setting the source port of the UDP packet to zero is conform to the protocol standard. However, it does not allow identifying the sending application of the datagram.

To retrace any of these three scenarios with a malicious UDP header, the Datagram Header (I.TU.01) including the Packet Header (I.NI.01) is needed. The Datagram Header shows the malicious header fields and the Packet Header is used to retrace the access port where the malicious datagram entered the network.

Interestingly, the Linux kernel creates a log message for the first two scenarios. While it records the IP addresses and ports of the connection for the short UDP packet (length smaller than eight), it does not do that for the UDP packet with the checksum set to zero. It only informs that a UDP datagram without a checksum was received. The third case, a UDP datagram with a source port set to zero, is not logged by the kernel at all. Therefore, only the first of the three scenarios can be retraced with the kernel log. For a forensically ready network it would make sense to log all three scenarios in detail. A log message should include the reason for the message (i.e. the kind of suspicious UDP datagram that was received), the port and the actual packet.

The case where the UDP datagram has the source port set to zero might present a problem for a forensic investigation that tries to determine what application created a specific datagram. However, it is unlikely that such a datagram is created by a genuine application. Proactively collected evidence of a system that runs malicious software is unreliable and, therefore, the limitations of network forensic readiness are reached. To retrace the sending application a forensic investigation of the system under scrutiny is necessary. This is out side the scope of this thesis.

7.2.7 Malicious TCP Header

Scenarios S.TM.05 and S.TM.11 set the TCP header fields to malicious values. The first scenario sets the destination port to zero to conclude the operating system running on the remote node based on the response it receives. The second scenario creates an overlong TCP option with the intent to crash the protocol parser of the receiving node.

To retrace the attacks the Segment Header (I.TT.01) including the Packet Header (I.NI.01) is needed. The Segment Header is needed to show the malicious header and the Packet Header is needed to retrace the access port where the malicious segments entered the network.

This and the previous scenario show the importance of retaining the actual protocol headers to show the malicious intent and to be able to retrace the source of the traffic.

7.2.8 Misusing the TCP Retransmission Mechanism

In scenario S.TM.06 and S.TM.14, the TCP retransmission mechanism is misused. The first scenario identifies the operating system running on the remote node based on the timeout between the retransmissions of segments. The other scenario creates a DoS attack by binding resources on the remote node. The attacker gets the remote node to send a large amount of data but does not acknowledge the receiving of the segments transporting the data. This forces the remote node into holding the data in its memory and resending it several times.

Retracing the attack is possible with the information provided by the Segment Header (I.TT.01) including the Packet Header (I.NI.01). The Segment Header shows the malicious behaviour and the Packet Header is used to retrace the access port where the malicious segments entered the network.

The misuse of the TCP retransmission mechanism is very hard to detect because they exploit regular protocol behaviour. Exact the same packet exchange would happen if there is actually a problem in the network and segments are dropped for some reason. It is also very hard to prove the malicious intent of an attacker based on the network traffic. Only in the second scenario, it might be possible, based on the data the attacker requests, to show that there is no genuine purpose in doing so and the only explanation is the attack described here.

7.2.9 Consuming Resources on Remote Nodes

Scenarios S.TM.08, S.TM.10 and S.TM.15 have the goal to consume resources on the remote node. S.TM.08 does it by sending SYN segments with a spoofed source IP address to closed ports. The receiving node replies with an RST segment to the spoofed IP. This scenario consumes resources on the directly targeted node and the one that receives the RST segment. S.TM.10 establishes a session with the remote node and then abandons the session (no FIN or RST segment is sent). The remote node keeps the session in memory until it runs into a timeout. S.TM.15 drops an early segment in a sequence of segments and abandons the session afterwards. The receiving node keeps the already received segments in the memory while waiting for the missing segment.

The access port of the attacker can be retraced with the Segment Header (I.TT.01) and the Packet Header (I.NI.01). The malicious intent of the first scenario can be proven because the attacker spoofed the IP source address. For the other two scenarios, it is a bit harder. The malicious intent is only evident by the large number of sessions needed for a successful DoS attack. Therefore, it is necessary to retain evidence for each of these sessions.

7.2.10 Manipulating Window Scaling

In scenarios S.TM.17, S.TM.19 and S.TM.21 the Window Scaling of the remote node is manipulated. In S.TM.17, rather than sending a single ACK, the attacker sends a series of ACKs, each acknowledging a part of the received data. In S.TM.19, the attacker sends an ACK before actually receiving any data. In both cases, the sending window of the remote node is opened faster than intended by the TCP standard and, in consequence, the attacker receives the data faster. In the last scenario, the attacker triggers ACK messages in a session established between two hosts by injecting out of sequence segments. The receiver replies to the spoofed segment with an ACK for the last received segment that was correctly received in sequence. The attacker continues triggering ACK messages in the session and forces the host receiving the duplicated ACKs into Fast Recovery.

To retrace these attacks all the Segment Headers (I.TT.01) including the Packet Headers (I.NI.01) are needed. The Segment Header shows the divided ACK messages or the prematurely sending of the ACK messages for the first two attacks. The Packet Header is used to retrace the access port where the malicious segments entered the network (see Section 6.2.1).

The malicious intent in the last scenario is obvious because the source IP address of the sender needs to be spoofed. For the other two scenarios, it is a bit harder. It is necessary to show that the ACK messages were unnecessarily sent and that a reference implementation of TCP behaves differently.

To retrace the last attack, IP address spoofing must be retraceable (see Section 6.2.4 regarding the source address spoofing).

7.2.11 Blind Attacks

An attack is called blind when the attacker has to guess the header fields for a successful attack. Examples of such attacks are S.TM.09, S.TM.12, S.TM.22 and S.TM.23. In scenario S.TM.09, the attacker has to guess only the TCP sequence number of the remote node. In all other scenarios, the attacker needs to guess the four tuples (source IP, source port, destination IP, and destination port) and the current sequence numbers to run the attack successfully.

An interesting effect of blind attacks (excluding S.TM.09) is that they often lead to a stale TCP connection for one side of the TCP session but they also might recover. In scenario S.TM.23, for example, a host M injects a TCP segment blindly in an established TCP session between S and R. M has to guess the four tuple and the current sequence number used in the direction where the segment is injected. M sends the spoofed segment to R.

Depending on the size of the next segment sent by S after the attack, either the attack triggers many retransmissions and the TCP session gets stale or the TCP session recovers fully.

The TCP session goes stale if the first segment after the attack sent by S to R is smaller than the injected segment. In that case, R replies to the segment with a duplicated ACK to acknowledge the injected segment. Essentially, R tells S that it already has the data received. This confuses S because its internal sequence counter shows that the ACK is ahead of what data has already been sent to R. S continues to retransmit the segment in the timeout intervals and R keeps acknowledging the injected segment. S is not able to send any further data through the TCP session because R keeps dropping the segments.

The TCP session recovers when the first segment after the attack is larger than the injected segment. In that case, S appends the received data to the data already received in the injected segment and replies with an ACK. After that exchange, both S and R have again corresponding sequence counters and the TCP session stays intact. The same effects happens for scenario S.TM.12. The recovery of the TCP session depends in that scenario on the timestamp used by S after the attack. If the timestamp is newer than the one used by the attacker, the session recovers, otherwise it does not.

To retrace the attack all the Segment Headers (I.TT.01) including the Packet Headers (I.NI.01) are needed. The Segment Header shows the overlapping of the sequence numbers between the segments that S sent and the one that M injected. The Packet Header is used to show that the packet was injected into an established session. However, the Packet Header would indicate S as source of the attack (the spoofed segment uses the source IP address of S). Therefore, to fully retrace the attack, IP address spoofing must be retraceable (see Section 6.2.4).

7.2.12 Port Scanning

Several methods exist to detect if a port on a remote node is open and in the listening state or not (see Scenario S.TM.24). All port scans can be retraced with the Segment Headers (I.TT.01) including the Packet Headers (I.NI.01). The Segment Header is needed to show the specific port scan and the Packet

Header to retrace the access port where the malicious segments entered the network (see Section 6.2.1). Spoofing the IP source address does not make sense for port scanning as the response traffic would not reach the initiator of the port scan and the attacker could not conclude if the specific port is open or not.

For all methods, the destination port in the segments can be correlated with the Active Sockets (I.TT.03) on the scanned node to determine if the segment was sent to an actual active socket or just used to determine if the port is open.

The malignity of a port scan is difficult to prove. Especially if the port scan is just a regular TCP handshake. The strongest argument is based on the number of TCP handshakes initiated from a single source to one specific destination and to different ports. Above a certain frequency it is hard to argue that the connection requests were initiated because of a mistake and without a malicious purpose.

7.3 Verification

The quality of the set of significant information after the training phase was assured with the set of verification network scenarios (see Table 7.1). All ten verification scenarios could be retraced with the set of significant information after the training phase.

7.4 Results

The following two subsections present the results from the extraction and refining process of forensic information from transport layer protocols. Each subsection shows the forensically relevant information for one transport layer protocol. The information should be retrained and changes in the information should be tracked. Information in the context of this research is not just the current value of a specific piece of information, for example, the value of a state variable, but also the time-period when the value was current. Retaining this information makes it more likely to be able to retrace network incidents on the transport layer.

7.4.1 UDP

Table 7.2 shows the information that should be preserved for the UDP protocol as defined in RFC 768 [66]. The only information identified but not appearing in the table is the Datagram Payload (I.TU.02). It is expected that the Datagram Payload is going to trigger events on the receiving host in the higher network layers and does not need to be retained separately.

Table 7.2: Information recommended to be retained for UDP.

Information	**Description**
I.TU.01	Datagram Header
I.TU.03	Active Sockets
I.TU.04	Traffic Filter Rules
I.TU.05	Traffic Filter Sessions
I.TU.06	Traffic Mangler
I.TU.07	Implementation Version
I.TU.08	UDP Log

7.4.2 TCP

Table 7.3 shows the information that should be preserved for the TCP protocol as defined in RFC 793 [67]. The only information identified but not appearing in the table is the Segment Payload (I.TT.02). It is expected that the Segment Payload is going to trigger events on the receiving host in the higher network layers and does not need to be retained separately.

The training and verification phase showed the importance of retaining the Segment Header (I.TT.01) field to be able to retrace several scenarios. It also shows the necessity of retaining the corresponding Packet Header (I.NI.01) to determine the access port of the attack.

Table 7.3: Information recommended to be retained for TCP.

Information	Description
I.TT.01	Segment Header
I.TT.03	Active Sockets
I.TT.04	Traffic Filter Rules
I.TT.05	Traffic Filter Sessions
I.TT.06	Traffic Mangler
I.TT.07	Implementation Version
I.TT.08	TCP Log

7.5 Summary

This chapter discussed several network scenarios of interest to a forensic investigator and identified the information necessary to retrace network scenarios on the transport layer. With this information retained a successful investigation of a network incident is more likely.

Even though the network scenarios in this chapter are on a higher layer than the IP protocol, retracing them leads to the same conclusion that the ability to retrace IP address source spoofing is crucial for retracing network scenarios. In addition, the chapter shows the importance of the UDP header for retracing attacks using malicious UDP headers and the TCP header for retracing network scenarios attacking the TCP retransmission algorithms.

The next chapter looks into scenarios related to transport layer protocol standards.

Chapter 8

Extraction and Refining of Forensic Information from Application Layer Protocols

8.1 Introduction

The extraction and refining of forensic information on the transport layer was discussed in Chapter 7. This chapter focuses on application layer protocols and discusses network scenarios that are especially interesting from a network forensics point of view.

An experimental approach was taken where network scenarios based on the application layer protocols were run in lab environment and retraced to identify relevant forensic information. This chapter covers the application layer protocols DHCPv6 and DNS.

Section 8.2 presents the network scenarios used to train the set of information and to verify the recommended set of information. Some relevant scenarios are discussed in details. Section 8.3 assesses the quality of the extracted and refined information and Section 8.4 presents the set of significant information recommended to be retained on the application layer in order to achieve a forensically ready network.

8.2 Network Scenarios

To extract the information from the application layer protocols the scenarios presented in Table 8.1 are used. Each scenario represents a sequence of events on the application layer with a malicious or genuine intent. The scenarios are randomly assigned to be used for the training of the set of information or the verification of the recommended set of information.

Table 8.1: Application layer scenarios with their intent (M=malicious, G=genuine) and purpose (T=training, V=verification).

Scenario	Description	Intent	Purpose
S.AG.01	Resolve a Name from a Local Zone	A	T
S.AG.02	Resolve a Name from a Remote Zone	A	T
S.AG.03	Request Zone Transfer	A	V
S.AG.04	DHCP Information Request	A	T
S.AG.05	DHCP Address Request	A	V
S.AM.01	DNS Reply Spoofing	A	T
S.AM.02	DNS Cache Poisoning	A	T
S.AM.03	DNS Reflector Attack	A	T
S.AM.04	Rogue DHCP Server	A	V
S.AM.05	Spoof DHCP Release Message	A	T
S.AM.06	DoS DHCP Server	A	V
S.AM.07	DoS DHCP Client after Solicit Message	A	T
S.AM.08	DoS DHCP Client after Renew Message	A	T

The following seven subsections discuss the retracing of selected scenarios that were either not retraceable in the training or verification phase, or they are otherwise interesting for a forensic investigation. The detailed description and discussion of all scenarios can be found in Appendix B.6.

8.2.1 Resolving a Name from a Local Zone

To retrace a DNS client that resolves a hostname from a local zone (S.AG.01), the DNS Server (I.AN.01), the DNS Server Log (I.AN.07), and the DNS Server Cache (I.AN.06) are needed. The DNS Server identifies the system that handles the query and allows determining the response based on the server's configuration. The DNS Server Log can be used to ascertain the exact timestamp indicating when the query was received, the IP address of the sender of the query, the content of the query, whether the Recursion Desired flag was set and on what IP address the query was received. The log does not show the answer of the DNS server. The DNS Server Cache can be used to determine the answer. If the resource record was in the cache when the client sent the query, the cache entry is the answer that the client received. If the resource record is in the cache after the query is received, the cache entry is the answer the client received. If the resource record is not in the cache after the query was received, the client received a reply without an answer (not found). The access port of the DNS client can be retraced based on the IP address in the DNS Server Log (see Subsection 6.2.1).

Logging the reply sent to the client in the DNS Server Log would simplify the retracing of this scenario. In addition, the actual data exchanged between the client and server (I.AN.01) could also be used to retrace the scenario and could serve as a secondary source.

8.2.2 DHCP Information Request

To retrace a client requesting stateless configuration information from a DHCP server (S.AG.04), the DHCP Server Log (I.AH.11) and the DHCP Server Configuration (I.AH.04) are very useful. The DHCP Server Log shows the messages received from the client including a timestamp, the source IP and port where they were sent from, and the transaction ID. The log also contains the kind of messages the server sent back. However, the log does not contain the content of the message the server sent to the client. This needs to be reconstructed based on the configuration of the DHCP server.

Logging the actual content of the message that was sent to the DHCP client would simplify the retracing of this scenario considerably. Currently, it is only possible to show that a certain kind of message was sent back to a client and not the specific content of the message. In addition, the actual data exchanged between the client and server (I.AH.01) could also be used to retrace the scenario and could serve as a secondary source.

8.2.3 DHCP Address Request

The four messages of a DHCP address assignment (S.AG.05) can be retraced with the DHCP Server Log (I.AH.11). It contains the timestamps and the kind of message that was received or sent and the IP and port the message was received from or sent to. The transaction ID of the received messages is also in the log. The first time the server assigns an IP address to the client, the address appears in the log. Unfortunately, the message stating that the server has assigned an IP address to a client can only by matched to a client's Solicit message by time proximity. There is no other way to identify what IP address was assigned to a client with a certain link-local address or client ID. The Binding Database (I.AH.07) can be used to determine what IP address or addresses were assigned to each client ID at a specific point in time. Again, it is only possible to match the address in the Binding Database with the log entries about messages sent and received from a specific link-local address by time proximity.

An additional clue could be that the IP link-local address and the client ID (I.AH.08) are both often generated based on the data-link layer address of the client. As a result, it is likely that the log entry with the data-link layer address can be correlated with the address in the Binding Database based on the data-link layer address that was used to generate both. However, this is not a necessity of the protocol but merely a coincidence.

The scenario is retraceable with the current set of significant information. However, because of the relative weak link between the Server Log and the Binding Database, the actual DHCP Traffic (I.AH.01) is needed in order to retrace the address assigning reliably.

8.2.4 DNS Cache Poisoning

DNS Cache Poisoning (S.AM.02) works by appending malicious entries to genuine replies. This is hard to detect and retrace because parts of the DNS query and reply are genuine while other parts are not. The DNS queries sent by the DNS recursive resolver have to be matched against the replies received from the different name servers to detect the cache poisoning. If a reply contains a resource record that was not asked for in the query, it is possible that a cache poisoning attack is in progress. However, related replies are not malicious and are even encouraged by the protocol standard. A related reply is, for example, an additional AAAA record if the original query resolved to a CNAME.

Retracing the scenario is possible with the DNS Traffic (I.AN.01) including Datagram Header (I.TU.01) and Packet Header (I.NI.01). The sender of the malicious replies can be identified with the IP address in the packet header of the reply. The IP address in the reply could be spoofed but it is not very likely because the DNS server needs to be reachable in order for the poisoning attack to work. If the DNS server spoofed the source IP of the packet carrying the reply, it could not be matched with a query and it would just be dropped.

8.2.5 DNS Reflector Attack

A DNS Reflector Attack (S.AM.03) can be detected and retraced with the DNS Traffic (I.AN.01). A query with a group of resource records that are highly unrelated to each other and that result in large replies are typical for the attack. The attack is only successful if the attacker is able to spoof the IP address of the DNS query that initiates the attack. Therefore, it must also be possible to detect and retrace IP address spoofing (see Section 6.2.4) to retrace this attack.

Both this and the previous scenarios show the importance of retaining the DNS Traffic (I.AN.01) for the successful retracing of DNS related attacks.

8.2.6 Rogue DHCP Server

A rogue DHCP server (S.AM.04) can be detected and retraced with the DHCP Traffic (I.AH.01). The access port of the rogue DHCP server can be retraced with the source IP address (link-local) the server uses to send its messages.

Any DHCP server would also be in a good position to detect the attack. This is because the DHCP client sends all messages (Solicit, Request) to the All-DHCP-Server multicast group. That means that all DHCP servers receive the Request message the client sends to the rogue DHCP server. If the genuine DHCP server holds a list of the server IDs of all genuine DHCP servers in the network, it could create at least a log message stating that an unexpected DHCP message was received. When the attacker decides to spoof the DHCP server ID with the one of the genuine DHCP server, at least this server would be able to detect that the Request message the client sends does not match the Advertise message previously sent by the server if the server sent an Advertise message at all. However, this allows only retracing the attack if the attacker does not spoof the source IP address in the offers. In that case, the actual DHCP Traffic is needed in order to retrace the access port of the attacker.

8.2.7 DoS DHCP Client after Solicit Message

The DoS attack with a spoofed Advertise message with the NoAddrAvail option set (S.AM.07)is easy to detect with the DHCP Traffic (I.AH.01). Two Advertise messages with the same server and client ID sent within a few seconds of each other indicate this scenario is in progress. Retracing this scenario is only possible if IP address spoofing is detectable and retraceable (see Section 6.2.4).

Both this and the previous scenario show the importance of retaining the DHCP Traffic (I.AH.01) to be able to retrace DHCP attacks.

8.3 Verification

The quality of the set of significant information after the training phase was assured with the set of verification network scenarios (see Table 8.1). All four verification scenarios could be retraced with the set of significant information after the training phase.

8.4 Results

The following two subsections present the results from the extraction and refining process of forensic information from application layer protocols. Each subsection shows the forensically relevant information for one application layer protocol. The information should be retrained and changes in the information should be tracked. Information in the context of this research is not just the current value of a specific piece of information, for example, the value of a state variable, but also the time-period when the value was current. Retaining this information makes it more likely to be able to retrace network incidents on the data-link layer.

8.4.1 DNS

Table 8.2 shows the information that should be preserved for the DNS protocol as defined in RFC 1033 [105], RFC 1034 [73], RFC 1035 [74], RFC 1123 [77] pages 72 to 86, RFC 2181 [75], RFC 3596 [85], RFC 3901 [89], RFC 4343 [76] and RFC 4472 [88]. The only information not included in the table is the DHCP Client Cache (I.AN.08). The reason for this exclusion is because client caching in not implemented in the lab network.

The training and verification phase showed the importance of retaining the DNS Traffic (I.AN.01) for the successful retracing of many scenarios. It also shows the necessity of retaining the corresponding Datagram Header (I.TU.01) and Packet Header (I.NI.01) to determine the access port of the attack and to retrace scenarios that depend on source IP address spoofing.

Table 8.2: Information recommended to be retained for the DNS protocol.

Information	Description
I.AN.01	DNS Traffic
I.AN.02	Traffic Filter
I.AN.03	Traffic Mangler
I.AN.04	DNS Server
I.AN.05	DNS Client
I.AN.06	DNS Server Cache
I.AN.07	DNS Server Log

8.4.2 DHCP

Table 8.3 shows the information that should be preserved for the DHCP protocol as defined in RFC 3315 [93], RFC 3646 [94] and RFC 3736 [95]. All pieces of information identified in the protocol standards are also part of the set of information recommended to be retained.

The training and verification phase showed the importance of retaining the DHCP Traffic (I.AN.01) for the retracing of many scenarios. It also shows the necessity of retaining the corresponding Datagram Header (I.TU.01) and Packet Header (I.NI.01) to determine the access port of the attack or to detect and retrace scenarios that depend on source IP address spoofing.

Table 8.3: Information recommended to be retained for the DHCPv6.

Information	Description
I.AH.01	DHCP Traffic
I.AH.02	Traffic Filter
I.AH.03	Traffic Mangler
I.AH.04	DHCP Server
I.AH.05	DHCP Relay Agent
I.AH.06	DHCP Client
I.AH.07	Binding Database
I.AH.08	DHCP Unique Identifiers
I.AH.09	Identity Association ID
I.AH.11	DHCP Server Log
I.AH.12	DHCP Relay Agent Log

8.5 Summary

This chapter discussed several network scenarios of interest to a forensic investigator and identified the information necessary to retrace network scenarios on the application layer. With this information retained, a successful investigation of a network incident is more likely.

While the previous chapters assumed that the payload of a protocol data unit triggers events on the next higher layer, this chapter shows that the events triggered by the payload are not logged in enough detail. Therefore, it is concluded that all network traffic should be retained in order to achieve network forensic readiness.

The next chapter discusses the findings of this research, its limitations and practical implications and also suggests directions for further research in the field.

Chapter 9

Practical Implications and Future Work

9.1 Introduction

The extraction and refining of forensic information on each network layer was discussed in Chapter 5 to 8. This chapter draws conclusion from the results shown in the previous chapters, presents the contribution of this thesis and answers the research questions. Further, it suggests how to apply the findings of this thesis in practice. The chapter discusses some of the main challenges of network forensic readiness that emerged during this research and formulates some further research directions.

Section 9.2 presents the findings of this research and discusses how to apply them in a real network while Section 9.3 points out its limitations. Section 9.4 indicates directions for further research in the area.

9.2 Implementation of Forensically Ready Network

This thesis answers the two research questions "What key information (e.g. states, mappings, events) is required to be retained in an IPv6 network to successfully investigate incidents?" and "How is this information stored in an IPv6 network?". The pieces of information that should be retained for a successful investigation are listed in the Tables 5.2 - 5.4, Tables 6.3 - 6.8, Tables 7.2 - 7.3 and Tables 8.2 - 8.3. The tables list a total of 158 pieces of information, which is only marginally less (8 %) than the set of available information. It is concluded that there is no set of relevant information that is significantly smaller than the set of all available information. Any set smaller than the presented one would not allow retracing all network scenarios analysed in this thesis. While this research has not succeeded in reducing the amount of information significantly, it has proved experimentally what information is necessary to be retained in order to achieve network forensic readiness. A detailed description of each piece of information, where the information can be found and how it can be extracted is provided in Appendix A.

9.2.1 Implementing Means to Collect Forensically Relevant Information

A practitioner interested in creating a forensically ready network can apply the findings of this research as follows. The first step is to identify all the network protocols active in the subject network. Based on the active protocols, the relevant tables (Tables 5.2 - 5.4, Tables 6.3 - 6.8, Tables 7.2 - 7.3 and Tables 8.2 - 8.3) are selected and the practitioner implements means to collect and retain the listed information in a forensically sound manner.

Because of different operating systems or operational constraints, the practitioner might not be able to collect all information suggested in this study. In that case, the practitioner has to consider whether the information in question is redundant and provided by another piece of information, or some other means exist to extract the same information from elsewhere, or the risk of not collecting the information is acceptable.

While the specific elements of the device configuration are covered in the tables mentioned above, it is often easier to retrieve the whole configuration of a network device. The current configuration of network devices (switches, routers) is crucial for the successful retracing of network scenarios. Therefore, a tight configuration change management is highly recommended when aiming for forensic readiness. Configuration management allows also detecting when an attacker succeeds in taking control over a network devices and starts changing the configuration. The configuration tracking should not just focus on the device settings but also on the physical configuration of the network and how the different devices are physically connected to each other.

9.2.2 Capture Network Traffic

The research has found that all protocol headers are crucial to retain in order to achieve network forensic readiness. This means essentially that all network traffic including all protocol headers should be captured and retained. However, the range of researched application layer protocols is too limited to come to a conclusive result. It might be that some application layer daemons log in much more detail and in a way that would suffice as evidence for a forensic investigation. In that case, the full network traffic, especially the application layer payload, would not be needed and would be redundant.

On the other hand, capturing network data has the advantage that it covers every new type of attack and any new protocols. Therefore, implementing a system that allows capturing all network traffic is part of creating a forensically ready network. While it is simple to install and run a network traffic capturing device, the placement of such a device and the handling of the data volume it creates is non trivial. A capturing device could be placed in front of each default router facing the access network. The device would be in a good position to capture all traffic that leaves or enters the network. However, the device cannot capture traffic that is exchanged between nodes in the same access network. In addition, a capturing device for each access network is needed.

To reduce the number of capturing devices, a device could be placed in the core of the network where most of the traffic passes through. This solution also suffers from the same problem as the previous one: traffic between hosts in the same access network is not visible to the system. It is actually worse than the previous solution because the system does not even see the traffic between access networks that are on the same branch relative to the core (traffic between these access networks does not flow through the core).

Traffic enters an access network in one of two ways. Either it is created by a node connected to the access network or it is addressed to a node in the access network and enters it through a router. An access network can also be seen as all switch ports connected to nodes (hosts or routers) with the same access VLAN configuration. An ideal solution would be able to capture all traffic that enters an access network through such a port and forwards it to a central collection device. Such setup could be implemented with the components used in the test network and with additional software or with features offered some commercial switches (e.g. Cisco VLAN Switched Port Analyser [106]). However, both setups would not allow determining at which port a specific frame in the captured traffic entered the network. This is important to identify the sender of network traffic. While the requirements are clear, the means for implementing traffic capturing in a satisfying way are not readily available. Section 9.4.5 discusses potential further research to solve the problem.

Capturing and storing large amount of network traffic is a big challenge for the implementation of a forensically ready network. The most promising approach seems to be a ring-buffer system as proposed by Maier et al. [20] that stores only the beginning of traffic sessions combined with a system that gradually summarises information as it gets older [19]. The second system would take information that falls out of the ring-buffer as its input. These two systems allow reducing the volume needed to retain the network traffic. Section 9.4.3 suggests further research to reduce the volume of network traffic captures.

9.2.3 Network Security and Network Forensics

Network security and network forensics are closely interrelated. Network security allows detecting attacks and fending them off. Network forensics, on the other hand, is used to analyse the attacks and to retrace their origin. Strong network security is beneficial for a forensically ready network and it improves the credibility of the collected evidence.

This research underlines the problem of IP address spoofing for a network forensic investigation. Many of the network scenarios used in this research are not retraceable if it is not possible to detect and retrace IP address spoofing. To be able to use the source IP address in a packet for identifying a sender, the network needs some defence against IP address spoofing (e.g. IP source guard [29], RPF filtering).

Further, it must be clear what IP addresses cannot be verified. For example, in a corporate network it is possible to implement strong measures to make it almost impossible to spoof source IP addresses (see Section 6.2.4). Consequently, IP addresses within the address range of the corporation can be reliably linked to an access port and IP addresses within the corporation's IP range can be used to identify the sender of network traffic. On the other hand, if the sender is located in the Internet the source IP address is not a reliable information to identify the sender.

Another important network security measure to sustain the credibility of the evidence is to appropriately protect devices that process network traffic. If an attacker manages to get access to a router or a switch, it cannot be excluded that the attacker manipulates the device in a way that changes the network traffic passing through the device. As a consequence, network traffic might not be reliable evidence anymore. Therefore, it is important to ensure the integrity of network devices.

Additionally, network security devices typically allow creating detailed logs of their activities. For example, a firewall can usually be configured to log the forming and tear-down of transport layer (UDP, TCP) sessions. Often it is also possible to record additional information as the amount of data exchanged, which site initiated and which site closed the connection. Other devices that might have similar abilities are IDS and proxy systems.

This research also revealed that the logging in several protocol implementations is inadequate for a forensic investigation. Either some specific events are not logged at all or the log does not contain sufficient details to satisfy a forensic investigation. In some cases, all the necessary events are logged but it is not possible to correlate them, for example assign them to a specific node. The correct configuration of the logging functions of any application layer daemon and the selection of an appropriate level of detail for the logging is important for achieving network forensic readiness.

9.2.4 Incident Response

In this research it was possible to retrace 31 of 33 verification scenarios (94 %) with the recommended set of significant information. However, it is unlikely to achieve the same result in a real network because of a wider spectrum of network protocols, the possibility that not all information can be collected in the specific environment, changes in the environment and errors in the collection process. In addition, new types of attacks emerge frequently and might prove the set of retained information insufficient.

To address this issue, this research proposes an extension to the NIST forensic life-cycle [107] with a dedicated Optimisation phase. An Optimisation phase introduced after the Reporting phase would allow to focus on analysing the collection phases and identifing missing information or problems with the Collection process. This information can then be added to the set of information that is proactively collected or the collection process can be improved adequately. While the Reporting phase in the current life-cycle already is used to "identify any problems that may need to be remedied, such as policy shortcomings or procedural errors" [107, p. 3-7], the author believes that a separate Optimisation phase should be added to the life-cycle to underline the importance of these feedback for the forensic readiness of a network.

Practitioner should handle network incidents according to the proposed extended forensic life-cycle (see Figure 9.1). A way to improve the initial proactive evidence collection rapidly is to investigate not only successful attacks but also attacks that were prevented by security measures. Using the extended forensic life-cycle for practice runs helps to cultivate a forensically ready network faster.

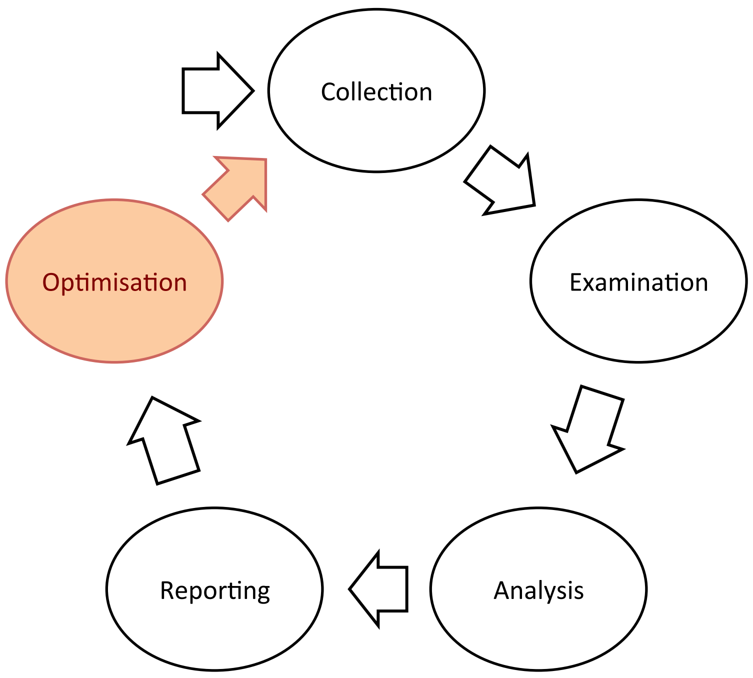

Figure 9.1: The extended forensic life-cycle for a forensically ready network with the new Optimisation phase.

9.3 Limitations of this Study

9.3.1 Methodology

The chosen methodology does not allow reducing the set of significant information. Therefore, the set is expanded with every new scenario or attack analysed. On the other hand, every piece of information was added to retrace at least one network scenario. Therefore, the set could only be reduced if some network scenarios are not relevant anymore.

The information identified in the protocol standards was categorised based on the potential usefulness for an investigation and the difficulty to collect the information. Based on this categorisation the first set of significant information was created. The categorisation, especially for the usefulness, is empirical and can be questioned. It is possible to argue that running the scenarios to identify the necessary information for retracing the scenario is sufficient and the first step is not required. This first step might have led to a slightly larger set of recommended information to be retained than is actually necessary. On the other hand, it can also be argued that the larger set does not harm the objective of forensic readiness and the possible redundancy would improve the credibility of the collected evidence.

It is possible that, while compiling the list of network scenarios on the different network layers, some scenarios were missed. Further, some information could have been missed while compiling the list of available information for each network protocol. The extension of the forensic life-cycle should counteract the consequences of the possible omissions by adding missing information when the omission is recognised during an investigation.

Grouping the network scenarios based on the network layer and randomly divide them into training and verification scenarios could be problematic as it leads to situations where all scenarios of one network protocol end up exclusively in the training or the verification group. For example, all VLAN and UDP scenarios are in the training group and none is in the verification group. In hindsight, grouping on the protocol level might have been a better choice. However, the grouping does not have a consequence for the actual set of information that is recommended to be retained. However, the verification process is less significant.

9.3.2 The Definition of Traceability

This thesis defined traceability as being able to reconstruct the sequence of events and to determine the access port where the network traffic subject of the investigation entered the network. In a real network, however, it would be necessary to link the sequence of events and the access port to a particular person. This could be the objective of further research projects.

9.3.3 Lack of Internet Connectivity

The network lab used in this research does not provide Internet connectivity. Consequently, the scenarios do not cover cases where one party (attacker or victim) is located in the Internet and no information can be collected from it. This is relevant for all scenarios on layers above the data-link layer.

9.4 Further Research Directions

This section discusses possible further research directions other than the obvious extension of the covered network protocols and adding IPv4 and Internet connectivity to the analysis.

9.4.1 Means to Collect the Recommended Information

This research has focused on identifying which information is crucial to collect but has not investigated the means to collect it. For a forensically ready network, it is also essential that the evidence is collected and retained in a forensically sound manner.

Some further research could analyse the means for collecting the information identified in this research. The objective would be to collect the information in a way that would ensure it can be used in a court of law. This means, amongst other things, that it must be possible to show the process, as well as the time and the place of the collection. Further, adequate measures to protect the integrity of the collected information need to be implemented and the chain of evidence needs to stay intact.

9.4.2 Prioritise the Recommended Information

The current set of forensically significant information simply recommends what information should be retained. This may lead to an overwhelming amount of information that needs to be retained to achieve forensic readiness. Consequently, a practitioner might face the problem that the resources to retain all the recommended information are not available. Therefore, a prioritised list of recommended information might need to be developed.

A way to come up with such a grading could be based on the damage potential of the scenarios. Damage could be categorised based on loss of confidentiality, integrity and availability. An algorithm would need to be developed that allows ordering the scenarios from most to least damaging. The information needed to retrace the most damaging scenarios, would be the information that should be retained first.

As damage is highly subjective, it could be sensible to align such a process with the risk management. The result could be a risk management process that not only suggests security measures but also which information should be retained related to the identified risks. Consequently, means to retain forensic information related to high risks would be implemented first.

9.4.3 Reducing the Volume of Network Traffic Captures

The result of this research suggests capturing and retaining traffic from several points in the network (see Section 9.2.2). Consequently, the same packet might be captured multiple times at different points in the network. In addition, some network protocols (e.g. ND, RSTP) transmit the same or very similar packets regularly. Both, double capturing and retransmitting similar packets create redundant information.

A useful contribution would concentrate on optimising the storage of network traffic captures that contain partly the same traffic. One option might be to retain only the first occurrence of a packet and just store the timestamp and the location where the packet was seen afterwards. Such a schema might also need to save the different data-link layer headers used to transport the same packet at different locations and the potential small differences between the packets captured at different locations. (e.g. the different Hop Limit of every packet). Devices that change packets on the way (e.g. through network address translation) and the identification of the original and the changed packet as the same packet, poses some challenges.

Additionally, the captured network traffic could be correlated with other information. Network traffic often triggers changes in state tables; for example, NA messages change the Neighbour Cache of the receiving node. A future research project could focus on finding and correlating such information. Software that is able to correlate such information and present it in an understandable user interface that shows the information and the correlation, would facilitate forensic investigations.

9.4.4 Forensic Readiness and Network Security Measures

Network security measures are used to protect data against manipulation, to verify the authenticity of data, or to authenticate users or machines (IEEE 802.1AE [27], IEEE 802.1X [108], IPSec [46, 47], DNS Security Extensions [78]). Others protect the link between data-link layer and network addresses (SEND [49], ARP inspection, NS/NA inspection [103]) or ensure that dynamic address assignments are not exploited (DHCP snooping [101], RA Guard [103]). While most of these security measures improve the network forensic readiness, some also hinder investigations (e.g. encryption). A research project could focus on analysing these security measures and protocols regarding the consequences for the network forensic readiness of a network. The following questions should be answered: In what way do they improve the forensic readiness? How do they increase the credibility of the collected evidence? In what scenarios do they facilitate the investigation and in what scenarios do they hinder the investigation?

9.4.5 Enhanced Capabilities for Capturing Network Traffic

This research highlighted the importance of capturing network traffic. However, capturing network traffic for a whole network segment is not trivial. While some commercial switch providers (e.g. Cisco VLAN Switched Port Analyser [106]) offer functionality to capture network traffic for a whole network segment, they do not necessarily fulfil the requirements of network forensics.

A traffic capture system would need to be able to capture all network traffic entering a network segment and would need to have the ability to name the access port where a specific frame entered the network. Furthermore, a system that is able to digitally sign a captured frame to ensure its integrity would be preferable. The system might extend the known capture library Libpcap [109] to support carrying extra meta-information such as the access port where the traffic entered the network, the system that captured the traffic and the digital signature of the traffic. This could build the base of such a system. A challenge would be ensureing that captured traffic is not captured again when it is transported to a central collection node. If this is not handled properly, an endless loop could be the consequence.

Such a system could not only be used to support network forensics and to improve the network forensic readiness. The additional information would also help troubleshooting network problems and could be used by network security systems such as IDS.

9.4.6 Verification of Data-Link Layer and Network Layer Addresses

Software could be developed that traces NS and NA messages to create a database of access port, data-link layer address, and network layer addresses as shown in Table 6.2. If the software has access to network traffic with the access port information as described in the previous subsection, the software could use the information to add the corresponding port to the database. When the access port information is not available, the software could use the Filter Database of switches in the corresponding network segment to add the access port information to the database. The database alone would be very helpful for any network investigation. It allows determining quickly and easily all related information if one of the elements in the database is known.

The software NDPMon [110], developed by the university of Lorraine, already does the NS/NA monitoring part and informs when irregularities are detected. It might be possible to extend the software with the suggested features or at least, reuse parts of it.

This database could then be used to verify network traffic. Software with access to the network traffic could verify that the data-link layer and network layer address pairs match the values in the database. If the software has access to the access port information, the software can also verify that the frames were received on the correct port and the frames and the packets inside use the correct source addresses. This would make it very hard for an attacker to spoof a data-link or network layer address and greatly improve the reliability of using data-link and network layer addresses for identifing the source of network traffic. As most of the information necessary to implement the software is accessible on network switches. Therefore, it might be a viable option to implement the software directly into network switches.

9.5 Summary

This chapter summarised and discussed the results obtained in this research and its practical implications. Further, the limitations of the research were pointed out and further research directions in the area were identified.

Chapter 10

Conclusions

A forensically ready network has the capability to provide sound evidence to facilitate forensic investigations. Forensic readiness is achieved through proactive collection of crucial evidence. In case of an incident, this allows concentrating on damage repair while not jeopardising a proper forensic investigation at a later point. While the concept and benefits of a forensically ready network are widely known and understood, the exact set of evidence that should be collected proactively is unknown. This thesis has focused on identifying the crucial forensic evidence in an IPv6 network in order to achieve network forensic readiness and where the evidence can be located.

The research started with reviewing contemporary literature regarding network forensics, forensic readiness, and network forensic readiness in Chapter 2. The lack of recommendations in current literature about what information should be retained to achieve forensic readiness became apparent and a solid understanding of the current state of the research field was gained.

Chapter 3 concentrated on reviewing network protocols relevant to the scope of the research. It identified the most important protocol standards and known network attacks. This was necessary to understand the kind of incidents a forensically ready network needs to be able to handle and what kind of network scenarios need to be retraceable.

The research methodology was outlined in Chapter 4. An experimental approach was chosen to identify and refine important forensic information in network protocols. A hypothesis was formulated that a certain set of information is enough to retrace network scenarios. This hypothesis was then tested with a number of experiments. Each experiment involved running a genuine or malicious network scenario in a lab network and retracing it with the proposed set of significant information. If the retracing was not possible, the set of significant information was expanded accordingly and the experiment was conducted again. The network protocols were grouped by network layer and processed bottom-up in the order: data-link, network, transport, and application layer. The results of the experiments were reported and discussed in Chapter 5, 6, 7, and 8. Network scenarios especially interesting for a network forensic investigation were discussed in detail and the final set of information recommended to be retained was presented for each network layer.

The answers to the two research questions and the practical implications of this research were presented in Chapter 9. This thesis identified a set of forensically significant information that should be retained to achieve network forensic readiness. The set also allows locating the information. A practitioner that implements means to collect the recommended information proactively and in forensically sound manner, creates a forensically ready network and improves the likelihood of a successful forensic investigation considerably. Further, this thesis proposed an extension of the NIST forensic life-cycle for forensically ready networks with an Optimisation phase (Figure 9.1). The Optimisation phase analyses the collection of evidence for the current investigation as the last step in a forensic investigation. It identifies missing evidence and weaknesses in the collection, and finally suggests improvements in the proactive collection of evidence. While part of this step is already covered in the existing Reporting phase, the author believes that the Optimisation phase should added as a distinct phase to highlight its importance for a forensically ready network.

Chapter 9 also outlined design and deployment strategies for implementing a forensically ready network and recommendations were made for mastering the implementation. Three key points were made. Firstly, network traffic is a crucial factor in network forensic investigation. Without having access to the actual network traffic, many network scenarios are not retraceable. The traffic is crucial to show the means and the malicious intent of many network attacks. The network traffic is especially important because the logging of the two application layer daemons studied is insufficient for the purpose of a forensic investigation. Secondly, IP address spoofing is a major problem while retracing network scenarios. Without being able to detect and retrace IP address spoofing many network forensic investigations fail to identify the host that sent the malicious traffic and, in consequence, the culprit that initiated the malicious traffic remains undetected. The research also outlines software that could simplify detecting IP address spoofing and facilitate forensic investigations by having corresponding data-link layer and network layer information readily available. Thirdly, capturing network traffic in a forensically sound manner is challenging. The placement of the capturing device and the potential huge volume of network traffic are some of the identified issues. Requirements for capturing network traffic and potential solutions like ring buffer systems were outlined.

The chapter finished with suggestions for further research in the field. To give a practitioner a guideline where to start with implementing means for collecting the recommended information, it is suggested to prioritise the list of information. Another line of research could focus on improving capabilities for capturing network traffic for a complete network segment. Enhancing the captured network traffic with additional meta-information and capturing in a forensically sound manner are other potential lines of further research.

This thesis shows that a forensically ready network is not easy to achieve and the research gaps are still many. On the other hand, a forensically ready network is a worthwhile objective and the contributions of this research facilitate attaining it.

References

[1] S. Ewing and J. Thomas, "The Internet in Australia," ARC Centre of Excellence for Creative Industries and Innovation, Tech. Rep., 2012. [Online]. Available: http://www.cci.edu.au/projects/digital-futures

[2] B. Endicott-Popovsky, D. A. Frincke, and C. A. Taylor, "A Theoretical Framework for Organizational Network Forensic Readiness," *Journal of Computers*, vol. 2, no. 3, pp. 1–11, 2007.

[3] S. Mukkamala and A. H. Sung, "Identifying Significant Features for Network Forensic Analysis Using Artificial Intelligent Techniques," *Intl. Journal of Digital Evidence*, vol. 1, p. 2003, 2003.

[4] A. Almulhem and I. Traoré, "Experience with Engineering a Network Forensics System," in *Information Networking. Convergence in Broadband and Mobile Networking*, ser. Lecture Notes in Computer Science. Springer, 2005, vol. 3391, pp. 62–71.

[5] S. Garfinkel, "Network Forensics: Tapping the Internet," 2002. [Online]. Available: http://www.oreillynet.com/pub/a/network/2002/04/26/nettap.html

[6] V. Corey, C. Peterman, S. Shearin, M. S. Greenberg, and J. V. Bokkelen, "Network Forensics Analysis," *IEEE Internet Computing*, vol. 6, no. 6, pp. 60–66, 2002.

[7] W. Ren and H. Jin, "Modeling the Network Forensics Behaviors," in *Proceedings of the 1st International Conference on Security and Privacy for Emerging Areas in Communication Networks*, 2005, pp. 1–8.

[8] J. Tan, "Forensic Readiness," Stake Organization, Tech. Rep., 2001. [Online]. Available: http://www.arcert.gov.ar/webs/textos/forensic_readiness.pdf

[9] R. Rowlingson, "A Ten Step Process for Forensic Readiness," *International Journal of Digital Evidence*, vol. 2, no. 3, pp. 1–28, 2004.

[10] B. Endicott-Popovsky and D. A. Frincke, "Embedding Hercule Poirot in Networks: Addressing Inefficiencies in Digital Forensic Investigations," in *Foundations of Augmented Cognition*, ser. Lecture Notes in Computer Science, D. Schmorrow and L. Reeves, Eds. Springer

Berlin / Heidelberg, 2007, vol. 4565, pp. 364–372. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-73216-7_41

[11] B. J. Nikkel, "An introduction to investigating IPv6 networks," *Digital Investigation*, vol. 4, no. 2, pp. 59–67, 2007.

[12] S. Hogg and E. Vyncke, *IPv6 Security*, B. Bartow, Ed. Cisco Press, 2009.

[13] S. Frankel, R. Graveman, J. Pearce, and M. Rooks, "Guidelines for the Secure Deployment of IPv6," National Institute of Standards and Technology, Tech. Rep. Special Publication 800-119, 2010.

[14] R. Hinden and S. Deering, "IP Version 6 Addressing Architecture," RFC 4291 (Draft Standard), Internet Engineering Task Force, Feb. 2006, updated by RFCs 5952, 6052. [Online]. Available: http://www.ietf.org/rfc/rfc4291.txt

[15] M. Van Hauser, "Recent advances in IPv6 insecurities," 2010. [Online]. Available: http://events.ccc.de/congress/2010/Fahrplan/attachments/1808_vh_thc-recent_advances_in_ipv6_insecurities.pdf

[16] "IPv6 Routing Header Vulnerability," Cisco Systems, Tech. Rep., May 2007. [Online]. Available: http://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-20070124-IOS-IPv6

[17] B. J. Nikkel, "Generalizing sources of live network evidence," *Digital Investigation*, vol. 2, no. 3, pp. 193–200, 2005.

[18] E. S. Pilli, R. C. Joshi, and R. Niyogi, "Network forensic frameworks: Survey and research challenges," *Digital Investigation*, vol. 7, no. 1-2, pp. 14–27, 2010.

[19] E. Cooke, A. Myrick, D. Rusek, and F. Jahanian, "Resource-aware multi-format network security data storage," in *Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense*, ser. LSAD '06. New York, NY, USA: ACM, 2006, pp. 177–184. [Online]. Available: http://doi.acm.org/10.1145/1162666.1162677

[20] G. Maier, R. Sommer, H. Dreger, A. Feldmann, V. Paxson, and F. Schneider, "Enriching network security analysis with time travel," in *Proceedings of the ACM SIGCOMM 2008 conference on data communication*, 2008, pp. 183–194.

[21] K. Shanmugasundaram and N. D. Memon, "Network Monitoring for Security and Forensics," in *Information Systems Security*, ser. Lecture Notes in Computer Science. Springer, 2006, vol. 4332, pp. 56–70.

[22] A. K. Kaushik, E. S. Pilli, and R. C. Joshi, "Network Forensic Analysis by Correlation of Attacks with Network Attributes," in *ICT*, ser.

Communications in Computer and Information Science, V. V. Das, R. Vijayakumar, K. G. Srinivasa, H. A. Aboalsamh, M. Hammoudeh, V. Salmani, D. K. Tyagi, A. Mohapatra, B. Jaysimha, E. Ambikairajah, and J. M. Blackledge, Eds., vol. 101. Springer, 2010, pp. 124–128.

[23] E. S. Pilli, R. C. Joshi, and R. Niyogi, "Data reduction by identification and correlation of TCP/IP attack attributes for network forensics," in *Proceedings of the ICWET '11 International Conference & Workshop on Emerging Trends in Technology*, 2011, pp. 276–283.

[24] A. K. Kaushik and R. C. Joshi, "Network Forensic System for ICMP Attacks," *International Journal of Computer Applications*, vol. 2, no. 3, pp. 14–21, May 2010.

[25] Y. Tang and T. Daniels, "A Simple Framework for Distributed Forensics," in *Distributed Computing Systems Workshops, 2005. 25th IEEE International Conference on*, June 2005, pp. 163 – 169.

[26] IEEE Std 802.3, "IEEE Standard for Information Technology - Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks - Specific Requirements, Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications," *IEEE Std 802.3-2008*, 2008.

[27] IEEE Std 802.3ab, "IEEE Standard for Information Technology - Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks - Specific Requirements, Supplement to Carrier Sense Multiple Access With Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications - Physical Layer Parameters and Specifications for 1000 Mb/s Operation Over 4-Pair of Category 5 Balanced Copper Cabling, Type 1000BASE-T," *IEEE Std 802.3ab-1999*, 1999.

[28] IEEE Std 802.1D, "IEEE Standard for Local and Metropolitan Area Networks - Media Access Control (MAC) Bridges," *IEEE Std 802.1D-2004 (Revision of IEEE Std 802.1D-1998)*, 2004.

[29] Y. Bhaiji, *Network Security Technologies and Solutions*, 1st ed. Cisco Press, 2008.

[30] H. Altunbasak, S. Krasser, H. L. Owen, J. Grimminger, H.-P. Huth, and J. Sokol, "Securing Layer 2 in Local Area Networks," pp. 699–706, 2005.

[31] K. Lauerman and J. King, "STP MiTM Attack and L2 Mitigation Techniques on the Cisco Catalyst 6500," Cisco Systems, Tech. Rep., 2010. [Online]. Available: http://www.cisco.com/en/US/prod/collateral/switches/ps5718/ps708/white_paper_c11_605972.html

[32] A. Yeung and A. Wong, *Network Infrastructure Security.* Springer, 2009.

[33] IEEE Std 802.1Q, "IEEE Standard for Local and Metropolitan Area Networks - Virtual Bridged Local Area Networks," *IEEE Std 802.1Q, 2003 Edition (Incorporates IEEE Std 802.1Q-1998, IEEE Std 802.1u-2001, IEEE Std 802.1v-2001, and IEEE Std 802.1s-2002)*, 2003.

[34] D. Taylor, "Intrusion Detection FAQ: Are there Vulnerabilites in VLAN Implementations? VLAN Security Test Report." [Online]. Available: http://www.sans.org/security-resources/idfaq/vlan.php

[35] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," RFC 2460 (Draft Standard), Internet Engineering Task Force, Dec. 1998, updated by RFCs 5095, 5722, 5871, 6437. [Online]. Available: http://www.ietf.org/rfc/rfc2460.txt

[36] J. Abley, P. Savola, and G. Neville-Neil, "Deprecation of Type 0 Routing Headers in IPv6," RFC 5095 (Proposed Standard), Internet Engineering Task Force, Dec. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc5095.txt

[37] S. Krishnan, "Handling of Overlapping IPv6 Fragments," RFC 5722 (Proposed Standard), Internet Engineering Task Force, Dec. 2009. [Online]. Available: http://www.ietf.org/rfc/rfc5722.txt

[38] S. Amante, B. Carpenter, S. Jiang, and J. Rajahalme, "IPv6 Flow Label Specification," RFC 6437 (Proposed Standard), Internet Engineering Task Force, Nov. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6437.txt

[39] S. Kawamura and M. Kawashima, "A Recommendation for IPv6 Address Text Representation," RFC 5952 (Proposed Standard), Internet Engineering Task Force, Aug. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5952.txt

[40] J. Loughney, "IPv6 Node Requirements," RFC 4294 (Informational), Internet Engineering Task Force, Apr. 2006, updated by RFC 5095. [Online]. Available: http://www.ietf.org/rfc/rfc4294.txt

[41] S. Thomson, T. Narten, and T. Jinmei, "IPv6 Stateless Address Autoconfiguration," RFC 4862 (Draft Standard), Internet Engineering Task Force, Sep. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4862.txt

[42] D. Johnson, C. Perkins, and J. Arkko, "Mobility Support in IPv6," RFC 3775 (Proposed Standard), Internet Engineering Task Force, Jun. 2004, obsoleted by RFC 6275. [Online]. Available: http://www.ietf.org/rfc/rfc3775.txt

[43] H. Holbrook, B. Cain, and B. Haberman, "Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast,"

RFC 4604 (Proposed Standard), Internet Engineering Task Force, Aug. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4604.txt

[44] E. Nordmark and T. Li, "Threats Relating to IPv6 Multihoming Solutions," RFC 4218 (Informational), Internet Engineering Task Force, Oct. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4218.txt

[45] C. Perkins, D. Johnson, and J. Arkko, "Mobility Support in IPv6," RFC 6275 (Proposed Standard), Internet Engineering Task Force, Jul. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6275.txt

[46] C. Kaufman, P. Hoffman, Y. Nir, and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)," RFC 5996 (Proposed Standard), Internet Engineering Task Force, Sep. 2010, updated by RFC 5998. [Online]. Available: http://www.ietf.org/rfc/rfc5996.txt

[47] S. Kent, "IP Authentication Header," RFC 4302 (Proposed Standard), Internet Engineering Task Force, Dec. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4302.txt

[48] S. Kent, "IP Encapsulating Security Payload (ESP)," RFC 4303 (Proposed Standard), Internet Engineering Task Force, Dec. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4303.txt

[49] J. Arkko, J. Kempf, B. Zill, and P. Nikander, "SEcure Neighbor Discovery (SEND)," RFC 3971 (Proposed Standard), Internet Engineering Task Force, Mar. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc3971.txt

[50] D. Borman, S. Deering, and R. Hinden, "IPv6 Jumbograms," RFC 2675 (Proposed Standard), Internet Engineering Task Force, Aug. 1999. [Online]. Available: http://www.ietf.org/rfc/rfc2675.txt

[51] "IPv6 Security Brief," Cisco Systems, Tech. Rep., 2011. [Online]. Available: (IPv6SecurityBrief)http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6553/white_paper_c11-678658.html

[52] P. Biondi and A. Ebalard, "IPv6 Routing Header Security," 2007. [Online]. Available: http://www.secdev.org/conf/IPv6_RH_security-csw07.pdf

[53] M. Van Hauser, "Attacking the IPv6 Protocol Suite," 2005. [Online]. Available: http://events.ccc.de/congress/2005/fahrplan/attachments/642-vh_thcipv6_attackccc05presentation.pdf

[54] E. Davies and J. Mohacsi, "Recommendations for Filtering ICMPv6 Messages in Firewalls," RFC 4890 (Informational), Internet Engineering Task Force, May 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4890.txt

[55] A. Conta, S. Deering, and M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification," RFC 4443 (Draft Standard), Internet Engineering Task Force, Mar. 2006, updated by RFC 4884. [Online]. Available: http://www.ietf.org/rfc/rfc4443.txt

[56] T. Narten, E. Nordmark, W. Simpson, and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)," RFC 4861 (Draft Standard), Internet Engineering Task Force, Sep. 2007, updated by RFC 5942. [Online]. Available: http://www.ietf.org/rfc/rfc4861.txt

[57] P. Nikander, J. Kempf, and E. Nordmark, "IPv6 Neighbor Discovery (ND) Trust Models and Threats," RFC 3756 (Informational), Internet Engineering Task Force, May 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3756.txt

[58] "CVE-2010-4669," National Vulnerability Database, Common Vulnerabilities and Exposures, 2011. [Online]. Available: http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2010-4669

[59] G. Malkin and R. Minnear, "RIPng for IPv6," RFC 2080 (Proposed Standard), Internet Engineering Task Force, Jan. 1997. [Online]. Available: http://www.ietf.org/rfc/rfc2080.txt

[60] A. Vainshtein and Y. Stein, "Structure-Agnostic Time Division Multiplexing (TDM) over Packet (SAToP)," RFC 4553 (Proposed Standard), Internet Engineering Task Force, Jun. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4553.txt

[61] R. Coltun, D. Ferguson, J. Moy, and A. Lindem, "OSPF for IPv6," RFC 5340 (Proposed Standard), Internet Engineering Task Force, Jul. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5340.txt

[62] M. Gupta and N. Melam, "Authentication/Confidentiality for OSPFv3," RFC 4552 (Proposed Standard), Internet Engineering Task Force, Jun. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4552.txt

[63] E. Nordmark and R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers," RFC 4213 (Proposed Standard), Internet Engineering Task Force, Oct. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4213.txt

[64] D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina, "Generic Routing Encapsulation (GRE)," RFC 2784 (Proposed Standard), Internet Engineering Task Force, Mar. 2000, updated by RFC 2890. [Online]. Available: http://www.ietf.org/rfc/rfc2784.txt

[65] G. Dommety, "Key and Sequence Number Extensions to GRE," RFC 2890 (Proposed Standard), Internet Engineering Task Force, Sep. 2000. [Online]. Available: http://www.ietf.org/rfc/rfc2890.txt

[66] J. Postel, "User Datagram Protocol," RFC 768 (Standard), Internet Engineering Task Force, Aug. 1980. [Online]. Available: http://www.ietf.org/rfc/rfc768.txt

[67] J. Postel, "Transmission Control Protocol," RFC 793 (Standard), Internet Engineering Task Force, Sep. 1981, updated by RFCs 1122, 3168, 6093. [Online]. Available: http://www.ietf.org/rfc/rfc793.txt

[68] V. Paxson, M. Allman, J. Chu, and M. Sargent, "Computing TCP's Retransmission Timer," RFC 6298 (Proposed Standard), Internet Engineering Task Force, Jun. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6298.txt

[69] V. Jacobson, R. Braden, and D. Borman, "TCP Extensions for High Performance," RFC 1323 (Proposed Standard), Internet Engineering Task Force, May 1992. [Online]. Available: http://www.ietf.org/rfc/rfc1323.txt

[70] M. Allman, V. Paxson, and E. Blanton, "TCP Congestion Control," RFC 5681 (Draft Standard), Internet Engineering Task Force, Sep. 2009. [Online]. Available: http://www.ietf.org/rfc/rfc5681.txt

[71] "Security Assessment of the Transmission Control Protocol," Centre for the Protection of National Infrastructure, Tech. Rep., 2009.

[72] V. A. Kumar, P. S. Jayalekshmy, G. K. Patra, and R. P. Thangavelu, "On remote exploitation of TCP sender for low-rate flooding denial-of-service attack," *Comm. Letters.*, vol. 13, no. 1, pp. 46–48, Jan. 2009. [Online]. Available: http://dx.doi.org/10.1109/LCOMM.2009.081555

[73] P. Mockapetris, "Domain names - concepts and facilities," RFC 1034 (Standard), Internet Engineering Task Force, Nov. 1987, updated by RFCs 1101, 1183, 1348, 1876, 1982, 2065, 2181, 2308, 2535, 4033, 4034, 4035, 4343, 4035, 4592, 5936. [Online]. Available: http://www.ietf.org/rfc/rfc1034.txt

[74] P. Mockapetris, "Domain names - implementation and specification," RFC 1035 (Standard), Internet Engineering Task Force, Nov. 1987, updated by RFCs 1101, 1183, 1348, 1876, 1982, 1995, 1996, 2065, 2136, 2181, 2137, 2308, 2535, 2845, 3425, 3658, 4033, 4034, 4035, 4343, 5936, 5966. [Online]. Available: http://www.ietf.org/rfc/rfc1035.txt

[75] R. Elz and R. Bush, "Clarifications to the DNS Specification," RFC 2181 (Proposed Standard), Internet Engineering Task Force, Jul. 1997, updated by RFCs 4035, 2535, 4343, 4033, 4034, 5452. [Online]. Available: http://www.ietf.org/rfc/rfc2181.txt

[76] D. Eastlake 3rd, "Domain Name System (DNS) Case Insensitivity Clarification," RFC 4343 (Proposed Standard), Internet Engineering

Task Force, Jan. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4343.txt

[77] R. Braden, “Requirements for Internet Hosts - Application and Support,” RFC 1123 (Standard), Internet Engineering Task Force, Oct. 1989, updated by RFCs 1349, 2181, 5321, 5966. [Online]. Available: http://www.ietf.org/rfc/rfc1123.txt

[78] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, “DNS Security Introduction and Requirements,” RFC 4033 (Proposed Standard), Internet Engineering Task Force, Mar. 2005, updated by RFC 6014. [Online]. Available: http://www.ietf.org/rfc/rfc4033.txt

[79] P. Vixie, S. Thomson, Y. Rekhter, and J. Bound, “Dynamic Updates in the Domain Name System (DNS UPDATE),” RFC 2136 (Proposed Standard), Internet Engineering Task Force, Apr. 1997, updated by RFCs 3007, 4035, 4033, 4034. [Online]. Available: http://www.ietf.org/rfc/rfc2136.txt

[80] B. Wellington, “Secure Domain Name System (DNS) Dynamic Update,” RFC 3007 (Proposed Standard), Internet Engineering Task Force, Nov. 2000, updated by RFCs 4033, 4034, 4035. [Online]. Available: http://www.ietf.org/rfc/rfc3007.txt

[81] P. Faltstrom, P. Hoffman, and A. Costello, “Internationalizing Domain Names in Applications (IDNA),” RFC 3490 (Proposed Standard), Internet Engineering Task Force, Mar. 2003, obsoleted by RFCs 5890, 5891. [Online]. Available: http://www.ietf.org/rfc/rfc3490.txt

[82] P. Hoffman and M. Blanchet, “Preparation of Internationalized Strings (“stringprep”),” RFC 3454 (Proposed Standard), Internet Engineering Task Force, Dec. 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3454.txt

[83] P. Hoffman and M. Blanchet, “Nameprep: A Stringprep Profile for Internationalized Domain Names (IDN),” RFC 3491 (Proposed Standard), Internet Engineering Task Force, Mar. 2003, obsoleted by RFC 5891. [Online]. Available: http://www.ietf.org/rfc/rfc3491.txt

[84] A. Costello, “Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA),” RFC 3492 (Proposed Standard), Internet Engineering Task Force, Mar. 2003, updated by RFC 5891. [Online]. Available: http://www.ietf.org/rfc/rfc3492.txt

[85] S. Thomson, C. Huitema, V. Ksinant, and M. Souissi, “DNS Extensions to Support IP Version 6,” RFC 3596 (Draft Standard), Internet Engineering Task Force, Oct. 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3596.txt

[86] D. Atkins and R. Austein, "Threat Analysis of the Domain Name System (DNS)," RFC 3833 (Informational), Internet Engineering Task Force, Aug. 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3833.txt

[87] J. Damas and F. Neves, "Preventing Use of Recursive Nameservers in Reflector Attacks," RFC 5358 (Best Current Practice), Internet Engineering Task Force, Oct. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5358.txt

[88] A. Durand, J. Ihren, and P. Savola, "Operational Considerations and Issues with IPv6 DNS," RFC 4472 (Informational), Internet Engineering Task Force, Apr. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4472.txt

[89] A. Durand and J. Ihren, "DNS IPv6 Transport Operational Guidelines," RFC 3901 (Best Current Practice), Internet Engineering Task Force, Sep. 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3901.txt

[90] Y. Morishita and T. Jinmei, "Common Misbehavior Against DNS Queries for IPv6 Addresses," RFC 4074 (Informational), Internet Engineering Task Force, May 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4074.txt

[91] "CVE-2003-1132," National Vulnerability Database, Common Vulnerabilities and Exposures, 2003. [Online]. Available: http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2003-1132

[92] A. Hubert and R. van Mook, "Measures for Making DNS More Resilient against Forged Answers," RFC 5452 (Proposed Standard), Internet Engineering Task Force, Jan. 2009. [Online]. Available: http://www.ietf.org/rfc/rfc5452.txt

[93] R. Droms, J. Bound, B. Volz, T. Lemon, C. Perkins, and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)," RFC 3315 (Proposed Standard), Internet Engineering Task Force, Jul. 2003, updated by RFCs 4361, 5494, 6221. [Online]. Available: http://www.ietf.org/rfc/rfc3315.txt

[94] R. Droms, "DNS Configuration options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)," RFC 3646 (Proposed Standard), Internet Engineering Task Force, Dec. 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3646.txt

[95] R. Droms, "Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6," RFC 3736 (Proposed Standard), Internet Engineering Task Force, Apr. 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3736.txt

[96] K. Lauerman and J. King, "DHCP Consumption Attack and Mitigation Techniques," Cisco Systems, White Paper, 2010. [Online]. Available: http://www.cisco.com/en/US/prod/collateral/switches/ps5718/ps708/white_Paper_C11_603833.html

[97] D. L. Olson and D. Delen, *Advanced Data Mining Techniques.* Springer, 2008.

[98] B. D. Carrier, "Defining Digital Forensic Examination and Analysis Tool Using Abstraction Layers," *International Journal of Digital Evidence*, vol. 1, no. 4, pp. 1–12, 2003.

[99] E. Casey, *Digital evidence and Computer Crime: Forensics Science, Computer and the Internet*, 3rd ed. Academic Press, 2011.

[100] N. Walliman, *Research Methods, the basics.* Routledge, 2011.

[101] I. Dubrawsky, "SAFE Layer 2 Security In-Depth," Cisco Systems, Tech. Rep. Version 2, 2004. [Online]. Available: http://www.cisco.com/warp/public/cc/so/cuso/epso/sqfr/sfblu_wp.pdf

[102] E. Davies, S. Krishnan, and P. Savola, "IPv6 Transition/Co-existence Security Considerations," RFC 4942 (Informational), Internet Engineering Task Force, Sep. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4942.txt

[103] "Implementing First Hop Security in IPv6." [Online]. Available: http://www.cisco.com/en/US/docs/ios/ipv6/configuration/guide/ip6-first_hop_security.html

[104] R. Bonica, D. Gan, D. Tappan, and C. Pignataro, "Extended ICMP to Support Multi-Part Messages," RFC 4884 (Proposed Standard), Internet Engineering Task Force, Apr. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4884.txt

[105] M. Lottor, "Domain Administrators Operations Guide," RFC 1033, Internet Engineering Task Force, Nov. 1987. [Online]. Available: http://www.ietf.org/rfc/rfc1033.txt

[106] "VLAN SPAN." [Online]. Available: http://www.cisco.com/en/US/tech/tk389/tk816/tk836/tsd_technology_support_sub-protocol_home.html

[107] K. Kent, S. Chevalier, T. Grance, and H. Dang, "Guide to Integrating Forensice Techniques into Incident Respone," National Institute of Standards and Technology, Tech. Rep. SP800-86, 2006.

[108] IEEE Std 802.1X, "IEEE Standard for Local and Metropolitan Area Networks - Port-Based Network Access Control," *IEEE Std 802.1X-2010*, 2010.

[109] "TCPDUMP/LIBPCAP public repository," December 2011. [Online]. Available: http://www.tcpdump.org/

[110] "NDPMon - IPv6 Neighbor Discovery Protocol Monitor," 2012. [Online]. Available: http://ndpmon.sourceforge.net

[111] Cisco, "Cisco Discovery Protocol (CDP)." [Online]. Available: http://www.cisco.com/en/US/tech/tk648/tk362/tk100/tsd_technology_support_sub-protocol_home.html

[112] C. Partridge and A. Jackson, "IPv6 Router Alert Option," RFC 2711 (Proposed Standard), Internet Engineering Task Force, Oct. 1999, updated by RFC 6398. [Online]. Available: http://www.ietf.org/rfc/rfc2711.txt

[113] *TIME(7) Linux Programmer's Manual*. [Online]. Available: http://www.kernel.org/doc/man-pages/online/pages/man7/time.7.html

[114] J. Mogul and S. Deering, "Path MTU discovery," RFC 1191 (Draft Standard), Internet Engineering Task Force, Nov. 1990. [Online]. Available: http://www.ietf.org/rfc/rfc1191.txt

[115] IANA, "IP Option Numbers." [Online]. Available: http://www.iana.org/assignments/ip-parameters

[116] E. Leblond, P. N. Ayuso, P. McHardy, J. Engelhardt, and M. D. Four, "Secure use of iptables and connection tracking helpers," 2011. [Online]. Available: http://home.regit.org/netfilter-en/secure-use-of-helpers/

[117] M. Litvak, *IP(8) Linux*. [Online]. Available: http://www.linuxcommand.org/man_pages/ip8.html

[118] T. Narten, E. Nordmark, and W. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)," RFC 2461 (Draft Standard), Internet Engineering Task Force, Dec. 1998, obsoleted by RFC 4861, updated by RFC 4311. [Online]. Available: http://www.ietf.org/rfc/rfc2461.txt

[119] N. B. Lucena, G. Lewandowski, and S. J. Chapin, "Covert Channels in IPv6," pp. 147–166, 2006.

[120] T. Sohn, J. Moon, S. Lee, D. H. Lee, and J. Lim, "Covert Channel Detection in the ICMP Payload Using Support Vector Machine," pp. 828–835, 2003.

[121] S. Cabuk, C. E. Brodley, and C. Shields, "IP Covert Channel Detection," *ACM Trans. Inf. Syst. Secur.*, vol. 12, no. 4, pp. 22:1–22:29, Apr. 2009. [Online]. Available: http://doi.acm.org/10.1145/1513601.1513604

[122] G. V. de Velde, T. Hain, R. Droms, B. Carpenter, and E. Klein, "Local Network Protection for IPv6," RFC 4864 (Informational), Internet Engineering Task Force, May 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4864.txt

Appendix A

Forensic Information in a Network Environment

This appendix presents the categories and the criteria used to group the identified information. Afterwards each piece of information is presented in detail with a description, references where the information was identified and the categorisation including a explanatory statement.

A.1 Categories and Criteria

The following two subsections define the criteria used to categorise the identified information.

A.1.1 Usefulness for an Investigation

The usefulness for an investigation is categorised into:

Critical Without this information it is not possible to retrace a network scenario.

Valuable Without this information it is still possible to retrace a scenario. However, the information simplifies the retracing significantly but it might be redundant.

Futile The information has no value for the investigation. It is not used to retrace any scenario at all.

A.1.2 Degree of Difficulty to Collect the Information

The degree of difficulty to collect the information is categorised into:

Easy The information can be collected from a single device with a single administrative interface. The device sends the information actively to a collector (e.g. Syslog) or it is possible to collect the information automatically.

Medium Everything between the categories easy and hard (e.g. information that can be collected from a single administrative interface but is complex to parse and to track changes).

Hard The information is not available through an administrative interface. It is not possible to access the information without either modifying systems beyond the administrative capabilities (e.g. changing source code) or additional systems have to be installed to collect the information (e.g. a packet sniffer).

A.2 Ethernet

A.2.1 Information I.DE.01 - Frame Header

Description: The Ethernet frame header includes the checksum, the destination address, the source address, an optionally tag and the type or length field.
References: IEEE Std 802.3 [26] Clause 3 p. 49
Difficulty to Collect: Hard - Frame needs to be collected by a packet capturer.
Usefulness for an Investigation: Critical - The source address of the frame allows determining the sender of the frame and the destination address the intended recipient. The tag allows relating a frame to a VLAN and the type/length field is needed to parse the frame correctly.

A.2.2 Information I.DE.02 - Frame Payload

Description: The payload the frame is transporting.
References: IEEE Std 802.3 [26] Clause 3 p. 49
Difficulty to Collect: Hard - Frame needs to be collected by a packet capturer.
Usefulness for an Investigation: Valuable - The payload is passed to the upper layer and can be inspected there.

A.2.3 Information I.DE.03 - Interface Address

Description: The MAC address of an interface.
References: IEEE Std 802.3 [26] p. 52
Difficulty to Collect: Easy - The command `ip link list` allows collecting the information in Linux. It shows the data-link layer addresses of all interfaces.
Usefulness for an Investigation: Critical - Without the knowledge which MAC address is used by with network interface and, therefore, which node it is impossible to retrace most scenarios.

A.2.4 Information I.DE.04 - Duplex

Description: The duplex setting of an interface (full-duplex or half-duplex),
References: IEEE Std 802.3ab [27] p. 62, IEEE Std 802.3 [26] p. 2, IEEE Std 802.1D [28] p. 151

Difficulty to Collect: Medium - Depending on the network card driver (kernel module), the command `ethtool` shows the duplex status of the interface. The User Mode Linux based virtualised network setup, used in this research, does not support the query of the duplex status but it is always full-duplex.

Usefulness for an Investigation: Futile - The actual duplex setting is not relevant to show that a frame was transferred. It merely has a performance impact.

A.2.5 Information I.DE.05 - Link Master/Slave

Description: The state (master/slave) of the physical layer of the port. It is used for the auto-negotiation of duplex and speed settings.

References: IEEE Std 802.3ab [27] p. 24

Difficulty to Collect: Medium - Depending on the network card driver (kernel module), the command `ethtool` shows the master/slave status of the interface. The User Mode Linux based virtualised network setup, used in this research, does not support the query of the master/slave status.

Usefulness for an Investigation: Futile - Which side is master or slave is not relevant when the interface is up.

A.2.6 Information I.DE.06 - Link Operational Status

Description: PMA_LINK.indicate and MAC_Operational define the operational status of an interface on the physical and MAC layer respectively.

References: IEEE Std 802.3ab [27] p. 22, IEEE Std 802.1D [28] p.30

Difficulty to Collect: Easy - The command `ip link list` allows collecting the information in Linux. It shows the interface status (UP, LOWER_UP, DOWN). LOWER_UP means that a network cable is connected to the interface.

Usefulness for an Investigation: Critical - The interface operation status is critical for an investigation to show that a nod was connected to a network.

A.2.7 Information I.DE.07 - Speed

Description: The speed of the interface (10 MB, 100 MB, 1 GB).

References: IEEE Std 802.3ab [27] p. 62

Difficulty to Collect: Medium - Depending on the network card driver (kernel module), the command `ethtool` shows the link speed of the interface. The User Mode Linux based virtualised network setup, used in this research, does not support the query of the link speed.

Usefulness for an Investigation: Valuable - The actual speed is not relevant to show that a frame was transferred. On the other hand, the available network bandwidth might be relevant to show the feasibility of a DoS attack.

A.2.8 Information I.DE.08 - Physical Layout

Description: The physical layout and connections of the network equipment. This includes all nodes (hosts and routers) and switches with all network connections between them.

References: -

Difficulty to Collect: Hard - The actual connections must be documented in person. Protocols such Link Layer Discover Protocol [27] and Cisco Discover Protocol [111] might simplify the task.

Usefulness for an Investigation: Critical - The physical layout and the connections between the network equipment are critical to understand the network, the active protocols and the device configuration.

A.2.9 Information I.DE.09 - Traffic Filter

Description: The position and configuration of any traffic filter (aka firewall) that selects traffic based on Ethernet protocol fields.

References: -

Difficulty to Collect: Easy - Linux has a built-in traffic filtering mechanism that is active by default if not explicitly deactivated during kernel compilation. However, the default settings are not to filter any traffic. The command `ebtables -L` allows collecting the current filter rules.

Usefulness for an Investigation: Critical - Devices that filter traffic are significant for any investigation trying to retrace traffic flows.

A.2.10 Information I.DE.10 - Implementation Version

Description: The exact version of software or hardware implementing the Ethernet protocol.

References: -

Difficulty to Collect: Easy - In the network scenario used the exact kernel version defines the behaviour of all the nodes in the network. All behaviour related to Ethernet is handled either in the Linux kernel or in the network card. The command `uname -a` shows the exact kernel version used. The compilation parameter of the kernel can be retrieved with the command `zless /proc/config.gz`. In the analysed scenario the exact version of the virtualiser (User Mode Linux) and the helper daemons are also necessary:

```
linux --version
linux --showconfig
/var/local/netkit/kernel/netkit-kernel --version
/var/local/netkit/kernel/netkit-kernel --showconfig
cat /var/local/netkit/netkit-version
```

Usefulness for an Investigation: Critical - The exact version of device or software used to bridge Ethernet frames allows recreating the network setup and retrace network scenarios.

A.2.11 Information I.DE.11 - Traffic Counter

Description: Counter that total received and sent bytes/frames on Ethernet ports.
References: -
Difficulty to Collect: Easy - The command `cat /proc/net/dev` allows collecting the information under Linux. It shows the bytes and packets received and transmitted. In addition, the number off errors and drops is shown.
Usefulness for an Investigation: Valuable - The amount of traffic sent or received might help to conclude that a certain attack (e.g. DoS attack) originated from a certain port.

A.3 Bridge

A.3.1 Information I.DB.01 - Ageing Time

Description: The time how long dynamically learned filter entries remain in the Filter Database (default 300 seconds).
References: IEEE Std 802.1D [28] p. 45
Difficulty to Collect: Easy - The command `brctl showstp <BRIDGE>` shows the ageing time.
Usefulness for an Investigation: Valuable - The aging time triggers changes in the Filter Database. When the changes are tracked, the aging time is not important. It can be used to calculate the time when the last time traffic with a specific MAC address was received.

A.3.2 Information I.DB.02 - Bridge Address

Description: The address of the bridge (lowest interface address).
References: IEEE Std 802.1D [28] p. 52
Difficulty to Collect: Easy - The command `rstpctl showbridge <BRIDGE>` shows the bridge identifier. The part after the dash is the bridge address.
Usefulness for an Investigation: Critical - The RSTP algorithm cannot be retraced without the bridge address.

A.3.3 Information I.DB.03 - Bridge Identifier

Description: Bridge Priority + Bridge Address
References: IEEE Std 802.1D [28] p. 139
Difficulty to Collect: Easy - The command `rstpctl showbridge <BRIDGE>` shows the bridge identifier.
Usefulness for an Investigation: Valuable - If the bridge address and priority is known, the bridge identifier can be concluded.

A.3.4 Information I.DB.04 - Bridge Priority

Description: The RSTP priority of the bridge.

References: IEEE Std 802.1D [28] p. 138
Difficulty to Collect: Easy - The command `rstpctl showbridge <BRIDGE>` shows the bridge priority.
Usefulness for an Investigation: Critical - The RSTP algorithm cannot be retraced without the bridge address.

A.3.5 Information I.DB.05 - Filter Database

Description: The MAC address table that is built by learning source MAC addresses from incoming frames and used to determine the outgoing interface for frames based on their destination MAC address.
References: IEEE Std 802.1D [28] p. 42
Difficulty to Collect: Medium - The command `brctl showmacs <BRIDGE>` shows the current Filter Database (MAC address table). However, changes to the Filter Database cannot be tracked directly. Because it is not just a simple value that needs to be tracked but also a whole state table, the difficulty to collect is set to medium.
Usefulness for an Investigation: Valuable - The Filter Database contains the source MAC address of frames entering the switch. Therefore, it allows retracing the access port were a device using a specific source MAC address is connected to the network and the time when it was the first time connected. Further, it can be used to determine the path a frame took through the network. The path has to correspond with the RSTP.

A.3.6 Information I.DB.06 - Port Identifier

Description: The Port Identifier is equal to Port Priority + Port Number
References: IEEE Std 802.1D [28] p. 139
Difficulty to Collect: Easy - The command `rstpctl showportdetail <BRIDGE>` shows the port identifier.
Usefulness for an Investigation: Valuable - If the port number and priority is known, the port identifier can be concluded.

A.3.7 Information I.DB.07 - Port Number

Description: The RSTP port number.
References: IEEE Std 802.1D [28] p. 138
Difficulty to Collect: Easy - The command `rstpctl showportdetail <BRIDGE>` shows the port number.
Usefulness for an Investigation: Critical - The RSTP algorithm cannot be retraced without the port number.

A.3.8 Information I.DB.08 - Port Path Cost

Description: The RSTP port cost.
References: IEEE Std 802.1D [28] p. 138

Difficulty to Collect: Easy - The command `rstpctl showportdetail <BRIDGE>` shows the port path cost.
Usefulness for an Investigation: Critical - The RSTP algorithm cannot be retraced without the port cost.

A.3.9 Information I.DB.09 - Port Priority

Description: The RSTP port priority.
References: IEEE Std 802.1D [28] p. 138
Difficulty to Collect: Easy - The command `rstpctl showportdetail <BRIDGE>` shows the port priority.
Usefulness for an Investigation: Critical - The RSTP algorithm cannot be retraced without the port priority.

A.3.10 Information I.DB.10 - Port State

Description: The RSTP port status (Forwarding, Learning, Discarding).
References: IEEE Std 802.1D [28] p. 35
Difficulty to Collect: Easy - The command `rstpctl showportdetail <BRIDGE>` shows the port status. Furthermore, the Linux kernel logs port changes to Syslog.
Usefulness for an Investigation: Valuable - The port state can be reconstructed based on the RSTP algorithm and the bridge identifier, port identifier and port path costs.

A.3.11 Information I.DB.11 - Root Path Cost

Description: The total cost to the Root Bridge.
References: IEEE Std 802.1D [28] p.139
Difficulty to Collect: Easy - The command `rstpctl showbridge <BRIDGE>` shows the root cost.
Usefulness for an Investigation: Valuable - The root path cost can be concluded based on the RSTP port states and the port path costs.

A.3.12 Information I.DB.12 - TCN Received

Description: The event that a bridge received a topology change notification (TCN).
References: -
Difficulty to Collect: Hard - The command `rstpctl showbridge <BRIDGE>` shows the time since the last topology change. However, the RSTP daemon would have to be modified to get an event notification for each TCN received.
Usefulness for an Investigation: Valuable - TCN messages mean changes in the network and lead to the deletion of entries in the Filter Database.

A.3.13 Information I.DB.13 - Port Role

Description: The RSTP port role (Root, Designated, Alternate, Backup, Disabled).
References: IEEE Std 802.1D [28] p. 145
Difficulty to Collect: Easy - The command `rstpctl showport <BRIDGE>` shows the port role.
Usefulness for an Investigation: Valuable - The port role is an important information that shows the result of the RSTP algorithm.

A.3.14 Information I.DB.14 - Root Bridge

Description: The Bridge Identifier of the current Root Bridge.
References: IEEE Std 802.1D [28] p. 139
Difficulty to Collect: Easy - The command `rstpctl showbridge <BRIDGE>` shows the current root bridge.
Usefulness for an Investigation: Valuable - The current root bridge is an important information that shows the result of the RSTP algorithm. The root bridge can also be determined by knowing all spanning tree parameters and the physical network layer.

A.3.15 Information I.DB.15 - Implementation Version

Description: The exact version of software or hardware bridging Ethernet frames.
References: -
Difficulty to Collect: Easy - In the network scenario used, the exact kernel version defines the behaviour of the bridges. The command `uname -a` shows the exact kernel version running. The compilation parameter of the kernel can be retrieved with the command `zless /proc/config.gz`.
Usefulness for an Investigation: Critical - The exact version of software or hardware used to bridge Ethernet frames allows retracing the network setup and network scenarios.

A.3.16 Information I.DB.16 - Bridge Log

Description: The log created by the bridge.
References: -
Difficulty to Collect: Easy - Because the bridging functionality is split into two functions: spanning tree and bridging, the log message end up in different logs. The RSTP daemon logs with the Syslog facility daemon and the log messages end up in the file `/var/log/daemon.log` the bridging module in the Linux kernel logs with the Syslog facility kern and the log messages are stored in the file `/var/log/kern.log`.
Usefulness for an Investigation: Valuable - The logging messages allow retracing events in the bridging code of the kernel and the RSTP daemon.

A.4 VLAN

A.4.1 Information I.DV.01 - Interface Role

Description: The role of the interface (access or trunk). An access interface accepts only untagged frames while a trunk interface accepts tagged and untagged frames (native VLAN).

References: IEEE Std 802.1Q [33] p. 41

Difficulty to Collect: Easy - The Linux command `ip link list` and `brctl` shows VLAN interfaces and their configuration.

Usefulness for an Investigation: Critical - Without the knowledge of the port role many scenarios cannot be retraced.

A.4.2 Information I.DV.02 - Interface VLAN ID

Description: The VLAN that untagged frames are assigned to on an access interface. The VLAN IDs that are accepted on a trunk interface and the VLAN untagged frames are assigned to (native VLAN).

References: IEEE Std 802.1Q [33] p. 42

Difficulty to Collect: Easy - The Linux command `ip link list` shows VLAN interfaces.

Usefulness for an Investigation: Critical - Without the knowledge which port is assigned to which VLAN many scenarios cannot be retraced.

A.4.3 Information I.DV.03 - Implementation Version

Description: The exact version of the software or hardware implementing the IEEE 802.1Q protocol.

References: -

Difficulty to Collect: Easy - In the network scenario used, the exact kernel version defines the exact implementation of the IEEE 802.1Q protocol. The command `uname -a` shows the exact kernel version running. The compilation parameter of the kernel can be retrieved with the command `zless /proc/config.gz`.

Usefulness for an Investigation: Critical - The exact version of software or hardware used to handle 802.1Q tagged frames allows recreating the network setup and retrace network scenarios.

A.4.4 Information I.DV.04 - Traffic Filter

Description: The position and configuration of any traffic filter (aka firewall) that selects traffic based on 802.1Q protocol fields.

References: -

Difficulty to Collect: Easy - Linux has a built-in traffic filtering mechanism that is active by default if not explicitly deactivated during kernel compilation. However, the default settings are not to filter any traffic. The command `ebtables -L` allows collecting the current filter settings under Linux.

Usefulness for an Investigation: Critical - Devices that filter traffic are significant for any investigation trying to retrace traffic flows.

A.5 IPv6

A.5.1 Information I.NI.01 - Packet Header

Description: The complete IP header (version, traffic class, flow label, payload length, next-header, hop-limit, source address, destination address) without any extension headers.

References: RFC 2460 [35] p. 4

Difficulty to Collect: Hard - The packet needs to be collected by a packet capturer.

Usefulness for an Investigation: Valuable - The version field is used by the receiving node to decide if it is able to interpret the traffic. The traffic class is used to distinguish between different classes and to priorities some packets.
Flows allow grouping packets. A flow is not necessarily 1:1 mapped to a transport protocol connect. Therefore, the classic flow 5-tuple (source and destination address, transport protocol, source and destination port) is not necessarily the same.
The payload length and the next-header is important for the parser.
While the hop-limit field is mostly used to make sure a packet does not loop endlessly in a network. It might also give some hints how far away (many hops) the attacker from the victim is.
The source address allows identifying the sender. However, it is simple to spoof the address if there is no rigorous source address verification in the sender network. On the other hand, it is quite difficult to establish a state-full session (e.g. TCP) with a spoofed source address.
The destination address identifies the intended receiver of the packet (see also Routing Header).

A.5.2 Information I.NI.02 - Hop-by-Hop Options Header

Description: None

References: RFC 2460 [35] p. 11

Difficulty to Collect: Hard - The packet needs to be collected by a packet capturer.

Usefulness for an Investigation: Valuable - As currently only a few Hop-by-Hop Options other than the padding options are defined (e.g. IPv6 Router Alert [112]), the part of the protocol is not critical to retrace incidents. However it might be valuable to retrace and log all packets with new and unknown header options.

A.5.3 Information I.NI.03 - Routing Header

Description: The routing header might be used to select intermediate systems where a packet has to pass through. The type 0 routing header is deprecated in RFC 5095 [36]. The type 2 header is used by mobile IPv6.

References: RFC 2460 [35] p. 12, RFC 5095 [36]

Difficulty to Collect: Hard - The packet needs to be collected by a packet capturer.

Usefulness for an Investigation: Critical - The type 0 routing header allows changing the path the packet takes through the network. Even though the type 0 header is deprecated, some nodes might still interpret it and the header might prove to be critical for an investigation.

A.5.4 Information I.NI.04 - Fragment Header

Description: Fragmentation is used when the IP packet does not fit inside the path MTU to its destination. When fragmentation is performed, a node uses the Fragment Header to break down the datagram into smaller IP fragments that fit in the path MTU [35]. RFC 5722 [37] explicitly forbids overlapping IP fragments.

References: RFC 2460 [35] p. 18, RFC 5722 [37]

Difficulty to Collect: Hard - The packet needs to be collected by a packet capturer.

Usefulness for an Investigation: Critical - Fragmentation is complex and allows hiding information from intermediate security systems (overlapping fragments). It also brings some burden on the intermediate systems scanning the traffic and end systems assembling the frame before passing it on to the transport layer. DoS attacks based on packet fragmentation are easily to implement, and, therefore, the header might be critical in an investigation.

A.5.5 Information I.NI.05 - Destination Options Header

Description: Used to send options to the receiver of the IP packet.

References: RFC 2460 [35] p. 23

Difficulty to Collect: Hard - The packet needs to be collected by a packet capturer.

Usefulness for an Investigation: Valuable - As currently no other Destination Options than the padding options are defined, the part of the protocol is not critical to retrace incidents. However, the option might be misuse as a covert channel and new or unknown Destination Options should be tracked and logged.

A.5.6 Information I.NI.06 - Authentication Header

Description: Used to authenticate the sender of the packet and validated the integrity of the packet.

References: RFC 2460 [35] p. 7

Difficulty to Collect: Hard - The packet needs to be collected by a packet capturer.

Usefulness for an Investigation: Futile - The header allows verifying the authenticity of the packet depending on the exact selection of algorithms used. The reason why it is classified as futile is because IPSec is out of the scope of this research.

A.5.7 Information I.NI.07 - Encapsulating Security Payload Header

Description: Used to encrypted the packet payload.

References: RFC 2460 [35] p. 7

Difficulty to Collect: Hard - The packet needs to be collected by a packet capturer.

Usefulness for an Investigation: Futile - It is classified as futile is because IPSec is out of the scope of this research.

A.5.8 Information I.NI.08 - Interface Addresses

Description: All IP addresses with subnet masks configured on an interface (link-local and global). All interfaces are required to have at least one link-local unicast address.

References: RFC 4862 [41] p. 11, RFC 4291 [14] p. 3

Difficulty to Collect: Easy - The command `ip address list` allows collecting the information under Linux. It shows the currently configured IP addresses for each interface.

Usefulness for an Investigation: Critical - The IP address identifies the nodes that sent or received traffic (source or destination IP in the packet header). Even though IP addresses are easy to spoofed they are an important hint to retrace network traffic flows and critical to investigations.

A.5.9 Information I.NI.09 - Routing Table

Description: The routing table combines the information of the default router list, the prefix list and statically added routes.

References: RFC 2460 [35] p. 6

Difficulty to Collect: Easy - The command `ip -6 route list` allows collecting the information under Linux. It shows the currently active routing table in the kernel.

Usefulness for an Investigation: Critical - The routing table decides how a node forwards or sends packets. That is critical information for retracing network traffic flows.

A.5.10 Information I.NI.10 - Node Type

Description: The type of the node: router or host.

References: RFC 2460 [35] p. 3

Difficulty to Collect: Easy - The command `cat /proc/sys/net/ipv6/ \ conf/*/forwarding` allows collecting the information under Linux. If the flag is set (1) the system forwards traffic and is a router, otherwise a host.

Usefulness for an Investigation: Critical - The way a node handles traffic that is not addressed to the node is significant in retracing traffic flows and, therefore, critical for an investigation.

A.5.11 Information I.NI.11 - Traffic Filter

Description: The position and configuration of any traffic filter (aka firewall) that selects traffic based on IPv6 or ICMPv6 protocol fields.

References: RFC 4942 [102] p. 5

Difficulty to Collect: Easy - Linux has a built-in traffic filtering mechanism that is active by default if not explicitly deactivated during kernel compilation. However, the default settings are not to filter any traffic. The commands `ip6tables -t filter -L`, `ip6tables -t mangle -L` and `ip6tables -t raw -L` allow collecting the current filter settings under Linux.

Usefulness for an Investigation: Critical - Devices that filter traffic are significant for any investigation trying to retrace traffic flows.

A.5.12 Information I.NI.12 - Traffic Mangler

Description: The position and configuration of any device that changes IPv6 or ICMPv6 protocol fields for any reason.

References: RFC 4942 [102] p. 5

Difficulty to Collect: Easy - Linux has a built-in traffic mangling mechanism that is active by default if not explicitly deactivated during kernel compilation. However, the default settings do not modify any traffic. The commands `ip6tables -t mangle -L` and `ip6tables -t raw -L` allow collecting the information under Linux.

Usefulness for an Investigation: Critical - Devices that change traffic are significant for any investigation trying to retrace traffic flows.

A.5.13 Information I.NI.13 - Implementation Version

Description: The exact version of the software or hardware implementing the IPv6 protocol.

References: -

Difficulty to Collect: Easy - In the network setup used for this research, the exact kernel version defines the exact implementation of the IPv6 protocol. The command `uname -a` shows the exact kernel version running. The compilation parameter of the kernel can be retrieved with the command `zless /proc/config.gz`.

Usefulness for an Investigation: Critical - The exact version of software or hardware used, that implements the IPv6 protocol allows recreating the network setup and retrace network scenarios.

A.5.14 Information I.NI.14 - Packet Payload

Description: The payload transported by the IP packet.
References: -
Difficulty to Collect: Hard - The packet needs to be collected by a packet capturer.
Usefulness for an Investigation: Valuable - The payload is passed to the upper layer and can, therefore, be inspected there.

A.5.15 Information I.NI.15 - Packet Unknown Header

Description: IPv6 allows creating new extension headers. This information covers all currently unknown extension headers.
References: -
Difficulty to Collect: Hard - The packet needs to be collected by a packet capturer.
Usefulness for an Investigation: Critical - Unknown extension headers might be used for covert channel or to crash IP protocol parsers.

A.5.16 Information I.NI.16 - Packet Unknown Option

Description: IPv6 allows creating new hop-by-hop and destination option. This information covers all currently unknown hop-by-hop and destination options.
References: -
Difficulty to Collect: Hard - The packet needs to be collected by a packet capturer.
Usefulness for an Investigation: Critical - Unknown hop-by-hop or destination options might be used for covert channel or to crash IP protocol parsers.

A.5.17 Information I.NI.17 - Protocol Parameter

Description: In the development of the protocol some features were deprecated (e.g. type 0 source routing header) others were added and some are configurable. Therefore, most IP stacks are configurable with parameters.
References: -
Difficulty to Collect: Easy - The IPv6 protocol parameter of the Linux kernel can be found inside the folder `/proc/sys/net/ipv6/conf`.
Usefulness for an Investigation: Critical - Some settings change the behaviour of the node significantly and are, therefore, crucial to retrace network scenarios.

A.6 ICMPv6

A.6.1 Information I.NC.01 - Router Solicitation Message

Description: Hosts send RS messages to prompt routers to generate RA messages quickly.

References: RFC 4861 [56] p. 18

Difficulty to Collect: Hard - The packet needs to be collected by a packet capturer.

Usefulness for an Investigation: Valuable - The message is sent by hosts recently connected to the network. Therefore, the message allows tracking of new hosts in the network. Furthermore, a mechanism that can be used by a culprit to trigger an action on a system bears the chance that it is misused to launch a DoS attacks, especially because the trigger message is not authenticated.

A.6.2 Information I.NC.02 - Router Advertise Message

Description: Routers send RA messages periodically, or in response to RS messages. The advertisement contains the Cur Hop Limit, the Router Lifetime, the Reachable Time, the Retrans Timer, and possible the Source data-link layer address, the MTU and Prefix Information.

References: RFC 4861 [56] p. 19

Difficulty to Collect: Hard - The packet needs to be collected by a packet capturer.

Usefulness for an Investigation: Valuable - The RA message is crucial for a host to determine several parameter in the local network, including the default gateway. Wrong or spoofed RA messages break network connections or redirect network traffic. It is also possible to negate information sent in a previous RA message with a subsequent RA message. The elementary part in the network connection and the diversity of attack options make the RA messages valuable for an investigation.

A.6.3 Information I.NC.03 - Neighbour Solicitation Message

Description: Nodes send NS messages to request the data-link layer address of a target node while also providing their own data-link layer address to the target. NS messages are multicast when the node needs to resolve an address and unicast when the node seeks to verify the reachability of a neighbour.

References: RFC 4861 [56] p. 22

Difficulty to Collect: Hard - The packet needs to be collected by a packet capturer.

Usefulness for an Investigation: Valuable - The packet is sent to resolve an neighbour's data-link layer address but also may communicate the link-layer address of the sender to the neighbour. The message is a crucial part of linking the data-link and the network layer and, therefore, valuable for an investigation.

A.6.4 Information I.NC.04 - Neighbour Advertise Message

Description: A node sends NA messages in response to NS messages and sends unsolicited NA messages to (unreliably) propagate new information quickly.

References: RFC 4861 [56] p. 23

Difficulty to Collect: Hard - The packet needs to be collected by a packet capturer.

Usefulness for an Investigation: Valuable - The packet is sent as a reply to a request for the receiver's data-link layer address. The message changes the neighbour cache of nodes receiving it. The message is a crucial part of linking the data-link and the network layer and, therefore, valuable for an investigation.

A.6.5 Information I.NC.05 - Redirect Message

Description: Routers send Redirect messages to inform a host of a better first-hop node on the path to a destination. Hosts can be redirected to a better first-hop router but can also be informed by a redirect that the destination is in fact a neighbour. The latter is accomplished by setting the ICMP Target Address equal to the ICMP Destination Address.

References: RFC 4861 [56] p. 26

Difficulty to Collect: Hard - The packet needs to be collected by a packet capturer.

Usefulness for an Investigation: Valuable - The message allows redirecting network traffic, which might be misused by attackers. As Redirect messages should not be very common in a neatly configured access network, they should be tracked.

A.6.6 Information I.NC.06 - Neighbour Cache

Description: The Neighbour Cache is a list of neighbours recently sent traffic to. An entry consists of the neighbour's unicast address, its link layer address and a flag whether is a router or a host. It also contains information used by the Unreachability Detection algorithm.

References: RFC 4861 [56] p. 33

Difficulty to Collect: Easy - The command `ip neighbour list` allows collecting the information under Linux. It shows the currently active neighbours with the corresponding data-link layer and network layer addresses. The cache timeout can be determined with the command `cat /proc \ /sys/net/ipv6/neigh/*/base_reachable_time_ms`. The default is 30 seconds.

Usefulness for an Investigation: Critical - The neighbour cache is used by the node to determine the link-layer addresses of the receiver or default router. Therefore, it is quite an elementary part of the communication and a worthwhile target for an attacker. It is the reference to retrace the connection between data-link and network layer addresses. Therefore, it is critical for an investigation.

A.6.7 Information I.NC.07 - Destination Cache

Description: The Destination Cache is a list of destinations recently sent traffic to. It contains entries for on-link and off-link destinations. An entry consists of the destination address, the next-hop address and optionally other information related such as the Path MTU or round-trip timer.
Entries might be updated through Redirect Message.

References: RFC 4861 [56] p. 34

Difficulty to Collect: Easy - The commands `netstat -6 -C -r` and `ip -6 -s route list cache` allow collecting the information under Linux. It shows destination prefixes and the corresponding next hop, metric, outgoing interface and use counters. How long an entry stays in the destination cache depends on the configuration of the garbage collector. The command `cat /proc/sys/net/route/gc_*` allows collecting the information under Linux. The default is 30 seconds.

Usefulness for an Investigation: Valuable - The cache stores the result of lookups in the Prefix List, the Default Router List, the Routing Table and the Neighbour Cache, therefore, the information is redundant. On the other hand, it might prove valuable in an investigation as a second source.

A.6.8 Information I.NC.08 - Prefix List

Description: The prefix list contains address ranges that are on-link. Entries are created by RA messages and they expire based on the time in the advertisement.

References: RFC 4861 [56] p. 34

Difficulty to Collect: Easy - The announced prefixes can be found in the Linux routing table. The command `ip -6 route list` allows collecting the information under Linux. It shows the currently active routing table. The on-link prefixes are the ones with a next-hop associated to them.

Usefulness for an Investigation: Critical - The prefix list is used by the node to determine which addresses are on-link. This influences the traffic routing, and, therefore, it is critical for an investigation.

A.6.9 Information I.NC.09 - Default Router List

Description: The default router list contains entries for potential routers. The entries expire based on the timer in the RA message they are based on. Other parameters received in the RA might be associated with the entry (e.g. whether the host should use DHCP or auto-configuration, hop limit, link MTU).

References: RFC 4861 [56] p. 34

Difficulty to Collect: Easy - Linux holds the default router list in the main routing table (relevant entries start with default). The command `ip -6 route list` allows collecting the information under Linux. The command shows the currently active routing table in the Kernel. The default routes are the ones with the prefix-length 0.

Usefulness for an Investigation: Critical - The default router list is used by the node to determine which router to use for unknown off-link destinations. This information is critical to retrace the traffic routing.

A.6.10 Information I.NC.10 - Routing Enabled

Description: A flag indicating whether routing is enabled on an interface.

References: RFC 4861 [56] p. 41

Difficulty to Collect: Easy - The command `cat /proc/sys/net/ipv6 \ /conf/all/forwarding` allows collecting the information under Linux. The routing is active if the flag is set (1).

Usefulness for an Investigation: Critical - If a node is actually forwarding packets or not changes significantly how it handles IP packets not addressed to the node. Therefore, this information is crucial to retrace traffic paths in the network.

A.6.11 Information I.NC.11 - Send Router Advertisements

Description: A flag indicating whether or not the router sends periodic RA messages and responds to RS messages on the interface.

References: RFC 4861 [56] p. 41

Difficulty to Collect: Easy - Linux does not send RA messages by itself. The routing daemon Quagga can be configured to send RA messages. The command `show interface` allows collecting the information in the Quagga shell (vtysh). The lines starting with *ND router advertisement* show if RA messages are sent.

Usefulness for an Investigation: Futile - As only router and malicious nodes send RA messages the value of the flag is not very relevant for genuine nodes. Furthermore, it is possible to see the effects of systems sending RA messages on all nodes (new routers appear in the Default Router List and new prefixes in the Prefix List) on the same link.

A.6.12 Information I.NC.12 - RA Maximum Interval

Description: The maximum time between sending unsolicited multicast router advertisements in seconds.

References: RFC 4861 [56] p. 41

Difficulty to Collect: Easy - Linux does not send RA messages by itself. The routing daemon Quagga can be configured to send RA messages. The command `show interface` allows collecting the information in the

Quagga shell (vtysh). The line starting with *ND router advertisements are sent every* shows the frequency the RA messages are sent.

Usefulness for an Investigation: Valuable - The value only defines how often an RA message can be expected by nodes on the same link. The actually sent and received RA messages are far more relevant as the actually change the Default Router and Prefix List on nodes. However, the value might be relevant for reference purposes to know what is the normal RA message interval of genuine routers.

A.6.13 Information I.NC.13 - RA Minimum Interval

Description: The minimum time between sending unsolicited multicast router advertisements in seconds.

References: RFC 4861 [56] p. 41

Difficulty to Collect: Easy - Linux does not send RA messages by itself. The routing daemon Quagga can be configured to send RA messages. The command `show interface` allows collecting the information in the Quagga shell (vtysh). The line starting with *ND router advertisements are sent every* shows the frequency the RA messages are sent.

Usefulness for an Investigation: Valuable - The value only defines how often an RA message can be expected by nodes on the same link. The actually sent and received RA messages are far more relevant as the actually change the Default Router and Prefix List on nodes. However, the value might be relevant for reference purposes to know what is the normal RA message interval of genuine routers.

A.6.14 Information I.NC.14 - RA Managed Flag

Description: The state of the Managed Address Configuration Flag in the Router Advertisements sent on the interface.

References: RFC 4861 [56] p. 41

Difficulty to Collect: Easy - Linux does not send RA messages by itself. The routing daemon Quagga can be configured to send RA messages. The command `show interface` allows collecting the information in the Quagga shell (vtysh). The line starting with *Hosts use* shows what kind of address configuration is announced.

Usefulness for an Investigation: Valuable - The flag only defines if the router sending the RA messages indicate to the receiving node that it should generate an address based on the received on-link prefixes. The actually sent and received RA messages are far more relevant. However, the value might be relevant for reference purposes to know what address configuration genuine routers indicate to the nodes.

A.6.15 Information I.NC.15 - RA Other Configuration Flag

Description: The value of the Other Configuration flag in the Router Advertisements sent on the interface.

References: RFC 4861 [56] p. 42

Difficulty to Collect: Easy - Linux does not send RA messages by itself. The routing daemon Quagga can be configured to send RA messages. The command `show interface` allows collecting the information in the Quagga shell (vtysh). The line starting with *Hosts use* shows what kind of address configuration is announced.

Usefulness for an Investigation: Valuable - The flag only defines if the router sending the RA messages indicates to the receiving node that it should try to receive an address by DHCP. The actually sent and received RA messages are far more relevant. However, the value might be relevant for reference purposes to know what address configuration genuine routers indicate to the nodes.

A.6.16 Information I.NC.16 - RA Link MTU

Description: The advertised Link MTU in the RA messages on the interface.

References: RFC 4861 [56] p. 42

Difficulty to Collect: Easy - Linux does not send RA messages by itself. The routing daemon Quagga can be configured to send RA messages. The command `show interface` allows collecting the information in the Quagga shell (vtysh). Quagga does not support to advertise the Link MTU.

Usefulness for an Investigation: Valuable - The value is used in the RA messages sent by the router to indicate what MTU the on-link nodes should use. The actually sent and received RA messages are far more relevant. However, the value might be relevant for reference purposes to know what Link MTU genuine routers indicate the nodes.

A.6.17 Information I.NC.17 - RA Reachable Time

Description: The value to be placed in the Reachable Time field in the RA messages sent by the router

References: RFC 4861 [56] p. 42

Difficulty to Collect: Easy - Linux does not send RA messages by itself. The routing daemon Quagga can be configured to send RA messages. The command `show interface` allows collecting the information in the Quagga shell (vtysh). The line starting with *ND advertised reachable time is* shows the advertised reachable time.

Usefulness for an Investigation: Valuable - The value is used in the RA messages sent by the router to indicate for how long a node should assume another node reachable after having received a reachability conformation. The actually sent and received RA messages are far more relevant. However, the value might be relevant for reference purposes to know what Reachability Time genuine routers indicate to the nodes.

A.6.18 Information I.NC.18 - RA Retransmission Timer

Description: The value to be placed in the Retrans Timer field in the RA messages sent by the router.

References: RFC 4861 [56] p. 42

Difficulty to Collect: Easy - Linux does not send RA messages by itself. The routing daemon Quagga can be configured to send RA messages. The command `show interface` allows collecting the information in the Quagga shell (vtysh). The line starting with *ND advertised retransmit interval is* shows the advertised reachable time.

Usefulness for an Investigation: Valuable - The value is used in the RA messages sent by the router to indicate for how long a node should wait before retransmitting a neighbour solicitation message. The actually sent and received RA messages are far more relevant. However, the value might be relevant for reference purposes to know what Retrans Timer genuine routers indicate to the nodes.

A.6.19 Information I.NC.19 - RA Current Hop Limit

Description: The default value to be placed in the Cur Hop Limit field in the RA messages sent by the router.

References: RFC 4861 [56] p. 42

Difficulty to Collect: Easy - Linux does not send RA messages by itself. The routing daemon Quagga can be configured to send RA messages. The command `show interface` allows collecting the information in the Quagga shell (vtysh). Quagga does not support to advertise Current Hop Limit.

Usefulness for an Investigation: Valuable - The value is used in the RA messages sent by the router to indicate what initial hop count should be used in new IP packets created by the node. However, as an attacker might send RA messages with a very low Current Hop Limit as a DoS attack, it might be valuable for an investigation to know what Current Hop Count Limit the genuine routers are configured to send.

A.6.20 Information I.NC.20 - RA Default Lifetime

Description: The value to be placed in the Router Lifetime field of RA messages sent from the interface.

References: RFC 4861 [56] p. 43

Difficulty to Collect: Easy - Linux does not send RA messages by itself. The routing daemon Quagga can be configured to send RA messages. The command `show interface` allows collecting the information in the Quagga shell (vtysh). The line starting with *ND router advertisements live for* shows the advertised lifetime.

Usefulness for an Investigation: Valuable - The value is used in the RA messages sent by the router to indicate how long a node should keep the router in the Default Router List. The Default Router List is relevant to retrace network traffic flows. In addition, the value might be useful for reference to identify spoofed RA messages. The value can also be seen in the Default Router List of on-link nodes.

A.6.21 Information I.NC.21 - RA Prefix List

Description: A list of prefixes to be placed in Prefix Information options in RA messages sent from the interface.

References: RFC 4861 [56] p. 43

Difficulty to Collect: Easy - Linux does not send RA messages by itself. The routing daemon Quagga can be configured to send RA messages. The command `show run` allows collecting the information in the Quagga shell (vtysh). The configuration is found in the interface section. The line starting with *ipv6 nd prefix* shows which prefixes are announced.

Usefulness for an Investigation: Valuable - The router announces prefixes that the receiving node should consider as on-link. Because attack can sent spoofed RA messages with more prefixes to redirect traffic to them, it might be useful to know which router announces which prefixes. The result of receiving an RA message with prefix information can be found in the Prefix List of a node.

A.6.22 Information I.NC.22 - Interface Link MTU

Description: The MTU of the link.

References: RFC 4861 [56] p. 52

Difficulty to Collect: Easy - The command `cat /proc/sys/net/ipv6 \ /conf/[interface]/mtu` allows collecting the information under Linux. It shows the link MTU in bytes.

Usefulness for an Investigation: Futile - The link MTU of nodes is not a critical information. If the node implements the RFC 2460 [35] correctly, it is at least 1280 byte. However, when the link would support jumbo frames the value becomes more critical and a DoS attack that lowers the MTU of node becomes much more relevant.

A.6.23 Information I.NC.23 - Interface Cur Hop Limit

Description: The default hop limit to be used when sending IP packets.

References: RFC 4861 [56] p. 52

Difficulty to Collect: Easy - The command `cat /proc/sys/net/ipv6 \ /conf/[interface]/hop_limit` allows collecting the information under Linux. It shows the current hop limit used for new IP traffic sent through that interface.

Usefulness for an Investigation: Valuable - The hop limit a node uses to create new IP packets directly determines how many hops an IP packet can travel before it gets dropped. Therefore, it is valuable information to retrace network traffic flows. It might also be used to indicate how many hops a packet has passed before seen if the start hop limit is known.

The value can be reproduced by tracing the RA messages a node receives, although a host might choose to ignore the indicated hop limit all together.

A.6.24 Information I.NC.24 - Interface Base Reachable Time

Description: A base value used for computing the random ReachableTime value.

References: RFC 4861 [56] p. 52

Difficulty to Collect: Easy - The command `cat /proc/sys/net/ipv6 \ /conf/[interface]/base_reachable_time` shows the currently active Base Reach Time.

Usefulness for an Investigation: Valuable - A very low reachability time value would keep a node in constant need to send NS messages to determine the reachability state of its neighbours. An attacker might misuse this for a DoS attack. Therefore, the value might prove useful during an investigation. However, the base reachable time does not determine the exact time a neighbour is recognised as reachable, but is used to calculate the exact reachable time.

A.6.25 Information I.NC.25 - Interface Reachable Time

Description: The time a neighbour is considered reachable after receiving a reachability confirmation.

References: RFC 4861 [56] p. 52

Difficulty to Collect: Hard - The information is not available through an administrative interface on Linux.

Usefulness for an Investigation: Valuable - A very low reachability time value would keep a node in constant need to send NS messages to determine the reachability state of its neighbours. An attacker might misuse this for a DoS attack. Therefore, the value might proven useful during an investigation.

A.6.26 Information I.NC.26 - Interface Retransmission Timer

Description: The time between retransmissions of Neighbour Solicitation messages to a neighbour when resolving the address or when probing the reachability of a neighbour.
For autoconfiguration purposes, RetransTimer specifies the delay between consecutive Neighbour Solicitation transmissions performed during Duplicate Address Detection (if DupAddrDetectTransmits is greater than 1), as well as the time a node waits after sending the last Neighbour Solicitation before ending the Duplicate Address Detection process.

References: RFC 4861 [56] p. 52, RFC 4862 [41] p. 11

Difficulty to Collect: Easy - The command `cat /proc/sys/net/ipv6 \ /conf/[interface]/retrans_timed` allows collecting the information under Linux. It shows the currently active retransmission timer.

Usefulness for an Investigation: Valuable - The timer is important to retrace ND behaviour of a node and the DAD behaviour of a node, and, therefore, valuable for an investigation. Furthermore, an attacker might create a DoS attack by answering all NS message during the DAD phase. The timer might be needed to show that the attack was possible and the attacker sent the NA messages fast enough.

A.6.27 Information I.NC.27 - Number of NS messages for DAD

Description: The number of consecutive Neighbour Solicitation messages sent while performing DAD on a tentative address. Zero means that no DAD is done.

References: RFC 4862 [41] p. 11

Difficulty to Collect: Easy - The command `cat /proc/sys/net/ipv6 \ /conf/[interface]/dad_transmits` allows collecting the information under Linux. It shows how many DAD are sent.

Usefulness for an Investigation: Valuable - The value is important to retrace the DAD behaviour of a node. Also, an attacker might create a DoS attack by answering all NS message during the DAD phase. The number of NS messages sent by the node might be needed to show that the attack was possible and the attacker sent the NA messages fast enough.

A.6.28 Information I.NC.28 - Destination Unreachable Message

Description: A Destination Unreachable message should be generated by a router, or by the IPv6 layer in the originating node, in response to a packet that cannot be delivered to its destination address for reasons other than congestion.

References: RFC 4443 [55] p. 8, RFC 4884 [104] p. 9

Difficulty to Collect: Hard - The packet needs to be collected by a packet capturer.

Usefulness for an Investigation: Valuable - The message is totally legitimate in normal network traffic. However, it might indicate problems in the network and it is also possible for an attacker, which is able to monitor network traffic of a node, to launch a DoS attack against the node by sending a destination unreachable message for every packet sent by the node.

A.6.29 Information I.NC.29 - Packet Too Big Message

Description: If a router cannot forward a packet because it is bigger than the MTU of the outing interface, the router drops the packet and informs the source with a Packet Too Big Message. This is part of the Path MTU discover process.

References: RFC 4443 [55] p. 10

Difficulty to Collect: Hard - The packet needs to be collected by a packet capturer.

Usefulness for an Investigation: Valuable - The message is totally legitimate in normal network traffic and also necessary to determine the path MTU for a connection. However, an attacker might force a node to send smaller packets than it actually is needed. In this case, the message might be valuable in an investigation.

A.6.30 Information I.NC.30 - Time Exceeded Message

Description: If the Hop Limit of a packet reaches zero, the router receiving the packet or decrementing the value to zero has to inform the source with an ICMP Time Exceeded message. This means that the initial value was chosen to small (network diameter is bigger than expected) or there is a routing loop.
References: RFC 4443 [55] p. 11, RFC 4884 [104] p. 9
Difficulty to Collect: Hard - The packet needs to be collected by a packet capturer.
Usefulness for an Investigation: Valuable - The message is totally legitimate in normal network traffic. However, it might indicate problems in the network and it is also possible for an attacker, which is able to monitor network traffic, to launch a DoS attack against the node by sending a Time Exceeded message for every packet of the node.

A.6.31 Information I.NC.31 - Parameter Problem Message

Description: A router informs the source of a packet with an ICMP Parameter Problem message if it finds a problem within the IPv6 header or extension headers that does not allow processing the packet.
References: RFC 4443 [55] p. 12
Difficulty to Collect: Hard - The packet needs to be collected by a packet capturer.
Usefulness for an Investigation: Valuable - The message is totally legitimate in normal network traffic. However, it might indicate problems in the network and it is also possible for an attacker that can see the traffic of a node to DoS the node by sending a parameter problem message for every packet of that node.

A.6.32 Information I.NC.32 - Echo Request Message

Description: If a node receives an ICMP Echo Request message it replies with an ICMP Echo Reply message. This is used for diagnostic purposes.
References: RFC 4443 [55] p. 13
Difficulty to Collect: Hard - The packet needs to be collected by a packet capturer.
Usefulness for an Investigation: Valuable - The message is totally legitimate in normal network traffic, often used to test network connections. However, it might be used to reconnoitre a network or as a covert channel

to transport information. Therefore, it might be valuable in an investigation.

A.6.33 Information I.NC.33 - Echo Reply Message

Description: If a node receives an ICMP Echo Request message it replies with an ICMP Echo Reply message. This is used for diagnostic purposes.
References: RFC 4443 [55] p. 14
Difficulty to Collect: Hard - The packet needs to be collected by a packet capturer.
Usefulness for an Investigation: Valuable - The message is totally legitimate in normal network traffic, often used to test network connections. However, it might be used to reconnoitre a network or as a covert channel to transport information. Therefore, it might be valuable in an investigation.

A.6.34 Information I.NC.34 - ICMP Error Message Rate Limit

Description: A node limits the number of ICMP error messages it sends during a certain time. This is often done with a token bucket.
References: RFC 4443 [55] p. 7
Difficulty to Collect: Easy - Linux defines the number of ICMP messages allowed to be sent within a jiffy (which is 4 ms by default [113]). The command `cat /proc/sys/net/ipv6/icmp/ratelimit` show the currently configured rate limit.
Usefulness for an Investigation: Valuable - To retrace a node's behaviour of sending ICMP message the value might be useful. Also, a possible DoS attack might force a node do send a large number of ICMP messages. To retrace the attack and show its effects the variable might be useful.

A.6.35 Information I.NC.35 - Traffic Filter

Description: The position and configuration of any traffic filter (aka firewall) that selects traffic based on ICMPv6 protocol fields.
References: -
Difficulty to Collect: Easy - Linux has a built-in traffic filtering mechanism that is active by default if not explicitly deactivated during kernel compilation. However, the default settings do not filter any traffic. The commands `ip6tables -t filter -L`, `ip6tables -t mangle -L` and `ip6tables -t raw -L` allow collecting the information under Linux.
Usefulness for an Investigation: Critical - Devices that filter traffic are significant for any investigation trying to retrace traffic flows.

A.6.36 Information I.NC.36 - Traffic Mangler

Description: The position and configuration of any device that changes ICMP protocol fields for any reason.
References: -

Difficulty to Collect: Easy - Linux has a built-in traffic mangling mechanism which is active by default if not explicitly deactivated during kernel compilation. However, the default settings do not modify any traffic. The commands `ip6tables -t mangle -L` and `ip6tables -t raw -L` allow collecting the information under Linux.

Usefulness for an Investigation: Critical - Devices that change traffic are significant for any investigation trying to retrace traffic flows.

A.6.37 Information I.NC.37 - Implementation Version

Description: The exact version of the software or hardware implementing the ICMPv6 protocol.

References: -

Difficulty to Collect: Easy - In the network scenario used, the exact kernel version defines the exact implementation of the ICMPv6 protocol. The command `uname -a` shows the exact kernel version running. The compilation parameter of the kernel can be retrieved with the command `zless /proc/config.gz`.

Usefulness for an Investigation: Critical - The exact version of software or hardware used, that implements the ICMPv6 protocol allows recreating the network setup and retrace network scenarios.

A.7 6in4

A.7.1 Information I.N6.01 - Tunnel Endpoint Address

Description: The IPv4 address of the 6in4 tunnelling mechanism. Typically source and destination addresses are configured.

References: RFC 4213 [63] p. 7

Difficulty to Collect: Easy - The command `ip tunnel list` allows collecting the information under Linux. It shows the local and remote tunnel endpoint address.

Usefulness for an Investigation: Critical - The tunnel is almost like a physical connection and the endpoint addresses identifies between which devices the connection is. Therefore, the tunnel endpoint addresses are critical for an investigation.

A.7.2 Information I.N6.02 - Tunnel Endpoint Type

Description: The type of tunnel endpoint (host or router).

References: RFC 4213 [63] p. 6

Difficulty to Collect: Easy - If the node forwards traffic not addressed to itself, it is a router, otherwise a host. The command `cat /proc/sys/net /ipv6/conf/*/forwarding` allows collecting the information under Linux.

Usefulness for an Investigation: Valuable - The information is already collected in I.NI.10.

A.7.3 Information I.N6.03 - Tunnel Type

Description: The kind of tunnel: Router-to-Router, Host-to-Router, Host-to-Host, Router-to-Host.

References: RFC 4213 [63] p. 6

Difficulty to Collect: Easy - When both endpoint addresses are known (I.N6.01) and the nodes using the addresses (I.NI.08), the node type of both systems can be determined (I.NI.10). With that information the tunnel type is also determined.

Usefulness for an Investigation: Valuable - The actual tunnel type is not very important for an investigation. It is more important what the endpoint of the tunnel is, meaning does it forwards the packet when it is not addressed to the endpoint itself or not. This is defined by the Node Type (I.NI.10).

A.7.4 Information I.N6.04 - Tunnel MTU Size

Description: The tunnel MTU. The MTU is either statically configured (between 1280 and 1480 byte) or dynamically determined (e.g. by MTU path discovery [114]).

References: RFC 4213 [63] p. 8

Difficulty to Collect: Easy - The command `ip link list` allows collecting the information under Linux. It shows the MTU per link.

Usefulness for an Investigation: Valuable - The tunnel MTU is not critical because IPv4, as the transport protocol, fragments packet that are too big for the tunnel. On the other hand, massive fragmentation and reassembling of packets creates load on the tunnel endpoint and opens an attack vector for a DoS attack.

A.7.5 Information I.N6.05 - Traffic Encapsulated

Description: Which traffic is sent into the tunnel and encapsulated to send to the tunnel endpoint.

References: RFC 4213 [63] p. 7

Difficulty to Collect: Easy - Linux selects traffic sent through the tunnel based on the routing (I.NI.09). If traffic is routed via the tunnel interface, it is encapsulated. The command `ip -6 route list` allows collecting the information under Linux. It shows the active kernel routing table.

Usefulness for an Investigation: Valuable - To know which traffic is sent through the tunnel is important to be able to retrace traffic flows. As the routing table is used to decide which traffic is sent through the tunnel it is redundant.

A.7.6 Information I.N6.06 - Traffic Decapsulated

Description: The traffic that is assumed to be from the other tunnel endpoint and is decapsulated. The receiving tunnel endpoint typically looks for packets with the next-header protocol 41 (IPv6). There might be some additional filtering which allows receiving such packets only from specific source IP addresses.

References: RFC 4213 [63] p. 7

Difficulty to Collect: Easy - Linux only checks if the traffic is received from the other tunnel endpoint address. However, the node might filter traffic (I.NI.11, I.NC.35).

Usefulness for an Investigation: Critical - Active traffic filters on the receiving side of the tunnel determine if the traffic is received or dropped. Therefore, it is crucial information to retrace network scenarios.

A.7.7 Information I.N6.07 - Tunnel MTU Mechanism

Description: If the tunnel MTU is statically configured or dynamically determine (e.g. by Path MTU Discovery [114]).

References: RFC 4213 [63] p. 8

Difficulty to Collect: Easy - Linux runs Path MTU Discovery by default. It is possible to disable it and set the MTU manually. The command `ip tunnel list` allows collecting the information under Linux. If the *nopmtudisc* flag is set, Path MTU Discovery is not active.

Usefulness for an Investigation: Valuable - The tunnel MTU is not critical because IPv4, as the transport protocol, fragments packet that are too big for the tunnel. On the other hand, massive fragmentation and reassembling of packets would create load on the tunnel endpoint and open an attack vector for a DoS attack. If there is a way to reducing the tunnel MTU further by manipulating the Path MTU Discovery, it further increases the load on the endpoints and make the attack more effective.

A.7.8 Information I.N6.08 - Tunnel Hop Limit

Description: The hop limit (time to live) in the encapsulating IPv4 header (the current suggested value is 64 [115]). An implementation might provide a mechanism to overwrite the default.

References: RFC 4213 [63] p. 11

Difficulty to Collect: Easy - The command `ip tunnel list` allows collecting the information under Linux. It shows the configured or inherited hop limit. The hop limit is taken from the physical interface used to transport the encapsulated packet.

Usefulness for an Investigation: Valuable - The value might be necessary to determine if the encapsulated packet was able to reach the tunnel endpoint. The information is redundant if the Packet Header is available (I.NI.01).

A.7.9 Information I.N6.09 - Strict Reverse Path Forwarding Check

Description: If the tunnel endpoint does strict RPF checks for encapsulated traffic received. If the tunnel endpoint does RPF checks, it would only accept an encapsulated packet when it receives the packet on the same interface as it would use to send a packet to the source of packet. This also includes the logging of packets that arrive on the wrong interface.

References: RFC 4213 [63] p. 15

Difficulty to Collect: Easy - Linux does currently not support RPF for IPv6 [116]. However, RPF can be implemented manually by filtering traffic accordingly (I.NI.11).

Usefulness for an Investigation: Valuable - As the RPF check would be implemented by traffic filter (I.NI.11) this information is redundant.

A.7.10 Information I.N6.10 - Ingress Filtering Tunnelled Traffic

Description: The traffic that is filtered by the tunnel endpoint after decapsulating the packet.

References: RFC 4213 [63] p. 17

Difficulty to Collect: Easy - Linux does not filter ingress traffic from the tunnel explicitly. However, the general traffic filtering capabilities can be used to do so (I.NI.11).

Usefulness for an Investigation: Valuable - As the RPF check would be implemented by traffic filter (I.NI.11) this information is redundant.

A.7.11 Information I.N6.11 - Link-Local Addresses

Description: The link-local addresses of the tunnel interfaces.

References: RFC 4213 [63] p. 17

Difficulty to Collect: Easy - The command `ip address list` allows collecting the information under Linux. It shows all active IP addresses on the interfaces.

Usefulness for an Investigation: Valuable - If the interface addresses (I.NI.10) are already collected, this information is redundant.

A.7.12 Information I.N6.12 - Tunnel IP Header

Description: The IPv4 header used by the tunnel.

References: -

Difficulty to Collect: Hard - The packet needs to be collected by a packet capturer.

Usefulness for an Investigation: Valuable - As the tunnel endpoints are statically configured it is easy to determine the expected IPv4 tunnel header. However, the header might be useful to retrace DoS MTU or spoofing attacks.

A.7.13 Information I.N6.13 - Implementation Version

Description: The exact version of the software or hardware implementing the 6in4 tunnel.

References: -

Difficulty to Collect: Easy - In the network scenario used, the exact kernel version defines the behaviour of the node terminating the 6in4 tunnel. The command `uname -a` shows the exact kernel version running. The compilation parameter of the kernel can be retrieved with the command `zless /proc/config.gz`.

Usefulness for an Investigation: Critical - The exact version of software or hardware used to implement the 6in4 tunnels allows recreating the network setup and retrace network scenarios.

A.7.14 Information I.N6.14 - ICMPv4 Packet

Description: The whole ICMPv4 packet.

References: -

Difficulty to Collect: Hard - The packet needs to be collected by a packet capturer.

Usefulness for an Investigation: Valuable - The ICMPv4 packet might be helpful in retracing ICMP attacks against tunnel endpoints.

A.7.15 Information I.N6.15 - IPv4 Routing Table

Description: The IPv4 routing table.

References: -

Difficulty to Collect: Easy - The command `ip route list` allows collecting the information under Linux. It shows the currently active routing table in the kernel.

Usefulness for an Investigation: Critical - The routing table decides how a tunnelled packet are forwarded. That is critical information for retracing network traffic flows involving tunnels.

A.8 GRE

A.8.1 Information I.NG.01 - Payload Type

Description: The type of protocol transported in the tunnel. For the purpose of this research only IPv6 is considered.

References: RFC 2784 [64] p. 2

Difficulty to Collect: Easy - The command `ip address list` allows collecting the information under Linux. The GRE payload depends on the protocol address configured on the interface. For example, if an IPv6 address is configured on the interface it transports IPv6 over GRE over IPv4.

Usefulness for an Investigation: Critical - Even though the scope of this research limits the transported protocols to IPv6 the information is critical to retrace traffic flows. For example, there might be an active GRE tunnel that does not transport IPv6. That could significantly change the result of retracing traffic flows.

A.8.2 Information I.NG.02 - GRE Header

Description: The GRE protocol header including the IPv4 header.
References: RFC 2784 [64] p. 2, RFC 2890 [65] p. 2
Difficulty to Collect: Hard - The packet needs to be collected by a packet capturer.
Usefulness for an Investigation: Valuable - As the tunnel endpoints are statically configured it is easy to determine the expected IPv4 and GRE tunnel header. However, the header might be useful to retrace DoS MTU or spoofing attacks (IP address or sequence number).

A.8.3 Information I.NG.03 - Out of Order Timer

Description: The timer defines how long a receiver should buffer packets that arrived ahead of the expected sequence number.
References: RFC 2890 [65] p. 4
Difficulty to Collect: Easy - Based on a quick look in the source code, the Linux kernel does not buffer any out of sequence packets but drops them (linux/net/ipv4/ip_gre.c line 647), therefore, no out of order timer is implemented.
Usefulness for an Investigation: Futile - The timer is not existing in the analysed systems.

A.8.4 Information I.NG.04 - Buffer Limit

Description: The number of packets a receiver is willing to buffer when packets arrive ahead of the expected sequence number.
References: RFC 2890 [65] p. 4
Difficulty to Collect: Easy - Based on a quick look in the source code, the Linux kernel does not buffer any out of sequence packets but drops them (linux/net/ipv4/ip_gre.c line 647). Therefore, no buffer limit is implemented.
Usefulness for an Investigation: Futile - The buffer limit is not existing in the analysed systems.

A.8.5 Information I.NG.05 - Tunnel Endpoint Address

Description: The IPv4 address of the GRE tunnelling mechanism. Typically source and destination addresses are configured.
References: IP(8) Linux [117]
Difficulty to Collect: Easy - The command `ip tunnel list` allows collecting the information under Linux. It shows the local and remote tunnel endpoint address.
Usefulness for an Investigation: Critical - The tunnel is almost like a physical connection and the endpoint addresses identifies between which devices the connection is. Therefore, the tunnel endpoint addresses are critical for an investigation.

A.8.6 Information I.NG.06 - Tunnel MTU Size

Description: The MTU of the tunnel.
References: IP(8) Linux [117]
Difficulty to Collect: Easy - The command `ip tunnel list` allows collecting the information under Linux. It shows the MTU per link.
Usefulness for an Investigation: Valuable - The tunnel MTU is not critical because IPv4, as the transport protocol, fragments packets that are too big for the tunnel. On the other hand, massive fragmentation and reassembling of packets create load on the tunnel endpoints and open an attack vector for a DoS attack.

A.8.7 Information I.NG.07 - Traffic Encapsulated

Description: The traffic that is sent into the tunnel and encapsulated.
References: -
Difficulty to Collect: Easy - Linux selects traffic sent through the tunnel based on the routing (I.NI.09). If traffic is routed via the tunnel interface, it is encapsulated. The command `ip -6 route list` allows collecting the information under Linux. It shows the active kernel routing table.
Usefulness for an Investigation: Valuable - To know which traffic is sent through the tunnel is important to be able to retrace traffic flows. As the routing table is used to decide which traffic is sent through the tunnel it is redundant.

A.8.8 Information I.NG.08 - Traffic Decapsulated

Description: The traffic that is assumed to be from the other tunnel endpoint and is decapsulated. The receiving tunnel endpoint typically looks for packets with the next-header protocol 47 (GRE). There might be some additional filtering which allows receiving such packets only from specific source IP addresses.
References: -
Difficulty to Collect: Easy - Linux only checks if the traffic is received from the other tunnel endpoint address. However, the node might filter traffic (I.NI.11, I.NC.35).
Usefulness for an Investigation: Critical - Active traffic filters on the receiving side of the tunnel determine if the traffic is received or dropped. Therefore, it is crucial information to retrace network scenarios.

A.8.9 Information I.NG.09 - Tunnel Features

Description: What optional features (checksum, key, sequence number) are active for the tunnel.
References: -
Difficulty to Collect: Easy - The command `ip tunnel list` allows collecting the information under Linux. It shows the GRE tunnel configuration with any optional feature configured.

Usefulness for an Investigation: Critical - The active features determine the behaviour of the tunnel and are, therefore, critical for an investigation.

A.8.10 Information I.NG.10 - Implementation Version

Description: The exact version of the software or hardware implementing the GRE tunnel.

References: -

Difficulty to Collect: Easy - In the network scenario used, the exact kernel version defines the behaviour of the node terminating the GRE tunnel. The command `uname -a` shows the exact kernel version running. The compilation parameter of the kernel can be retrieved with the command `zless /proc/config.gz`.

Usefulness for an Investigation: Critical - The exact version of software or hardware used to implement the GRE tunnels allows recreating the network setup and retrace network scenarios.

A.9 RIPng

A.9.1 Information I.NR.01 - Network Cost

Description: The cost of each traversed network (hop), the default is 1.

References: RFC 2080 [59] p. 3

Difficulty to Collect: Easy - The command `show ipv6 ripng` allows collecting the information in the Quagga shell (vtysh). It shows an overview of the RIPng configuration with the active interface and the cost associated with them.

Usefulness for an Investigation: Critical - The network cost (metric) is critical information to retrace the RIPng protocol. It is used to determine the best route to be installed in the routing table.

A.9.2 Information I.NR.02 - RIP Routers

Description: The routers with a RIP daemon installed and running.

References: RFC 2080 [59] p. 4

Difficulty to Collect: Easy - The command `show ipv6 ripng` allows collecting the information in the Quagga shell (vtysh). It shows an overview of the RIPng configuration or that RIPng is inactive.

Usefulness for an Investigation: Critical - To retrace the operation of the RIPng routing protocol and the building of the routing tables it is critical to know the systems that take part in the protocol.

A.9.3 Information I.NR.03 - Directly Connected Networks

Description: The networks a router is responsible for. The router announces their presence to other RIP routers.

The router associates with each network a metric, an address prefix and a prefix length. The metric is an integer between 1 and 15 with the default of 1.

References: RFC 2080 [59] p. 4

Difficulty to Collect: Easy - The command `show ipv6 ripng status` allows collecting the information in the Quagga shell (vtysh). It shows an overview of the RIPng configuration with the active interface and the prefixes associated with them.

Usefulness for an Investigation: Critical - To retrace the operation of the RIPng routing protocol and the building of the routing table it is critical to know which are announced in the protocol.

A.9.4 Information I.NR.04 - Routing Table

Description: Each RIP router has a routing table. For each reachable destination an entry with the following information is kept: the IPv6 prefix of the destination, the metric, the next hop, a flag that indicates that the information has changed recently, the timeout timer, the garbage-collection timer and the route tag.
The entries for the directly connected networks are created by the local router.
The timeout timer (default 180 s) defines when the route is recognised as invalid and the garbage-collection timer (default 120 s) when it is completely removed.

References: RFC 2080 [59] p. 3, RFC 2080 [59] p. 7, RFC 2080 [59] p. 9

Difficulty to Collect: Easy - The command `show ipv6 route` allows collecting the information in the Quagga shell (vtysh) and the command `ip -6 route list` allows collecting the information under Linux. While the first command shows the information from the viewpoint of the routing daemon, the second one shows all currently active routes in the kernel (I.NI.09). The two routing tables should correspond.

Usefulness for an Investigation: Valuable - The routing table is the result of the operation of the RIPng protocol. It is critical information in tracing traffic flows but redundant if the table is already collected in I.NI.09.

A.9.5 Information I.NR.05 - External Routes

Description: Routes propagated in the routing protocol originating outside the scope of the routing system. For example, a default route or routes learned by other routing protocols. The tag associated with the external routes when used.

References: RFC 2080 [59] p. 5, RFC 2080 [59] p. 7

Difficulty to Collect: Easy - The command `show ipv6 ripng` allows collecting the information in the Quagga shell (vtysh). It shows the origin of the routes and the tags. The origin is *redistributed* for an external route on the router that added the external route. On all other routers the origin is *normal*, which means learned by RIP.

Usefulness for an Investigation: Critical - To retrace the operation of the RIPng routing protocol and the building of the routing table it is critical to know which system imports and announces external routes into the RIPng domain.

A.9.6 Information I.NR.06 - Socket

Description: The combination of IPv6 address, protocol and port the daemon is listening on for RIPng datagrams and uses to send RIPng datagrams. The daemon can be listening on multiple sockets.

References: RFC 2080 [59] p. 5

Difficulty to Collect: Easy - The command `netstat -lnp | grep ripngd` allows collecting the information under Linux. It shows any process named ripngd listening on network ports.

Usefulness for an Investigation: Valuable - The information would be only interesting if there is doubt that the daemon was able to receive RIPng packets. In most other cases the result of the RIPng protocol, the routing table, would show that the daemon, indeed received RIPng packets and created corresponding entries in the routing table.

A.9.7 Information I.NR.07 - Passive Interface

Description: Network interfaces that are configured as passive (receive RIPng updates but do not send updates).

References: -

Difficulty to Collect: Easy - The command `show run` allows collecting the information in the Quagga shell (vtysh). In the subsection *router ripng* the command *passive-interface* shows the interfaces set passive.

Usefulness for an Investigation: Critical - To retrace the operation of the RIPng routing protocol and the building of the routing table it is critical to know which interfaces were configured to not send RIP updates.

A.9.8 Information I.NR.08 - Propagation Timer

Description: The timeout between two unsolicited response messages. The default is 30 seconds.

References: RFC 2080 [59] p. 9

Difficulty to Collect: Easy - The command `show ipv6 ripng status` allows collecting the information in the Quagga shell (vtysh). The line beginning with *Sending updates every* shows how often RIP updates are sent and when the next one is due.

Usefulness for an Investigation: Critical - To retrace the operation of the RIPng routing protocol and the building of the routing table it is critical to know the configured timer.

A.9.9 Information I.NR.09 - Link Local Addresses

Description: The link-local addresses of the router. A router might have multiple link-local addresses assigned to an interface.

References: RFC 2080 [59] p. 16

Difficulty to Collect: Easy - The command `show interface` allows collecting the information in the Quagga shell (vtysh) and the command `ip address list` allows collecting the information under Linux (I.NI.08).

Usefulness for an Investigation: Valuable - To understand the result of the RIPng protocol, the routing table, it is critical to know the link-local address of the routers. However, if they are already collected (I.NI.08), this information is redundant.

A.9.10 Information I.NR.10 - Split Horizon Setting

Description: The split horizon setting on the router's interfaces. The router might be configured to do no horizoning, split horizoning, and split horizoning with poison reverse.

References: RFC 2080 [59] p. 16

Difficulty to Collect: Easy - The command `show run` allows collecting the information in the Quagga shell (vtysh). The relevant configuration is found in the interface section and is called *ipv6 ripng split-horizon poisoned-reverse*.

Usefulness for an Investigation: Critical - To retrace the operation of the RIPng routing protocol and the building of the routing table it is critical to the split-horizon and poison-reverse settings of each interface taking part in the RIPng protocol.

A.9.11 Information I.NR.11 - Valid Neighbour List

Description: The list contains routers that are considered valid neighbours and, therefore, the router accepts updates from or sends updates to. The default is to accept updates from all neighbours.

References: RFC 2080 [59] p. 17

Difficulty to Collect: Easy - Quagga does not support filtering neighbours directly in the routing protocol configuration or any other network traffic filtering. However, is possible to achieve the filtering by the normal Linux traffic filtering mechanisms (I.NI.11) by blocking any UDP traffic of a specific neighbour to the RIPng port (521).

Usefulness for an Investigation: Valuable - To retrace the operation of the RIPng routing protocol and the building of the routing table it is critical to know if RIPng packets from certain systems are filtered. However, this information is redundant if the information about traffic filter (I.NI.11) is already collected.

A.9.12 Information I.NR.12 - Filter List

Description: A list of prefixes that are filtered in incoming and outgoing response messages. The list is typically assigned to an interface and the default is not to filter any prefixes.

References: RFC 2080 [59] p. 17

Difficulty to Collect: Easy - The command `show ipv6 ripng status` allows collecting the information in the Quagga shell (vtysh). The relevant lines begin with *Outgoing update filter list* and *Incoming update filter list*.

Usefulness for an Investigation: Critical - To retrace the operation of the RIPng routing protocol and the building of the routing table it is critical to know if certain network are filtered.

A.9.13 Information I.NR.13 - RIP Packet

Description: All RIPng packets sent by any router.

References: -

Difficulty to Collect: Hard - The packet needs to be collected by a packet capturer.

Usefulness for an Investigation: Valuable - With the knowledge of all the other critical information it is possible to determine the content of all RIP packets. However, the actual RIPng packets might be useful in an investigation where some information is missing or the result, the routing table, is unknown. In addition, the actual packet provides some redundancy in the evidence.

A.9.14 Information I.NO.14 - Implementation Version

Description: The exact version of the software or hardware implementing the RIPng protocol.

References: -

Difficulty to Collect: Easy - In the network setup used for this research, the Linux kernel forwards the network traffic while the routing daemon Quagga implements the routing protocol and populates the routing table. The exact version of the routing daemon Quagga can be collected with the command `show version` inside the Quagga shell (vtysh).

Usefulness for an Investigation: Critical - The exact version of software or hardware used, that implements the RIPng protocol allows recreating the network setup and retrace network scenarios.

A.10 OSPF

A.10.1 Information I.NO.01 - OSPF Routers

Description: The routers having a OSPF daemon installed and running. This includes information directly associated with them such as the Router ID.

References: RFC 5340 [61] p. 86

Difficulty to Collect: Easy - The command `show ipv6 ospf6` allows collecting the information in the Quagga shell (vtysh). It shows an overview of the OSPF configuration or that OSPFv3 is inactive. It also shows the Router ID.

Usefulness for an Investigation: Critical - To retrace the operation of the OSPFv3 routing protocol and the building of the routing table it is critical to know the systems taking part in the protocol.

A.10.2 Information I.NO.02 - Instance

Description: The OSPF instance with associated information such as the Instance ID. The default Instance ID is 0. The Instance ID is only used if multiple separate communities of OSPF routers reside inside on the same link.

References: RFC 5340 [61] p. 7

Difficulty to Collect: Easy - The command `show ipv6 ospf6 interface` allows collecting the information in the Quagga shell (vtysh). It shows the interfaces assigned to the different OSPF instance.

Usefulness for an Investigation: Critical - To retrace the operation of the OSPFv3 routing protocol and the building of the routing table it is critical to know the configured instances.

A.10.3 Information I.NO.03 - Interfaces

Description: The interfaces that are part of the OSPF routing domain. This includes information associated with them such as the Area ID, Instance ID, Interface ID, link-local address, a list of all prefixes configured on the link, the link LSA suppression and the options (V6-Bit, E-Bit, x-Bit, N-Bit, R-Bit, DC-Bit) the router announces on that interface. Also the cost associated with the interface, several timers (retransmission, transmit delay, hello interval, dead interval) and the router priority,

References: RFC 5340 [61] p. 6, RFC 5340 [61] p. 59, RFC 5340 [61] p. 88

Difficulty to Collect: Easy - The command `show ipv6 ospf6 interface` allows collecting the information in the Quagga shell (vtysh). It shows the interface ID, the link-local address and the prefixes configured on the link. The command `show ipv6 ospf6 linkstate` shows all the flags. Finally, the command `show ipv6 ospf6 database details` shows the complete link state database with all received options.
Quagga does not allow configuring link LSA suppression in the tested release.

Usefulness for an Investigation: Critical - To retrace the operation of the OSPFv3 routing protocol and the building of the routing table it is critical to know the interfaces participate in the protocol.

A.10.4 Information I.NO.04 - Areas

Description: The OSPF areas and associated information such as Area ID and Area Type (Stub, Totally Stubby, Not So Stubby, Totally Not So Stubby).

References: RFC 5340 [61] p. 87

Difficulty to Collect: Easy - The command `show ipv6 ospf6` allows collecting the information in the Quagga shell (vtysh). It shows an overview of the currently configure areas on the router. The area type can be determined by the running configuration (`show ipv6 ospf6`). The area type defines the other parameters mentioned in the RFC (ExternalRoutingCapability, StubDefaultCost, NSSATranslatorRole, TranslatorStabilityInterval, ImportSummaries)

Usefulness for an Investigation: Critical - To retrace the operation of the OSPFv3 routing protocol and the building of the routing table it is critical to know the configured areas.

A.10.5 Information I.NO.05 - Neighbour Table

Description: The OSPF neighbour table and associated information such as the neighbour's interface IDs, IP addresses, DR/BDR and link-local addresses.

References: RFC 5340 [61] p. 16

Difficulty to Collect: Easy - The command `show ipv6 ospf6 neighbor detail` allows collecting the information in the Quagga shell (vtysh). It shows the router IDs of currently active neighbours, the state of the neighbour relationship, the local and remote interface ID, the DR/BDR election and the link-local addresses of the neighbour.

Usefulness for an Investigation: Valuable - The neighbour table shows which routers successfully formed a neighbour relationship. Based on the physical layout of the network (I.DE.08) and the other critical OSPF information (I.NO.01, I.NO.02, I.NO.03, I.NO.04, I.NO.09, I.NO.10) this information can be concluded.

A.10.6 Information I.NO.06 - Link State Database

Description: The Link State Database is composed of LSAs and synchronised between adjacent routers.

References: RFC 5340 [61] p. 13

Difficulty to Collect: Easy - The command `show ipv6 ospf database detail` allows collecting the information in the Quagga shell (vtysh). It shows the complete link state database.

Usefulness for an Investigation: Valuable - The link state database contains the result of OSPF protocol. Based on the physical layout of the network (I.DE.08) and the other critical OSPF information (I.NO.01, I.NO.02, I.NO.03, I.NO.04, I.NO.09, I.NO.10, I.NO.12) this information can be concluded.

A.10.7 Information I.NO.07 - Routing Table

Description: The routing table resulting from the OSPF routing daemon.
References: RFC 5340 [61] p. 44
Difficulty to Collect: Easy - The command `show ipv6 route` allows collecting the information in the Quagga shell (vtysh) and the command `ip -6 route list` allows collecting the information under Linux. While the first command shows the information from the viewpoint of the routing daemon, the second shows all currently active routes in the kernel (I.NI.09). The two routing tables should correspond.
Usefulness for an Investigation: Valuable - The routing table contains the Dijkstra Algorithm and the link state database. Based on the link state database (I.NO.07) this information can be concluded. However, as the routing table is used to actually forward traffic, it might be useful in tracing network traffic flows.

A.10.8 Information I.NO.08 - Virtual Links

Description: Any configured virtual links and parameter associated with them such as the Router IDs of the routers the link is between and the Area ID which both routers are connected to (transit area).
References: RFC 5340 [61] p. 20, RFC 5340 [61] p. 91
Difficulty to Collect: Easy - Quagga does not support virtual-links for OSPFv3 in the tested release.
Usefulness for an Investigation: Critical - To retrace the operation of the OSPFv3 routing protocol and the building of the routing table it is critical to know the active virtual links.

A.10.9 Information I.NO.09 - Interface State

Description: The OSPF state of an interface: Waiting, DR, DR Other, Backup, Standby, Down.
References: RFC 5340 [61] p. 51
Difficulty to Collect: Easy - The command `show ipv6 ospf6 interface` allows collecting the information in the Quagga shell (vtysh). It shows the current state of the interfaces.
Usefulness for an Investigation: Critical - The interface state determines if actually LSAs are exchanged about and over that interface. Therefore, they are critical in an investigation to retrace the building of the routing table.

A.10.10 Information I.NO.10 - OSPF Packet

Description: The OSPF packet.
References: -
Difficulty to Collect: Hard - The packet needs to be collected by a packet capturer.

Usefulness for an Investigation: Valuable - With the knowledge of all the other critical information (I.DE.08, I.NO.01, I.NO.02, I.NO.03, I.NO.04, I.NO.09, I.NO.12) it is possible to determine the content of all OSPF packets. However, the actual OSPF packets might be useful in an investigation where some information is missing or the result, the routing table, is unknown.

A.10.11 Information I.NO.11 - Prefixes

Description: The prefixes a router announces into OSPF and the prefix options (NU-bit, LA-bit, x-bit, P-bit, DN-bit).

References: RFC 5340 [61] p. 6, RFC 5340 [61] p. 70

Difficulty to Collect: Easy - The command `show ipv6 ospf6` allows collecting the information in the Quagga shell (vtysh). It shows the currently configured prefixes on each interface. The command `show ipv6 \ ospf6 database details` shows the complete link state database with all received options.

Usefulness for an Investigation: Critical - To retrace the operation of the OSPFv3 routing protocol and the building of the routing table it is critical to know which prefixes are announced by which router.

A.10.12 Information I.NO.12 - Area Prefix Filter

Description: The concept of OSPF does not allow filtering prefixes inside an area because all routers need to have the same LS database to create a loop free topology. However, OSPF allows filtering on routers between areas.

References: RFC 5340 [61] p. 87

Difficulty to Collect: Easy - While the Quagga shell supports the commands `area 0.0.0.0 export-list`, `area 0.0.0.0 filter-list` and `area 0.0.0.0 import-list`, they do not appear in the running config (`show run`) and do not have any effect. It is concluded that the test release of Quagga does not support prefix filtering.

Usefulness for an Investigation: Critical - To retrace the operation of the OSPFv3 routing protocol and the building of the routing table it is critical to know which prefixes are filtered between areas.

A.10.13 Information I.NO.13 - Implementation Version

Description: The exact version of the software or hardware implementing the OSPFv3 protocol.

References: -

Difficulty to Collect: Easy - In the network setup used for this research, the Linux kernel forwards the network traffic while the routing daemon Quagga implements the routing protocol and populates the routing table. The exact version of the routing daemon Quagga can be collected with the command `show version` inside the Quagga shell (vtysh).

Usefulness for an Investigation: Critical - The exact version of software or hardware used, that implements the OSPFv3 protocol allows recreating the network setup and retrace network scenarios.

A.11 TCP

A.11.1 Information I.TT.01 - Segment Header

Description: The complete TCP segment header (source port, destination port, sequence and acknowledgment number, offset, flags, window, checksum, urgent pointer, options, padding).

References: RFC 793 [67] p. 15

Difficulty to Collect: Hard - The segment needs to be collected by a packet capturer.

Usefulness for an Investigation: Valuable - The source port is used to identify the sending process on the sending node. The destination port identifies the receiving process on the receiving node. The sequence and acknowledgment number provide reliability. The offset is needed to parse the header as it states the length of the TCP header. The flags are uses for the connection establishment and teardown and also to mark special segments. The window is used for the flow control and the checksum to recognise errors in the header or the payload.

A.11.2 Information I.TT.02 - Segment Payload

Description: The payload the TCP segment is carrying.

References: RFC 793 [67] p. 15

Difficulty to Collect: Hard - The segment needs to be collected by a packet capturer.

Usefulness for an Investigation: Valuable - The payload is passed to the upper layer and can, therefore, be inspected there.

A.11.3 Information I.TT.03 - Active Sockets

Description: The currently active TCP sockets on a node, including sockets in the listening state. A pair of sockets, one on the client side and one on the server side, uniquely identifies a connection between two applications.

References: RFC 793 [67] p. 5

Difficulty to Collect: Easy - The command `netstat -6ntpoeea` allows collecting the information in Linux. It shows all currently active TCP sockets (listening or established) with the corresponding inode, the username, programme ID and programme name that created the connection, and the number of packets in the receive and send queue.

Usefulness for an Investigation: Critical - It is critical for an investigation to know which application have active communications (active sockets) or are willing to create connections (listening sockets). This changes considerably how a node reacts to a receiving TCP segment.

A.11.4 Information I.TT.04 - Traffic Filter Rules

Description: The position and configuration of any traffic filter (aka firewall) that selects traffic based on TCP protocol fields.

References: -

Difficulty to Collect: Easy - Linux has a built-in traffic filtering mechanism that is active by default if not explicitly deactivated during kernel compilation. However, the default settings are not to filter any traffic. The commands `ip6tables -t filter -L`, `ip6tables -t mangle -L` and `ip6tables -t raw -L` allow collecting the current filter settings under Linux.

Usefulness for an Investigation: Critical - Devices that filter traffic are significant for any investigation trying to retrace network scenarios.

A.11.5 Information I.TT.05 - Traffic Filter Sessions

Description: Traffic filter often reach a filter decision based on the session state of a TCP session (e.g. allow traffic related to an establish TCP session on the outside interface). Therefore, traffic filter track TCP sessions and maintain their state.
This information includes the session tracking configuration (what sessions are tracked, what is the timeout of a session) and the currently active session and their state. Not supported by the network lab but a great add-on would be the traffic counter (packets and bytes) for the tracked sessions.

References: -

Difficulty to Collect: Easy - The command `cat /proc/sys/net/netfilter /nf_conntrack_tcp_timeout_*` allows collecting the different TCP timeout values and the command `conntrack -f ipv6 -p tcp -L` shows all currently tracked TCP sessions with their state.

Usefulness for an Investigation: Critical - The state of the session tracking allows retracing active session and filter decision made by the traffic filter. Therefore, it is critical for an investigation.

A.11.6 Information I.TT.06 - Traffic Mangler

Description: The position and configuration of any device that changes TCP fields for any reason.

References: -

Difficulty to Collect: Easy - Linux has a built-in traffic mangling mechanism which is active by default if not explicitly deactivated during kernel compilation. However, the default settings do not modify any traffic. The commands `ip6tables -t mangle -L` and `ip6tables -t raw -L` allow collecting the information under Linux..

Usefulness for an Investigation: Critical - Devices that change traffic are significant for any investigation trying to retrace traffic flows.

A.11.7 Information I.TT.07 - Implementation Version

Description: The exact version of the software or hardware implementing the TCP protocol.

References: -

Difficulty to Collect: Easy - In the network scenario used, the exact kernel version defines the exact implementation of the TCP protocol. The command `uname -a` shows the exact kernel version running. The compilation parameter of the kernel can be retrieved with the command `zless /proc/config.gz`. As the protocol is highly configurable, the current settings are also important. They can be retrieved with the commando `cat /proc/sys/net/ipv4/tcp*`

Usefulness for an Investigation: Critical - The exact version of software or hardware used, that implements the TCP protocol allows recreating the network setup, comprehend connection tracking the traffic filtering, verify receiving ports and ultimately retrace network scenarios.

A.11.8 Information I.TT.08 - TCP Log

Description: The log of the TCP implementation. The content, form and location of the log heavily depend on the configuration of the DHCP server. The log might also be rotated, that is deleted, after a while.

References: -

Difficulty to Collect: Easy - The Linux kernel implements the TCP protocol. The kernel logs to Syslog with the facility *kern*. Rsyslog is configured to write the messages to `/var/log/kern.log`.

Usefulness for an Investigation: Valuable - The log indicates error is received traffic and observed abnormalities. Therefore, it might provide useful indications for an investigation.

A.12 UDP

A.12.1 Information I.TU.01 - Datagram Header

Description: The complete UDP datagram header (source port, destination port, length, checksum).

References: RFC 768 [66] p. 1

Difficulty to Collect: Hard - The datagram needs to be collected by a packet capturer.

Usefulness for an Investigation: Valuable - The source port is used to identify the sending process on the sending node. However, the RFC allows setting the source port to 0. The destination port identifies the receiving process on the receiving node. The length and checksum are important for any parser (e.g. traffic filter).

A.12.2 Information I.TU.02 - Datagram Payload

Description: The payload the UDP datagram is carrying.
References: RFC 768 [66] p. 1
Difficulty to Collect: Hard - The datagram needs to be collected by a packet capturer.
Usefulness for an Investigation: Valuable - The payload is passed to the upper layer and can, therefore, be inspected there.

A.12.3 Information I.TU.03 - Active Sockets

Description: The currently active UDP sockets on a node, including sockets in the listening state.
References: RFC 768 [66] p. 2
Difficulty to Collect: Easy - The command `netstat -6nupoeea` allows collecting the information in Linux. It shows all currently active UDP sockets (listening or established) with the corresponding inode, the username, programme ID and programme name that created the connection, and the number of packets in the receive and send queue.
Usefulness for an Investigation: Critical - It is critical for an investigation to know which application have active communications (active sockets) or are willing to create connections (listening sockets). This changes considerably how a node reacts to a receiving UDP datagram.

A.12.4 Information I.TU.04 - Traffic Filter Rules

Description: The position and configuration of any traffic filter (aka firewall) that selects traffic based on UDP protocol fields.
References: -
Difficulty to Collect: Easy - Linux has a built-in traffic filtering mechanism that is active by default if not explicitly deactivated during kernel compilation. However, the default settings are not to filter any traffic. The commands `ip6tables -t filter -L`, `ip6tables -t mangle -L` and `ip6tables -t raw -L` allow collecting the current filter settings under Linux.
Usefulness for an Investigation: Critical - Devices that filter traffic are significant for any investigation trying to retrace network scenarios.

A.12.5 Information I.TU.05 - Traffic Filter Sessions

Description: Traffic filter often reach a filter decision based on the session state of a UDP session (e.g. allow traffic related to an establish UDP session on the outside interface but not on the inside interface). Therefore, traffic filter track UDP sessions and maintain their state. However, as there is no session establishment and session teardown the use a timeout mechanism to track UDP session.
This information includes the session tracking configuration (what sessions are tracked, what is the timeout of a session) and the currently

active session and their state. Not supported by the network lab but a great add-on would be the traffic counter (packets and bytes) for the tracked sessions.

References: -

Difficulty to Collect: Easy - The commands `cat /proc/sys/net/netfilter \ /nf_conntrack_udp_timeout` and `cat /proc/sys/net/netfilter \ /nf_conntrack_udp_timeout_stream` allow collecting the timeout values and the command `conntrack -f ipv6 -p udp -L` shows all currently tracked UDP sessions with their state.

Usefulness for an Investigation: Critical - The state of the session tracking allows retracing active session and filter decision made by the traffic filter. Therefore, it is critical for an investigation.

A.12.6 Information I.TU.06 - Traffic Mangler

Description: The position and configuration of any device that changes UDP protocol fields for any reason.

References: -

Difficulty to Collect: Easy - Linux has a built-in traffic mangling mechanism which is active by default if not explicitly deactivated during kernel compilation. However, the default settings do not modify any traffic. The commands `ip6tables -t mangle -L` and `ip6tables -t raw -L` allow collecting the information under Linux.

Usefulness for an Investigation: Critical - Devices that change traffic are significant for any investigation trying to retrace traffic flows.

A.12.7 Information I.TU.07 - Implementation Version

Description: The exact version of the software or hardware implementing the UDP protocol.

References: -

Difficulty to Collect: Easy - In the network scenario used, the exact kernel version defines the exact implementation of the UDP protocol. The command `uname -a` shows the exact kernel version running. The compilation parameter of the kernel can be retrieved with the command `zless /proc/config.gz`.

Usefulness for an Investigation: Critical - The exact version of software or hardware used, that implements the UDP protocol allows recreating the network setup, comprehend connection tracking the traffic filtering, verify receiving ports and ultimately retrace network scenarios.

A.12.8 Information I.TU.08 - UDP Log

Description: The log of the UDP implementation. The content, form and location of the log heavily depend on the configuration of the DHCP server. The log might also be rotated, that is deleted, after a while.

References: -

Difficulty to Collect: Easy - The Linux kernel implements the UDP protocol. The kernel logs to Syslog with the facility *kern*. Rsyslog is configured to write the messages to `/var/log/kern.log`.

Usefulness for an Investigation: Valuable - The log indicates error is received traffic and observed abnormalities. Therefore, it might provide useful indications for an investigation.

A.13 DNS

A.13.1 Information I.AN.01 - DNS Traffic

Description: The data the DNS client and server exchange (the UDP or TCP payload - I.TU.02/I.TT.02).

References: RFC 1035 [74]

Difficulty to Collect: Hard - The packet needs to be collected by a packet capturer.

Usefulness for an Investigation: Valuable - The content of the DNS messages can be retraced with the client and server configuration. However, if there is no access to the client or the server, the actual DNS messages are helpful in retracing DNS network scenarios. Furthermore, only the configuration does not show if a client actually resolved a name.

A.13.2 Information I.AN.02 - Traffic Filter

Description: The position and configuration of any traffic filter (aka firewall) that selects traffic based on DNS protocol fields.

References: -

Difficulty to Collect: Easy - Linux has a built-in traffic filtering mechanism that is active by default if not explicitly deactivated during kernel compilation. However, the default settings are not to filter any traffic. The commands `ip6tables -t filter -L`, `ip6tables -t mangle -L` and `ip6tables -t raw -L` allow collecting the current filter settings under Linux.

Usefulness for an Investigation: Critical - Devices that filter traffic are significant for any investigation trying to retrace DNS network scenarios.

A.13.3 Information I.AN.03 - Traffic Mangler

Description: The position and configuration of any device that changes DNS data passing by for any reason.

References: -

Difficulty to Collect: Easy - Linux has a built-in traffic mangling mechanism which is active by default if not explicitly deactivated during kernel compilation. However, the default settings do not modify any traffic. The commands `ip6tables -t mangle -L` and `ip6tables -t raw -L` allow collecting the information under Linux.

Usefulness for an Investigation: Critical - Devices that change DNS traffic are significant for any investigation trying to retrace DNS network scenarios.

A.13.4 Information I.AN.04 - DNS Server

Description: The nodes with a DNS server installed including the exact version and the configuration of the server. The configuration includes the IP addresses and transport protocols the server receives and sends DNS query and replies, the zone configuration with all resource records, the refresh and cache policy and pointers to other name servers and root servers. In addition, part of the configuration is the server's function (authoritative, caching or both) and if the server supports recursive queries or not.
The configuration could be divided into separate pieces of information. However, as usually the whole configuration is needed to retrace a scenario and the configuration can easily be extracted as a whole, there is no advanced of doing so.

References: RFC 1033 [105] p.1, RFC 1034 [73] p.6, RFC 4472 [88] p.4

Difficulty to Collect: Easy - The command `ps aux | grep named` allows determining if a DNS server is running on a node and the command `named -v` shows its version. The configuration of the daemon can be found in the files in the folder `/etc/bind`. The main configuration file is `named.conf` and the other files are included in appropriate sections. The start parameter of the daemon are in the file `/etc/default/bind9`.

Usefulness for an Investigation: Critical - To retrace the operation of the DNS protocol it is critical to identify the involved systems.

A.13.5 Information I.AN.05 - DNS Client

Description: The nodes that have a DNS client (resolver) installed, the exact version of it and the configuration (DNS server, if the client follows referrals, caching policy, domain name search list for the completion of relative names).

References: RFC 1034 [73] p.6, RFC 1123 [77] p.83

Difficulty to Collect: Easy - The file `/etc/resolv.conf` contains the list of available DNS servers and the domain search list. It may also contain other configuration such as the local domain name, a network sort list and a variety of options. The file `/etc/nsswitch.conf` controls what means, including DNS, are used to resolve hostnames and the order they are used. The resolver under Linux is part of the standard library (libc). Therefore, the library version defines the resolver. The command `dpkg -l | grep libc6` shows the version of the standard library.

Usefulness for an Investigation: Critical - To retrace the operation of the DNS protocol it is critical to identify the systems involved.

A.13.6 Information I.AN.06 - DNS Server Cache

Description: The content of the DNS Server cache.
References: RFC 1034 [73] p.6
Difficulty to Collect: Medium - The command `rndc dumpdb -cache` create a file `/var/cache/bind/named_dump.db` containing the current DNS cache. Unfortunately, the cache dump does not provide a way to determine when an entry was added into the cache as it only contains how many seconds an entry is still valid.
Usefulness for an Investigation: Valuable - The content of the DNS cache at a specific time allows determining the reply a DNS client received when the exact time is known a client sent a DNS query to the server.

A.13.7 Information I.AN.07 - DNS Server Log

Description: The log of the DNS server's activity. The content, form and location of the log heavily depend on the configuration of the DNS server. The log might also be rotated, that is deleted, after a while.
References: `http://www.zytrax.com/books/dns/ch7/logging.html`
Difficulty to Collect: Easy - The setup used in this research writes all activity to the file `/var/log/bind.log`.
Usefulness for an Investigation: Critical - The log allows retracing the DNS server activities.

A.13.8 Information I.AN.08 - DNS Client Cache

Description: The content of the DNS cache. The standard resolver (DNS client) used in this research does not support caching. An additional daemon would need to be installed to support client side caching (nscd). This scenario is ignored in this research.
References: RFC 1034 [73] p.6
Difficulty to Collect: Hard - The systems used in the lab network do not cache DNS replies.
Usefulness for an Investigation: Valuable - The content of the DNS cache allows determining the name to IP address mapping a client used at a specific time.

A.14 DHCP

A.14.1 Information I.AH.01 - DHCP Traffic

Description: The data the DHCP client and server exchange (the UDP datagram payload - I.TU.02).
References: RFC 3315 [93]
Difficulty to Collect: Hard - The packet needs to be collected by a packet capturer.

Usefulness for an Investigation: Valuable - The content of the DHCP messages can be retraced with the client and server configuration. However, if there is no access to the client or the server, the actual DHCP messages are helpful in retracing DHCP network scenarios. The timestamp when a client requested an IP address can be found in the Bindings (I.AH.07).

A.14.2 Information I.AH.02 - Traffic Filter

Description: The position and configuration of any traffic filter (aka firewall) that selects traffic based on the DHCPv6 protocol fields.

References: -

Difficulty to Collect: Easy - Linux has a built-in traffic filtering mechanism that is active by default if not explicitly deactivated during kernel compilation. However, the default settings are not to filter any traffic. The commands `ip6tables -t filter -L`, `ip6tables -t mangle -L` and `ip6tables -t raw -L` allow collecting the current filter settings under Linux.

Usefulness for an Investigation: Critical - Devices that filter traffic are significant for any investigation trying to retrace DHCP network scenarios.

A.14.3 Information I.AH.03 - Traffic Mangler

Description: The position and configuration of any device that changes DHCPv6 data passing by for any reason.

References: -

Difficulty to Collect: Easy - Linux has a built-in traffic mangling mechanism which is active by default if not explicitly deactivated during kernel compilation. However, the default settings do not modify any traffic. The commands `ip6tables -t mangle -L` and `ip6tables -t raw -L` allow collecting the information under Linux.

Usefulness for an Investigation: Critical - Devices that change DHCP traffic are significant for any investigation trying to retrace DHCP network scenarios.

A.14.4 Information I.AH.04 - DHCP Server

Description: The nodes with a DHCP server installed on including the exact version and the configuration of the server. The configuration of the DHCP server includes the administrative policy and what to reply to client requests (what IP address and other options to offer clients, the preference value used, if reconfiguration is supported, the supported modes - stateless or stateful).

References: RFC 3736 [95] p.1

Difficulty to Collect: Easy - The command `ps aux | grep dhcpd` allows determining if a DHCP server is running on a node and the command `dhcpd --version` shows its version. The DHCP server configuration is found in the files `/etc/dhcp/dhcpd.conf`, `/etc/default/isc-dhcp-server` and `/etc/init.d/isc-dhcp-server`.

Usefulness for an Investigation: Critical - To retrace the operation of the DHCP protocol it is critical to identify the involved systems. Based on the DHCP server's behaviour and the addresses it assigns clients it is possible to reconstruct the server configuration. However, aligning the observed behaviour with the actually configuration creates more credible evidence if a DHCP network scenario is retraced.

A.14.5 Information I.AH.05 - DHCP Relay Agent

Description: The nodes with a DHCP relay agent installed and the exact version of it.

References: RFC 3315 [93] p.6

Difficulty to Collect: Easy - The command `ps aux | grep dhcp6r` allows determining if a DHCP relay agent is running on a node and the command `dpkg -l | grep wide-dhcpv6-relay` shows its version

Usefulness for an Investigation: Critical - To retrace the operation of the DHCP protocol it is critical to identify the involved systems.

A.14.6 Information I.AH.06 - DHCP Client

Description: The nodes with a DHCP client installed, the exact version of it and its configuration.

References: RFC 3315 [93] p.6

Difficulty to Collect: Easy - The command `ps aux | grep dhcp6c` allows determining if a DHCP client is running on a node and the command `dpkg -l | grep wide-dhcpv6-client` shows its version. The configuration is found in the file `/etc/wide-dhcpv6/dhcp6c.conf`.

Usefulness for an Investigation: Critical - To retrace the operation of the DHCP protocol it is critical to identify the involved systems.

A.14.7 Information I.AH.07 - Binding Database

Description: Addresses or configuration information assigned to a specific client. Policy based information, for example assigned to all nodes on the same link, is not part of the binding.

References: RFC 3315 [93] p.10

Difficulty to Collect: Easy - The current bindings are stored in the file `/var/lib/dhcp/dhcpd6.leases` on the DHCP server. As the client uses the information received by the DHCP server, it can be found in several configuration files or current settings on the client as well. The IP address assigned by the server, for example, can be found in the currently assigned IP addresses (I.NI.08) and the DNS server in the file `/etc/resolv.conf`. The client might ignore some information returned by the DHCP server.

Usefulness for an Investigation: Critical - The actual binding and the use of the information is critical to retrace DHCP network scenarios because it shows what information the client actually used.

A.14.8 Information I.AH.08 - DHCP Unique Identifiers

Description: The DHCP Unique Identifier (DUID) used by each DHCP client and server including the DUID type used.

References: RFC 3315 [93] p.19, `http://tldp.org/HOWTO/Linux+IPv6-HOWTO/hints-daemons-isc-dhcp.html`

Difficulty to Collect: Easy - The server DUID can be found in the head of the lease database (`head /var/lib/dhcp/dhcpd6.leases`) and the client DUID can be extracted with the command `hexdump -e \ '"%07.7_ax " 1/2 "%04x" " " 14/1 "%02x:" "\n"' /var/lib \ /dhcpv6/dhcp6c_duid`. Further, it can be extracted on the server from the address bindings (I.AH.08). The server DUID is stored in octal form.

Usefulness for an Investigation: Critical - The DUID allows assigning a DHCP message to a client. Further, the DUID can be aligned with the link-local IP addresses a client uses.

A.14.9 Information I.AH.09 - Identity Association ID

Description: An Identity Association ID (IAID) identifies a group of IP addresses that are managed together by the server or client. A client often uses an IAID for the set of IP addresses of one interface. A client could use multiple IAIDs for a single interface. The IAID does not change across reboots of the client.

References: RFC 3315 [93] p.23

Difficulty to Collect: Easy - The IAID can be set in the WIDE DHCP client configuration file `/etc/wide-dhcpv6/dhcp6c.conf`. It can also be extracted on the DHCP server from the address bindings (I.AH.08).

Usefulness for an Investigation: Critical - The IAID allows aligning IP addresses with binding on the DHCP server. Therefore, they are critical to retrace DHCP network scenarios.

A.14.10 Information I.AH.10 - DHCP Authentication

Description: Is DHCP authentication active and the keys that are configured including the keys used to verify the authenticity of DHCP messages.

References: RFC 3315 [93] p.65

Difficulty to Collect: Hard - The Internet Systems Consortium DHCP server does not support authenticated at the current time. The Wide DHCP client does support authentication. The authentication and the keys are configure in the client configuration file `/etc/wide-dhcpv6/dhcp6c.conf`.

Usefulness for an Investigation: Critical - If DHCP authentication is used, the current settings (authentication method and keys) are critical to retrace DHCP network scenarios because it allows verifying the messages.

A.14.11 Information I.AH.11 - DHCP Server Log

Description: The log of the DHCP server's activity. The content, form and location of the log heavily depend on the configuration of the DHCP server. The log might also be rotated, that is deleted, after a while.

References: -

Difficulty to Collect: Easy - The setup used in this research writes all activity to the file `/var/log/dhcp.log`.

Usefulness for an Investigation: Critical - To log allows retracing the DHCP server activities.

A.14.12 Information I.AH.12 - DHCP Relay Agent Log

Description: The log of the DHCP relay agent's activity. The content, form and location of the log heavily depend on the configuration of the DHCP relay agent. The log might also be rotated, that is deleted, after a while.

References: -

Difficulty to Collect: Easy - The setup used in this research writes all activity to the file `/var/log/daemon.log`.

Usefulness for an Investigation: Critical - To log allows retracing the DHCP relay agent's activities.

Appendix B

Network Scenarios

This appendix presents the network scenarios used in this research. It starts with an overview of all scenarios and introduces the network used to run them. Afterwards it presents each scenario in detail with a description and a flow diagram, a discussion of the scenario, a list of the required information necessary to retrace the scenario and possible dependencies on other scenarios.

B.1 Overview

Table B.1 presents an overview of all network scenarios used in this thesis. The network used to run the scenarios is shown in Figure B.1 on page 157.

Table B.1: Overview of all scenarios used in this research with the layer (D=Data-Link, N=Network, T=Transport, A=Application), the intent (G=Genuine, M=Malicious) and the purpose (T=Training, V=Verification).

Scenario	Description	Layer	Protocol	Intent	Purpose
S.DG.01	Send a Frame	D	Ethernet	G	T
S.DG.02	Establishing Spanning Tree	D	Bridge	G	T
S.DG.03	Send Frame over Trunk	D	VLAN	G	T
S.DM.01	Device is Removed or Destroyed	D	Ethernet	M	T
S.DM.02	Link Unplugged	D	Ethernet	M	V
S.DM.03	MAC Flooding	D	Bridge	M	T
S.DM.04	MAC Spoofing	D	Bridge	M	T
S.DM.05	RSTP Root Claim	D	Bridge	M	V
S.DM.06	RSTP Eternal Root Election	D	Bridge	M	V
S.DM.07	RSTP Topology Change Flooding	D	Bridge	M	T
S.DM.08	VLAN Hopping	D	VLAN	M	T
S.DM.09	Occupying Link	D	Ethernet	M	V
S.DM.10	Administrative Access	D	Ethernet	M	T
S.NG.01	Enable and Configure IP	N	IPv6	G	T

Continued on next page

Scenario	Description	Layer	Protocol	Intent	Purpose
S.NG.02	Send a Packet to an On-Link Host	N	IPv6	G	T
S.NG.03	Send a Packet to an Off-Link Host	N	IPv6	G	T
S.NG.04	Send Packet to a Non-Existing Host	N	ICMPv6	G	T
S.NG.05	Send a Too Large Packet	N	ICMPv6	G	T
S.NG.06	Send a Packet with a Small Hop Limit	N	ICMPv6	G	V
S.NG.07	Send a Packet with Unknown Parameter	N	ICMPv6	G	V
S.NG.08	Send an ICMP Echo Request	N	ICMPv6	G	V
S.NG.09	Send Traffic through a 6in4 Tunnel	N	6in4	G	T
S.NG.10	Send Traffic through a GRE Tunnel	N	GRE	G	T
S.NG.11	Establish Routing with RIPng	N	RIPng	G	V
S.NG.12	Establish Routing with OSPFv3	N	OSPF	G	T
S.NM.01	Send a Packet with a Spoofed Source Address	N	IPv6	M	T
S.NM.02	Circumvent Traffic Filtering with the Source Routing Header	N	IPv6	M	T
S.NM.03	Overlapping IP Fragments	N	IPv6	M	T
S.NM.04	DoS IP Fragment Reassembler	N	IPv6	M	T
S.NM.05	Unknown Extension Header	N	IPv6	M	V
S.NM.06	DoS attack with excessive Hop-by-Hop Options	N	IPv6	M	T
S.NM.07	IP Options as a Covert Channel	N	IPv6	M	T
S.NM.08	Specially crafted Header Option	N	IPv6	M	T
S.NM.09	Administrative Access	N	IPv6	M	T
S.NM.10	DoS with the Router Alert Option	N	IPv6	M	T
S.NM.11	DoS Last Hop Router	N	ICMPv6	M	V
S.NM.12	Spoof NA messages	N	ICMPv6	M	T
S.NM.13	Spoof NS messages	N	ICMPv6	M	T
S.NM.14	DoS Neighbour Cache	N	ICMPv6	M	T
S.NM.15	Spoof RA messages	N	ICMPv6	M	T
S.NM.16	Clean the Default Router List	N	ICMPv6	M	T
S.NM.17	Flood the Default Router List	N	ICMPv6	M	T
S.NM.18	Lower Hop-Count	N	ICMPv6	M	T
S.NM.19	DoS Auto-configuration	N	ICMPv6	M	V
S.NM.20	Traffic Redirect with ICMP Redirect Messages	N	ICMPv6	M	V
S.NM.21	DoS with the ICMP Packet Too Big message	N	ICMPv6	M	V
S.NM.22	DoS with ICMP Error Messages	N	ICMPv6	M	V
S.NM.23	Covert Channel with ICMP messages	N	ICMPv6	M	T

Continued on next page

Scenario	Description	Layer	Protocol	Intent	Purpose
S.NM.24	DoS with ICMP message flooding	N	ICMPv6	M	T
S.NM.25	Spoof Tunnel Packet Source	N	6in4/GRE	M	T
S.NM.26	IPv4 Reassembly Attack against the Tunnel Endpoint	N	6in4/GRE	M	T
S.NM.27	Extend Attack Reach with Encapsulated Packet	N	6in4/GRE	M	T
S.NM.28	Circumvent Traffic Filtering	N	6in4/GRE	M	T
S.NM.29	DoS the tunnel with malicious payload addressing	N	6in4/GRE	M	V
S.NM.30	Lower Tunnel MTU	N	6in4/GRE	M	V
S.NM.31	DoS on GRE tunnels with Sequence Numbers	N	GRE	M	V
S.NM.32	Redirect Traffic with more Specific Routes	N	RIPng	M	V
S.NM.33	Redirect Traffic with Lower Metric	N	RIPng	M	T
S.NM.34	DoS RIPng Router with Requests	N	RIPng	M	T
S.NM.35	DoS RIPng Routers with Triggered Updates	N	RIPng	M	T
S.NM.36	Redirect Traffic with more Specific Routes	N	OSPF	M	T
S.NM.37	DoS OSPF Router with Topology Changes	N	OSPF	M	V
S.NM.38	DoS OSPF Router by Flooding Hello Packets	N	OSPF	M	T
S.NM.39	DoS OSPF Router by Spoofed Hello Packets	N	OSPF	M	T
S.TG.01	Send an UDP Segment	T	UDP	G	T
S.TG.02	Send UDP Segment to closed Port	T	UDP	G	T
S.TG.03	UDP Stream	T	UDP	G	T
S.TG.04	Establish a TCP Session	T	TCP	G	T
S.TG.05	Initiate a TCP Session to Closed Port	T	TCP	G	T
S.TM.01	UDP Flood Attack	T	UDP	M	T
S.TM.02	UDP Endless Stream	T	UDP	M	T
S.TM.03	Crash UDP Parser	T	UDP	M	T
S.TM.04	UDP Hide Sending Process	T	UDP	M	T
S.TM.05	TCP Fingerprinting Header Fields	T	TCP	M	T
S.TM.06	TCP Fingerprinting by Retransmission Timeout Sampling	T	TCP	M	T
S.TM.07	DoS with Push Flag	T	TCP	M	V
S.TM.08	SYN Flooding Attack	T	TCP	M	T

Continued on next page

Scenario	Description	Layer	Protocol	Intent	Purpose
S.TM.09	Connection Forgery Attack	T	TCP	M	V
S.TM.10	Connection Flooding Attack (Naptha)	T	TCP	M	T
S.TM.11	Overlong TCP Option	T	TCP	M	T
S.TM.12	Blind In-Window Attack	T	TCP	M	V
S.TM.13	FIN-WAIT-2 Flooding Attack	T	TCP	M	V
S.TM.14	Resource Exhaustion Attack (Netkill)	T	TCP	M	T
S.TM.15	TCP Reassembly Buffer Attack	T	TCP	M	T
S.TM.16	Overlapping TCP segments	T	TCP	M	V
S.TM.17	ACK Division Attack	T	TCP	M	T
S.TM.18	Duplicated ACK Forgery	T	TCP	M	V
S.TM.19	Optimistic ACKing	T	TCP	M	T
S.TM.20	Blind Throughput-Reduction Attack	T	TCP	M	V
S.TM.21	Blind Flooding Attack	T	TCP	M	T
S.TM.22	Blind Connection Reset with RST Flag	T	TCP	M	V
S.TM.23	Blind Data-Injection Attack	T	TCP	M	T
S.TM.24	TCP Port Scan	T	TCP	M	V
S.TM.25	Blind Performance-Degrading Attack with ICMP	T	TCP	M	T
S.TM.26	Blind Connection-Reset Attack with ICMP	T	TCP	M	V
S.TM.27	TCP-based Traceroute	T	TCP	M	T
S.AG.01	Resolve a Name from a Local Zone	A	DNS	G	T
S.AG.02	Resolve a Name from a Remote Zone	A	DNS	G	T
S.AG.03	Request Zone Transfer	A	DNS	G	V
S.AG.04	DHCP Information Request	A	DHCP	G	T
S.AG.05	DHCP Address Request	A	DHCP	G	V
S.AM.01	DNS Reply Spoofing	A	DNS	M	T
S.AM.02	DNS Cache Poisoning	A	DNS	M	T
S.AM.03	DNS Reflector Attack	A	DNS	M	T
S.AM.04	Rouge DHCP Server	A	DHCP	M	V
S.AM.05	Spoof DHCP Release Message	A	DHCP	M	T
S.AM.06	DoS DHCP Server	A	DHCP	M	V
S.AM.07	DoS DHCP Client during Solicit	A	DHCP	M	T
S.AM.08	DoS DHCP Client during Renew	A	DHCP	M	T

B.2 Network Layout

Figure B.1 shows the network layout used to run the scenarios. The hosts H1 - H3 are genuine hosts in the network, I1 is a DNS/DHCP server and M1-M2 are malicious hosts. S1 - S3 are switches and R1 - R3 are routers. The different devices are connected with each other with the links L1 - L14. The number assigned to each host (H1, H2, H3, I1, M1, M2) is used to systematically assign data-link layer addresses and network layer address to them. This is only done for convenience and to easier identify a host in network traces.

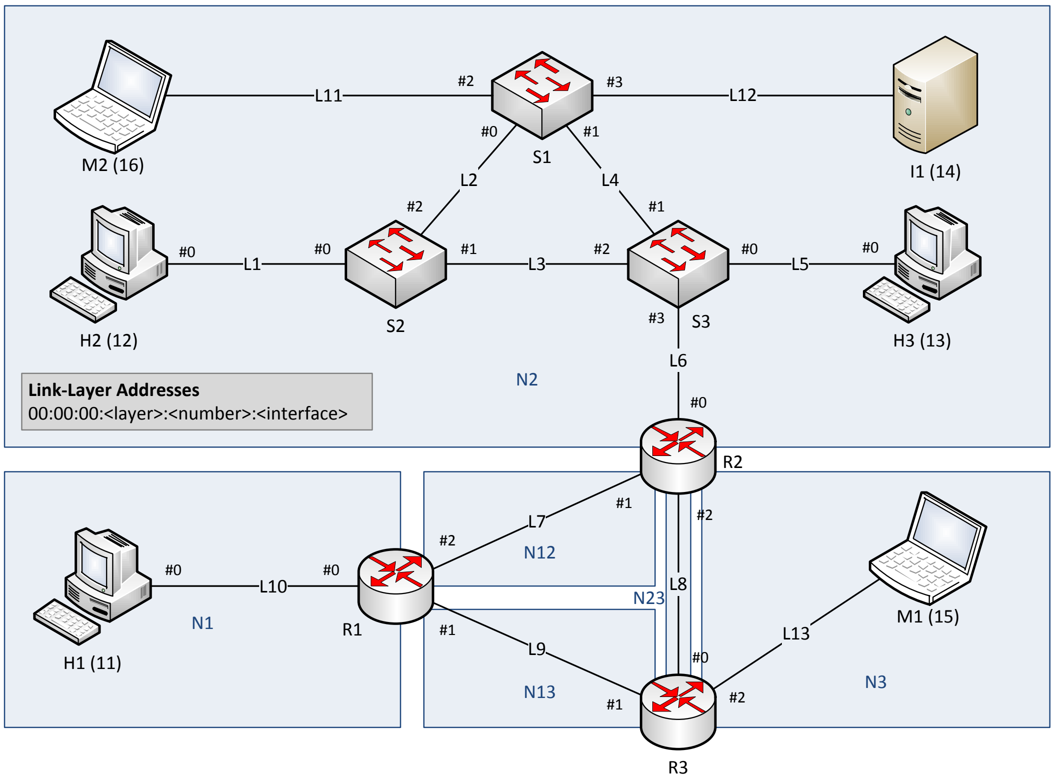

Figure B.1: The lab network layout

B.3 Data-Link Layer Scenarios

B.3.1 Sending a Frame (S.DG.01)

Description

H2 sends a frame to H3. The frame is switched by S2 and S3.

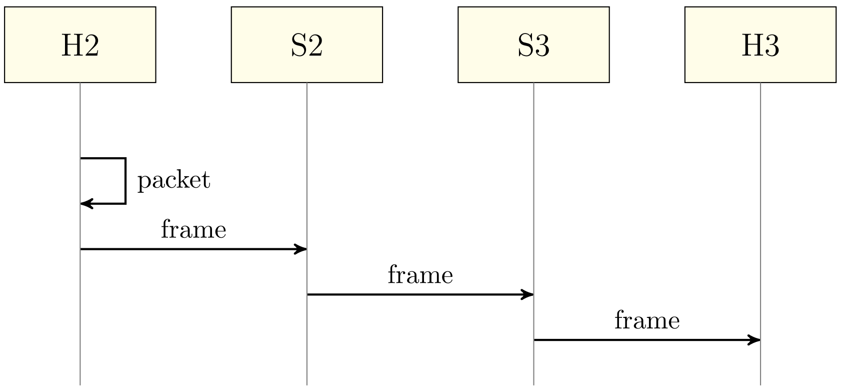

Figure B.2: Flow Diagram for the Scenario S.DG.01

Discussion

There are several different things that can be proven about the scenario: that H2 sent a frame, the payload of the frame, the way the frame travelled through the network, the timestamp when H2 sent the frame, that H3 received a frame and the timestamp when H3 received the frame.

Without actually capturing the frames directly in front of host H2 on the link L1, it is not possible to prove that H2 sent a specific frame. However, it is possible to show that H2 sent frames within a certain timeframe. When the Filter Database (I.DB.05) is known with the timestamps when entries were created and remove, it is possible to show when a certain MAC address (I.DE.03) sent traffic for the first time and on which switch port. In addition, it is possible to determine the last time when a certain MAC address sent traffic by subtracting the Filter Database ageing time (I.DB.01) from the event when the MAC address was remove from the Filter Database.

The path the frame took through the network is retraceable by the state of the spanning tree at the time the frame was sent or by the Filter Database (I.DB.05) of all network switches. For a previously unknown destination MAC address (not in any Filter Database), the spanning tree at the time the frame was sent is necessary to retrace how the frame was flooded through the network. I.DB.05 is already part of the set of significant information and the spanning tree can be retraced with the information in the set of significant information.

The exact timestamp when a frame was received by H3 and that it actually was received by H3 can only be retraced by a packet sniffer. However, the Frame Payload (I.DE.02) is not added to the set of significant information because it is likely that the frame also triggered other events on higher layers on H3.

Required Information I.DE.03, I.DE.09, I.DE.10, I.DB.01, I.DB.02, I.DB.03, I.DB.05

Scenario Dependency S.DG.02

B.3.2 Establishing Spanning Tree (S.DG.02)

Description

The switches S1, S2 and S3 establish a spanning tree. All bridges start with announcing themselves as root bridge. S1 and S3 stop as soon as they receive a BPDU from S2 with the lower Bridge ID. They set their interfaces towards S2 in the state *Root Port*. Then they start announcing S2 as Root Bridge on the other interfaces. S1 sets the interface to S3 to *Alternate Port* when it receives an announcement from S3 with the same cost but the lower Bridge ID.

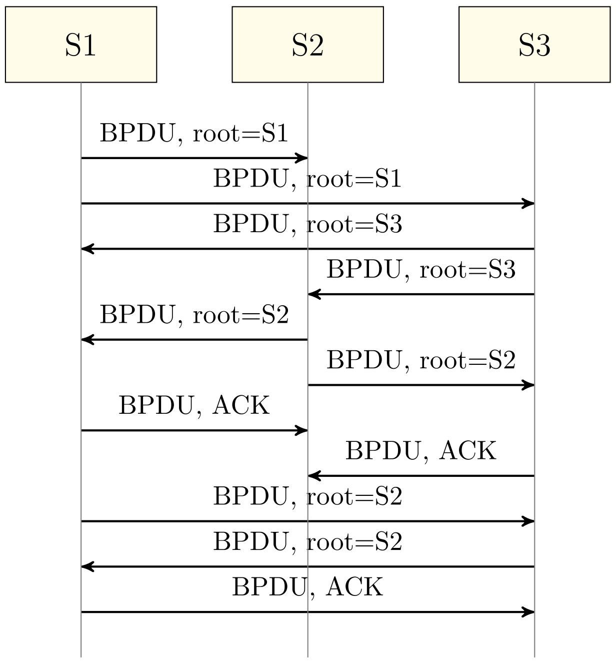

Figure B.3: Flow Diagram for the Scenario S.DG.02

Discussion

The spanning tree can be retraced with the information I.DB.02 - I.DB.09 and I.DB.15.

Required Information I.DB.02, I.DB.03, I.DB.04, I.DB.05, I.DB.06, I.DB.07, I.DB.08, I.DB.09, I.DB.15

Scenario Dependency -

B.3.3 Send Frame over Trunk (S.DG.03)

Description

H2 sends a frame to H3. The link L3 is configured as trunk. S2 adds the tag VLAN 10 to the frame before sending it to S3. S3 removes the tag before sending it to H3.

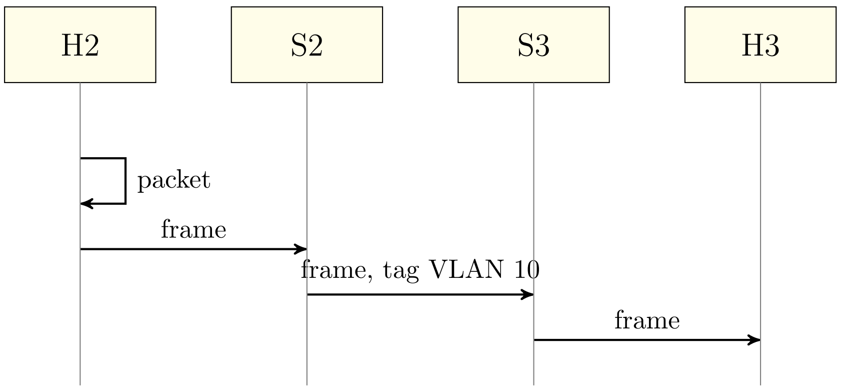

Figure B.4: Flow Diagram for the Scenario S.DG.03

Discussion

The path (spanning tree) and the VLAN tagging can be retrace with the interface role (I.DV.01) and the interface VLAN ID (I.DV.02).

Required Information I.DV.01, I.DV.02, I.DV.03, I.DV.04

Scenario Dependency S.DG.01

B.3.4 Device is Removed or Destroyed (S.DM.01)

Description

S1 is removed or destroyed during operation.

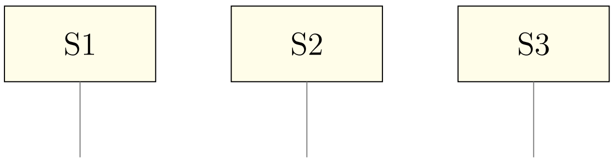

Figure B.5: Flow Diagram for the Scenario S.DM.01

Discussion

Assuming the interfaces connected to that device were up, the disappearance of the device leads to state changes on neighbour devices connected to the disappearing device. With the identified ports and the Physical Layout (I.DE.08). The removed device can be determined.

Required Information I.DE.06, I.DE.08

Scenario Dependency -

B.3.5 Link Unplugged (S.DM.02)

Description

The link L2 on is unplugged on S1.#0.

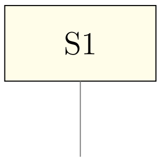

Figure B.6: Flow Diagram for the Scenario S.DM.02

Discussion

If a network link is unplugged the Link Operation Status (I.DE.06) changes from up to down, assuming the link was up before unplugging the link. With the Physical Layout (I.DE.08) and the affected ports it is possible locate the attack. It is not possible to determine on which side a link was unplugged or the cable was separated between the two ports. This has do be done on location by following the actual cable.

Required Information I.DE.06, I.DE.08

Scenario Dependency -

B.3.6 MAC Flooding (S.DM.03)

Description

M2 sends frames with randomly generated source MAC addresses. The switch S2 learns the MAC addresses and adds them to the Filter Database (I.DB.05) with the port the are received on. When the switch reaches the maximum memory allocated to the Filter Database, it starts overwriting the oldest entries. Typically the oldest entries are the correct ones (belonging to valid hosts). When the switch receives frames with a destination MAC address that was removed from the Filter Database because of the new entries, the switch forwards the frame to all interfaces but the one it was received on.

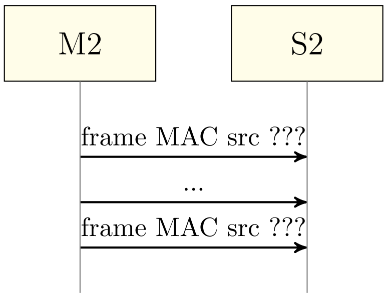

Figure B.7: Flow Diagram for the Scenario S.DM.03

Discussion

An upper limit needs to be defined to distinguish between a busy port and a malicious attack. It might be considered malicious if a new MAC address is learned on a port without the Link Operation Status (I.DE.06) changing to down first. Some commercially available switches support such features that limit the number of different MAC addresses per port (e.g. Cisco Port Security [101]). On the other hand, simply tracking new entries on the Filter Database (I.DB.05) also allows retracing the flooding attack.

Another way to detect a MAC flooding attack is to trigger a message when a switch needs to remove elements from the Filter Database because its memory is exhausted. However, this does not allow retracing on which port the malicious frames were received.

Required Information I.DB.05, I.DE.06

Scenario Dependency S.DG.01

B.3.7 MAC Spoofing (S.DM.04)

Description

M2 sends frames with the MAC source address of H2. S1 receives the frames and updates the Filter Database accordingly. The Filter Database of S2 states afterwards that the MAC address of H2 is behind interface S2.#2 instead of S2.#0. Any time H2 sends a frame the Filter Database changes back.

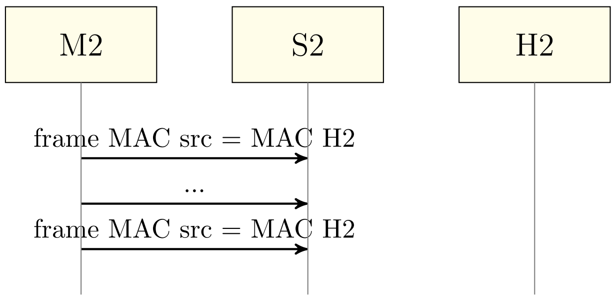

Figure B.8: Flow Diagram for the Scenario S.DM.04

Discussion

MAC spoofing can be retraced with the content of the Filter Database (I.DB.05). The attacks presents itself through a MAC address that keeps switching between two interfaces.

It is difficult to determine which port the attacker is connected to the network. To do so the Link Operation Status is important (I.DE.06). From the viewpoint of S2, if an address that was learned on an access port changes to another interface before the Link Operational Status goes down, the attack is in progress and the attacker is connected on the port where the MAC address was learned later or in this direction if the new port is connected to another switch.

Required Information I.DB.05, I.DE.06

Scenario Dependency S.DG.01

B.3.8 RSTP Root Claim (S.DM.05)

Description

M2 announces BPDUs with a better Bride Priority (a value of 0) than S2. M2 is elected as Root Bridge and S2 loses the role. The spanning tree in the network changes accordingly.

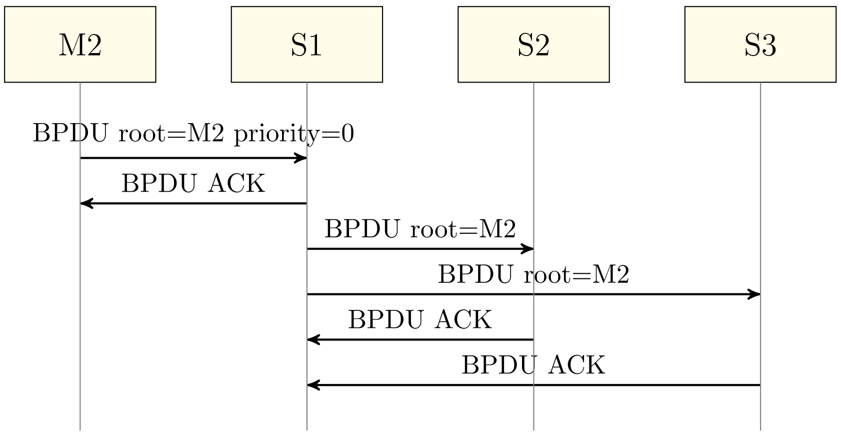

Figure B.9: Flow Diagram for the Scenario S.DM.05

Discussion

When an attacker claims the root role in an RSTP environment it leads to changes in Port Roles (I.DB.13) of all the bridges. At least the port where the attacker is connected to the network changes from Designated Port to Root Port. That is the case, when the attacker is directly connected to the Root Bridge. The changing port role allows identifying the port where the attacker is connected to the network. Another indicator of the attack is the new Root Bridge (I.DB.14) all bridges have.

As the bridge address (I.DB.02) and also the bridge identifier (I.DB.03) are based on the MAC address of the lowest interface, it might help to determine the MAC address of the node used by the attacker. Of course, the attacker might spoof the MAC address to hide tracks.

Required Information I.DB.13, I.DB.14

Scenario Dependency S.DG.02

B.3.9 RSTP Eternal Root Election (S.DM.06)

Description

M2 sends BPDUs with a decrementing Bridge ID, starting with lowest active Bridge ID in the network. The switches in the network stay in the state of electing a new Root Bridge and temporary loops might be the consequence.

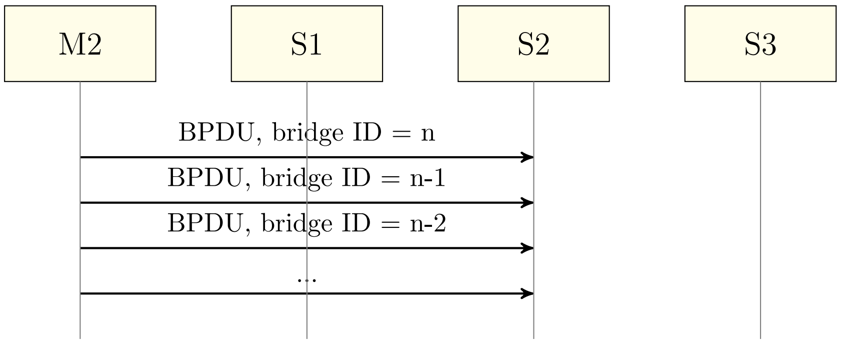

Figure B.10: Flow Diagram for the Scenario S.DM.06

Discussion

An RSTP eternal root election attack can be detected similar to the scenario S.DM.05, the RSTP root claim. The Port State (I.DB.13) towards the attacker also changes from Designated Port to Root Port, which allows locating the port where the attacker is connected to the network. Similar, the Bridge Address (I.DB.02) and the Bridge Identifier (I.DB.03) allows potentially identifying the MAC address of the system the attacker uses.

A further indication of the attack is a changed Root Path Cost (I.DB.11) and new Root Bridge ID (I.DB.14) on all switches in the network. Based on the Port Path Cost (I.DB.08) and several bridges to start from, it might be possible to pin-point the bridge where the attacker is connected to the network.

Required Information I.DB.11, I.DB.13, I.DB.14

Scenario Dependency S.DG.02

B.3.10 RSTP Topology Change Flooding (S.DM.07)

Description

M2 floods the network with TCN. This persistent TCN messages attack can cause loops and keeps the network in an unstable state.

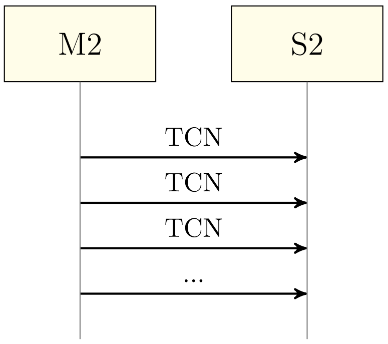

Figure B.11: Flow Diagram for the Scenario S.DM.07

Discussion

An RSTP TCN flooding attack can be detected on any bridge in the network by monitoring the number of TCNs received (I.DB.12) during a certain timeframe. To retrace the source of the attack, it is necessary to monitor all bridges. With the help of the spanning tree information, it is possible to determine the node furthest away from the root bridge with an unusual high number of TCN messages and, therefore, the source of the attack.

Another way to retrace the attack is to create a log message whenever a TCN message on an edge port is received. As edge ports are not connected to other bridges, there should not be any TCN messages received. Some commercial switches support similar features (e.g. Cisco BPDU Filter/Guard [101]).

Required Information I.DB.12

Scenario Dependency S.DG.02

B.3.11 VLAN Hopping (S.DM.08)

Description

M2 tags a frame with two IEEE 802.1Q tags. The outer tag is equal to the access VLAN of M2. The inner tag is equal to the access VLAN of H3. When the switch receives the frame, it removes the outer tag and forwards it on the trunk link. Because the native VLAN on the trunk link is equal to the access VLAN of M2, the frame is not tagged with the access VLAN of M2 to send it across the trunk. S3 receives then the frame and interprets, not knowing that it was sent in the native VLAN, the inner tag and assigns the frame to the corresponding VLAN. The frame is forwarded to H3.

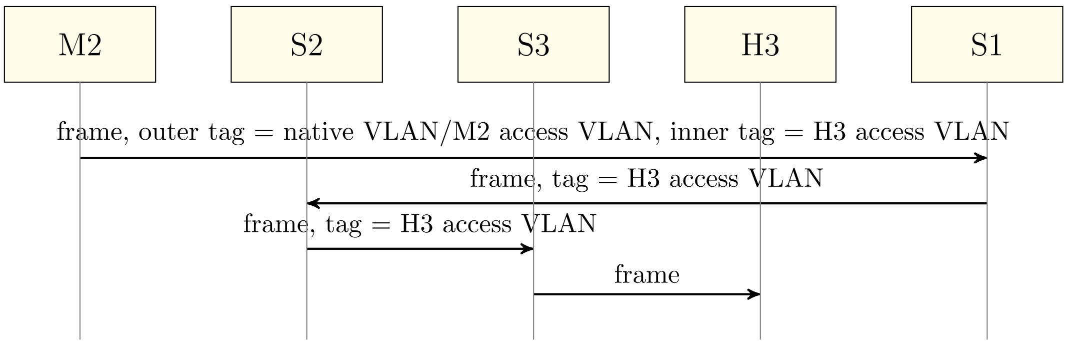
M2
S2
S3
H3
S1
frame, outer tag = native VLAN/M2 access VLAN, inner tag = H3 access VLAN
frame, tag = H3 access VLAN
frame, tag = H3 access VLAN
frame

Figure B.12: Flow Diagram for the Scenario S.DM.08

Discussion

VLAN hopping with double tagged frames is difficult to retrace. Basically, if a frame is received on an access interface (Interface Role - I.DV.01) with one or multiple VLAN tags in the Frame Header (I.DE.01) the frame should be dropped and a log message should be created. Such a log message allows retracing the attack. However, the switches used in this thesis do now allow creating such a log message. Capturing the network traffic allows detecting the attack but not retracing it to the access port. Therefore, additional software needs to be developed for the switches to be able to retrace this scenario.

Required Information I.DV.01, I.DE.01

Scenario Dependency S.DG.03

B.3.12 Occupying a Link (S.DM.09)

Description

M2 runs a DoS attack by sending so much traffic to H3 that the links L11, L2, L3 and L5 are occupied in one direction.

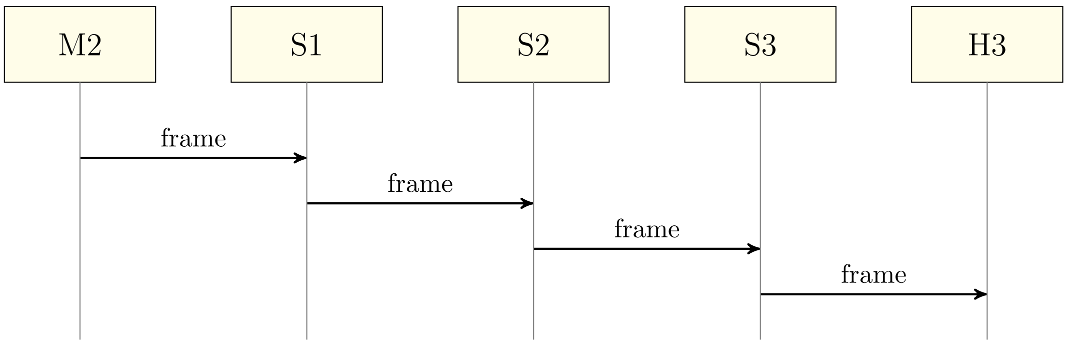

Figure B.13: Flow Diagram for the Scenario S.DM.09

Discussion

With the physical layout of the network (I.DE.08), the traffic counter (I.DE.11) and the spanning tree is is possible to locate the attacker. The interfaces with a large amount of incoming traffic point towards the attacker.

To show the plausibility of the attack, the Link Speed (I.DE.07) is relevant. For example, if an attacker connects only with a 10 Mbps to the network is is not plausible that the attacker occupies a link with 1 Gbps capacity.

Required Information I.DE.07, I.DE.08, I.DE.11

Scenario Dependency S.DG.01

B.3.13 Administrative Access (S.DM.10)

Description

M2 gets administrative access on S1. In this position M2 is able to run a variety of attacks. The attacker could delete the configuration and reboot the switch (S.DM.01), shut interfaces (SDM.02), manipulate the Filter Database or the spanning tree (SDM.05, SDM.06), redirect traffic and change the VLAN configuration.

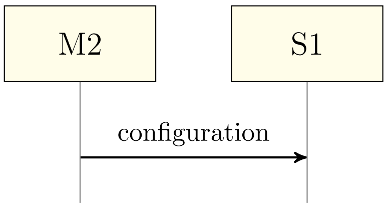

Figure B.14: Flow Diagram for the Scenario S.DM.10

Discussion

Manipulations of the Filter Database are difficult to detect. If static entries in the Filter Database are not common in the network, a configuration management system that traces changes is able to pick them up. However, configuration management is out of the scope of this research. Logging successful and failed login attempts is a way to generally detect and retrace malicious administrative access to network devices. As this is part of the user management, it is also out of the scope of this research.

The information needed to retrace this scenarios is outside the scope of this research.

Required Information -

Scenario Dependency S.DM.01, S.DM.02, S.DM.05, S.DM.06

B.4 Network Layer Scenarios

B.4.1 Enabling and Configuring IP (S.NG.01)

Description

H2 activates its IP layer and generates a link-local address. It joins the solicited-node multicast address and creates a DAD message to ensure the local address is not already used by another host. Then, H2 sends an RS message to receive RA messages from all router in the same data-link layer. Once received, the node generates its global IP address, joins the appropriate multicast groups and creates another DAD to make sure that the address is not already used by another host.

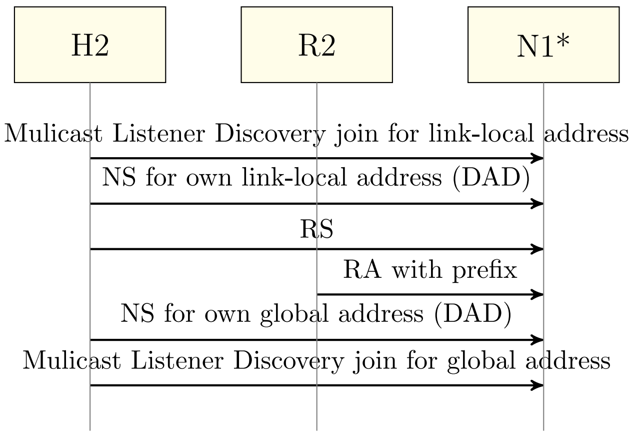
H2
R2
N1*
Mulicast Listener Discovery join for link-local address
NS for own link-local address (DAD)
RS
RA with prefix
NS for own global address (DAD)
Mulicast Listener Discovery join for global address

Figure B.15: Flow Diagram for the Scenario S.NG.01

Discussion

The IP address of H2 can be determined with the Interface Address (I.NI.08). To retrace the whole procedure the actual protocol messages need to be captured: NS message (I.NC.03), NA message (I.NC.04), RA message (I.NC.01), RS message (I.NC.02).

The access port of an IP address can be determined with the Neighbour Cache (I.NC.06) of R2 and the Filter Database (I.DB.05) of S1 - S3. The Neighbour Cache of the first-hop router contains the data-link layer address corresponding to IP addresses of H2. If the Neighbour Cache does not contain an entry for H2, an NS/NA message exchange can be triggered by sending any IP traffic to H2 (e.g. ICMP Echo Request). This populates the Neighbour Cache of R2. The data-link layer address can also be found in the Filter Database of the switches S1 - S3 in the same network. The Filter Database contains pairs of data-link layer addresses and ports. Tracing a data-link layer address through the Filter Database of the switches allows reaching the access port of the node with the specific data-link layer address. For example, the Filter Database of S3, the closest switch to R2, contains an entry that the data-link layer address of H2 is behind port #2, which is S2. The Filter Database of S2 contains an entry that the data-link layer address of H2 is behind port #0. Because no other switch is connected to this port, the access port of H2 is S2.#0.

Other options to determine the IP address of a node are discussed in Section 6.2.1.

Required Information I.NI.08, I.NC.01, I.NC.02, I.NC.03, I.NC.04, I.DB.05, I.DE.08

Scenario Dependency SDG.01

B.4.2 Sending a Packet to an On-Link Host (S.NG.02)

Description

H2 sends an IP packet to H3. H2 consults its Destination Cache (I.NC.07) to determine how to reach H3. Because it is the first time H2 sends a packet to H3, the Destination Cache of H2 does not have an entry for H3. Therefore, H2 consults its Routing Table (I.NI.09) to determine how to reach H3. It shows that H3 is on the same data-link layer and that it is reachable through interface #0. H2 consults then the Neighbour Cache (I.NC.06) to determine the data-link layer address of H3. Again, because it is the first time H2 sends a packet to H3, the Neighbour Cache does not have an entry for H3. Consequently, H2 crafts and sends an NS message (I.NC.03) to determine the data-link layer address of H3. H3 receives the NS message and replies with an appropriate NA message (I.NC.04). H2 receives the NA message, updates its neighbour and Destination Cache, and sends the packet to H3. Sending the packet on the data-link layer is described in scenario S.DG.01.

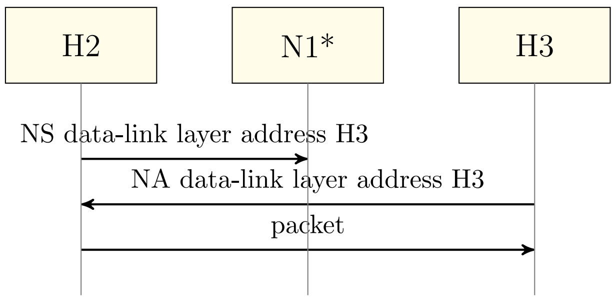

Figure B.16: Flow Diagram for the Scenario S.NG.02

Discussion

Without actually capturing the packet in front of H2 and H3, it is not possible to prove that a specific packet was sent from H2 to H3 at a specific time. However, it is possible to show that there was communication between H2 and H3 based the changes in the Neighbour and Destination Caches. With the knowledge of the cache timeouts, it is also possible to estimate the time of the last communication between the two systems.

The scenario is not fully retraceable with the current set of significant information. However, the Packet Payload (I.NI.14) is not added to the set of significant information because it is likely that the packet triggers events when it is received and passed to the next higher layer on H3. This would render the information redundant. The access ports can be identified based on scenario S.NG.01.

Required Information I.NI.09, I.NC.03, I.NC.04, I.NC.06, I.NC.07

Scenario Dependency SDG.01, S.NG.01

B.4.3 Sending a Packet to an Off-Link Host (S.NG.03)

Description

H2 sends an IP packet to H1. H2 consults its Destination Cache (I.NC.07) to determine how to reach H1. Because it is the first time H2 sends a packet to H1, the Destination Cache of H2 does not have an entry for H1. Therefore, H2 consults its Routing Table (I.NI.09) to determine how to reach H1. It shows no specific route to reach H1, so H2 uses the default route. H2 does not need to determine the data-link layer address of the default router R2 because it is already in the Neighbour Cache (I.NC.06). It is added and updated every time H2 receives an RA message with the source data-link layer option. H2 sends the packet to R2 (see scenario S.DG.01). R2 receives the packet, decrements the hop limit, looks up the destination IP address in the Destination Cache (I.NC.07) or, if not found, in the Routing Table (I.NI.09) and forwards the packet to R1. R1 repeats the steps of R2 but determines that the destination address is on a directly connected network. Therefore, R2 looks the data-link layer address of H1 up in the Neighbour Cache (I.NC.06) or crafts and sends an NS message if the address is not found. R1 sends then the packet to H1.

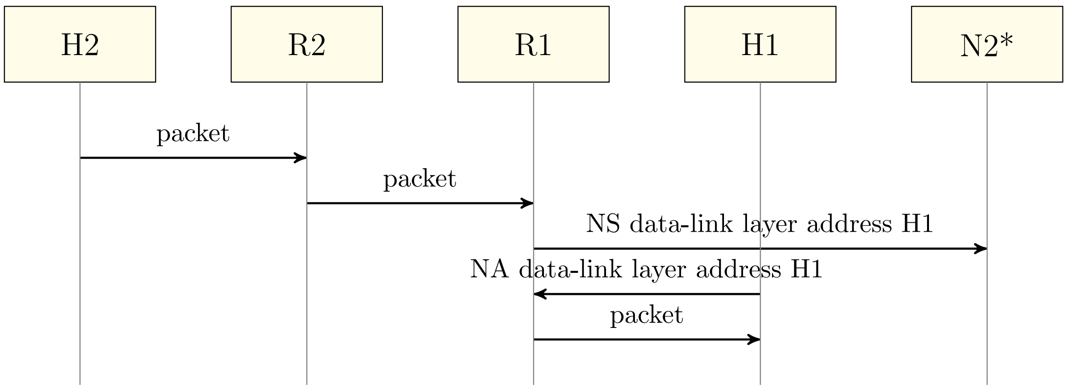

Figure B.17: Flow Diagram for the Scenario S.NG.03

Discussion

The scenario is identical to S.NG.02 but adds the element of packet routing. Therefore, the Routing Table (I.NI.09) and/or the Destination Cache (I.NC.07) of R1 and R2 are crucial to be able to retrace the path of the packet through the network. Further, the Routing Table and the Destination Cache can be verified by the physical network layout (I.DE.08).

As in the previous scenario, it is not be possible to retrace the exact content of a packet or the exact timestamp when it was sent and received. However, the Packet Payload (I.NI.14) is not added to the set of significant information because it is likely that the packet triggers events when it is received and passed to the next higher layer on H3. This would render the information redundant. The access ports can be identified based on scenario S.NG.01.

Required Information I.NI.09, I.NC.03, I.NC.04, I.NC.06, I.NC.07, I.DE.08

Scenario Dependency SDG.01, S.NG.01

B.4.4 Sending Packet to a Non-Existing Host (S.NG.04)

Description

H2 sends an IP packet to non-existing host Hx. The address of Hx is in the same subnet as H1. The scenario is almost identical to the scenario S.NG.03. The only difference is that the destination address is not used by any hosts. Once R1 receives the packet from R2, it sends an NS message (I.NC.03) to determine the data-link layer address of the non-existing network layer address. It waits a second for a reply and repeats the step twice before sending an ICMP Destination Unreachable message (I.NC.28) code 3 (Address Unreachable) to H2. The source network layer address is the one on the interface facing the access network of the non-existing host. The access port of the sender can be identified with its IP address and the process described in scenario S.NG.01.

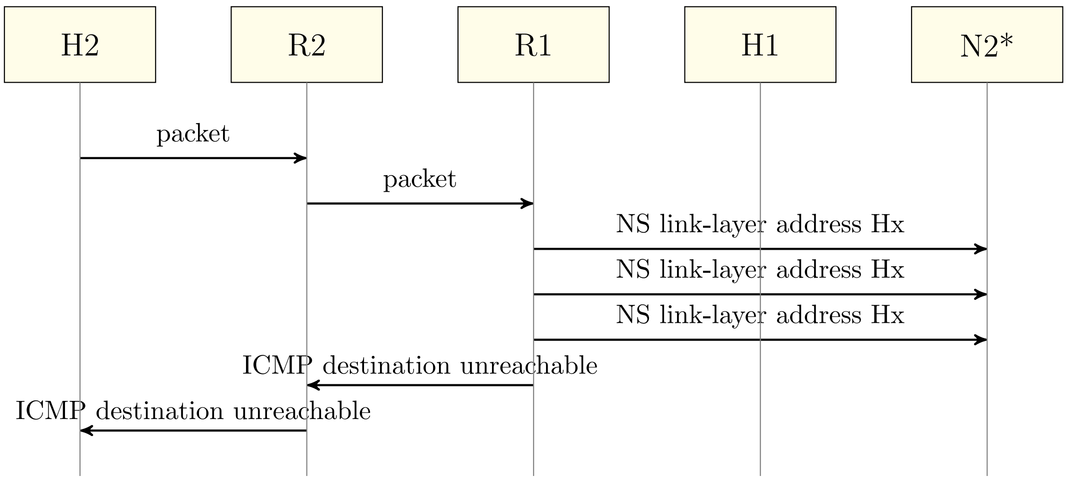

Figure B.18: Flow Diagram for the Scenario S.NG.04

Discussion

The scenario is identical to S.NG.03 but the behaviour of R1 is different. After scenario S.NG.03, the router has entries in the Destination Cache (I.NC.07) and in the Neighbour Cache (I.NC.06) for the address of H1. In this scenario, R1 has only an entry in the Destination Cache (I.NC.07) but not in the Neighbour Cache (I.NC.06). That is because R2 does not succeed with resolving the data-link layer address of Hx. Furthermore, R1 sends an ICMP Destination Unreachable message (I.NC.28) back to H2.

The current set of significant information allows retracing the scenario based on the Destination Cache (I.NC.07), the Neighbour Cache (I.NC.06), and the prefix list (I.NC.08) or the Routing Table (I.NI.09). The conclusion this scenario happened is based on the fact that a new entry in the Destination Cache appeared for an address the is in a directly connected network (determined by the prefix list or Routing Table) but a corresponding entry in the neighbour table was not created. Collecting the NS (I.NC.03) and NA messages (I.NC.04) and the ICMP Destination Unreachable (I.NC.28) messages would simplify the task of retracing this scenario.

Required Information I.NC.03, I.NC.04, I.NC.06, I.NC.07, I.NC.08, I.NC.28

Scenario Dependency S.NG.01, S.NG.03

B.4.5 Sending a Too Large Packet (S.NG.05)

Description

H2 sends an IP packet to H1 that is bigger than the link MTU of L7. R2 drops the packet and sends an ICMP Packet Too Big message to H2.

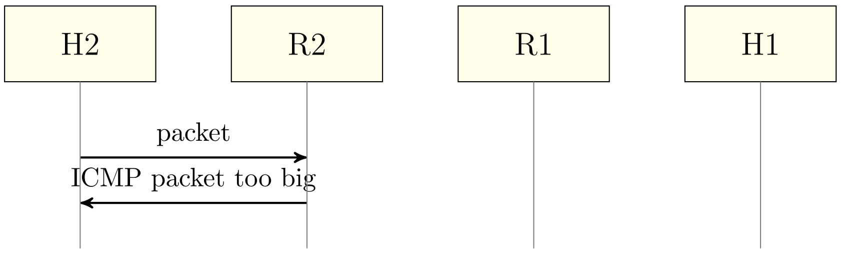

Figure B.19: Flow Diagram for the Scenario S.NG.05

Discussion

The scenario can be retraced because H2 updates the Destination Cache (I.NC.07) when it receives the ICMP Packet Too Big message (I.NC.29) from R2. The scenario can be further verified with the physical network layout (I.DE.08), the Interface Link MTU (I.NC.22) and the Routing Table (I.NI.09) information. The access port of the sender can be identified with its IP address and the process described in scenario S.NG.01.

The actual ICMP Packet Too Big message (I.NC.07) would simplify the retracing but it is not needed to retrace this scenario.

Required Information I.NC.07, I.NC.22, I.NC.29, I.DE.08, I.NI.09

Scenario Dependency S.NG.01, S.NG.03

B.4.6 Sending a Packet with a Small Hop Limit (S.NG.06)

Description

H2 sends a packet to H1 with a hop limit of two, R2 decrements the limit to one and R1 to zero. R1 drops the packet and sends an ICMP Hop Limit Exceeded message to H2.

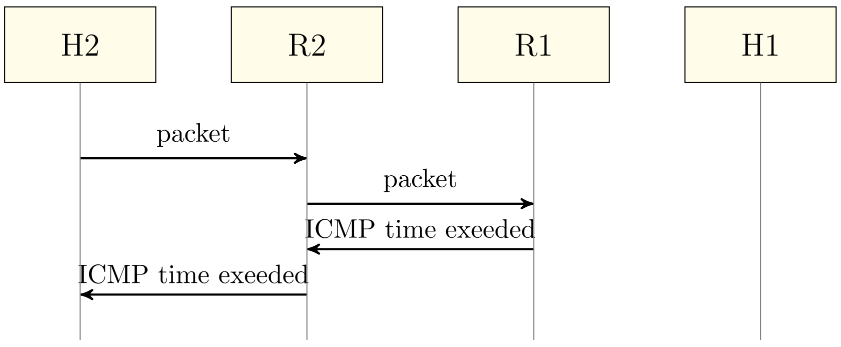

Figure B.20: Flow Diagram for the Scenario S.NG.06

Discussion

The Routing Table (I.NI.09) and/or the Destination Cache (I.NC.07) allow retracing the path of a packet with a very low hop limit. However, this does not allow retracing the exact timestamp when the packet was sent and not the content of the packet. The ICMP Hop Limit Exceeded message (I.NC.30) does allows retracing which router dropped the packet, when it was dropped and part of packet that was dropped (as much as fit into a single MTU). The senders IP address can be extracted from the ICMP error message and the access port of the sender with the procedure described in scenario S.NG.01.

The scenario is retraceable with parts of the IP header that triggered the error. With the part of the IP header, the original Packet Header (I.NI.01) can be found and the timestamp when it was sent can be recovered.

Required Information I.NI.09, I.NC.07, I.NC.30

Scenario Dependency S.NM.01, S.NM.03

B.4.7 Sending a Packet with Unknown Parameter (S.NG.07)

Description

H2 sends a packet to H1 with an unknown parameter (e.g. set Next Header to 254). H1 replies with an ICMP parameter problem.

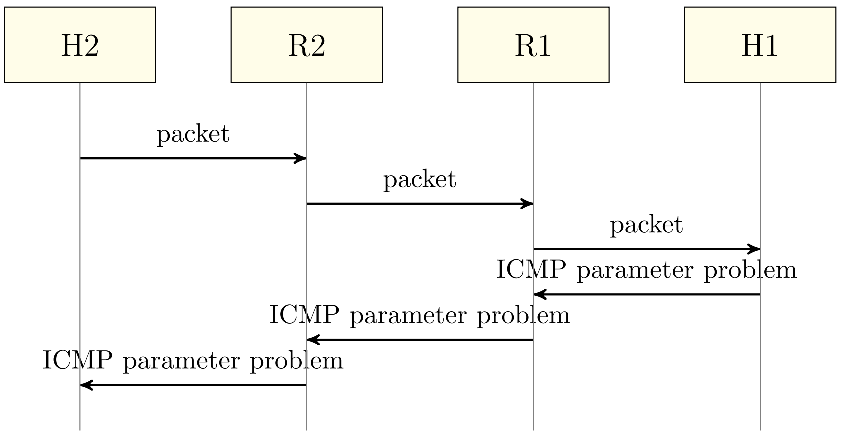

Figure B.21: Flow Diagram for the Scenario S.NG.07

Discussion

This scenario is almost identical to scenario S.NG.06 but instead of a small hop limit, an unknown parameter in the IP packet is used. The retracing of this scenario is also similar to the scenario S.NG.06. The only difference is that the ICMP Parameter Problem message (I.NC.31) is crucial this time. The senders IP address can be extracted from the ICMP error message and the access port of the sender with the procedure described in scenario S.NG.01.

The scenario is retraceable with parts of the IP header that triggered the error. With the part of the IP header, the original Packet Header (I.NI.01) can be found and the timestamp when it was sent can be recovered.

Required Information I.NC.31

Scenario Dependency S.NM.01, S.NM.03, S.NM.06

B.4.8 Sending an ICMP Echo Request (S.NG.08)

Description

H2 sends an ICMP Echo Request message to H1. H1 replies with an ICMP Echo Reply message.

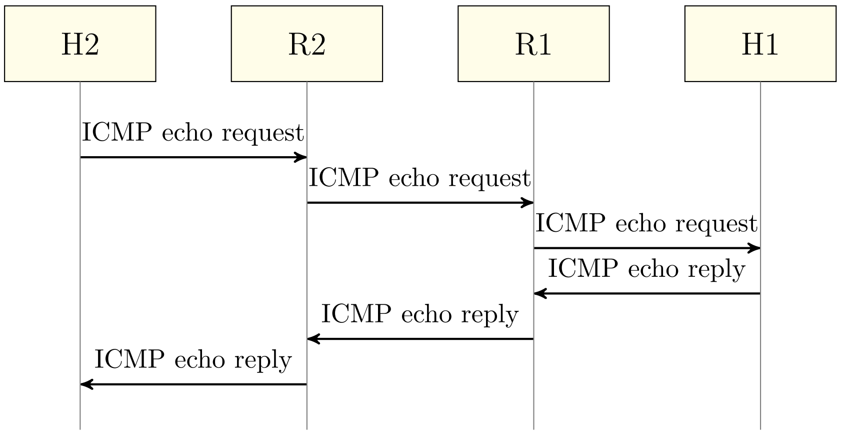

Figure B.22: Flow Diagram for the Scenario S.NG.08

Discussion

The actual messages ICMP Echo Request I.NC.33 and ICMP Echo Reply I.NC.34 messages allow determining the exact content of the messages, the time they were sent, and the sender and receiver. The Routing Table (I.NI.09) and/or the Destination Cache (I.NC.07) allow retracing the path the messages took through the network.

Required Information I.NI.09, I.NC.07, I.NC.33, I.NC.34

Scenario Dependency S.NM.01, S.NM.03

B.4.9 Sending Traffic through a 6in4 Tunnel (S.NG.09)

Description

R1 and R2 are configured as tunnel endpoints for a 6in4 tunnel. The tunnel goes through R3 as the link L7 is disabled. H2 sends a packet to H1.

When R2 receives a packet addressed to H1 it does the normal routing lookup. It determines the tunnel interface as the outgoing interface. R2 encapsulates the IPv6 packet in an IPv4 packet addressed to R1 and injects it into the IPv4 routing process. R2 sends the encapsulated packet to R3, which forwards it to R1. R1 inspects the packet because it is addressed to itself. R1 detects the next header field IPv6 (41) that tells the router to decapsulate the packet and inject it back into the IPv6 packet routing. R1, finally, forwards the packet to H1.

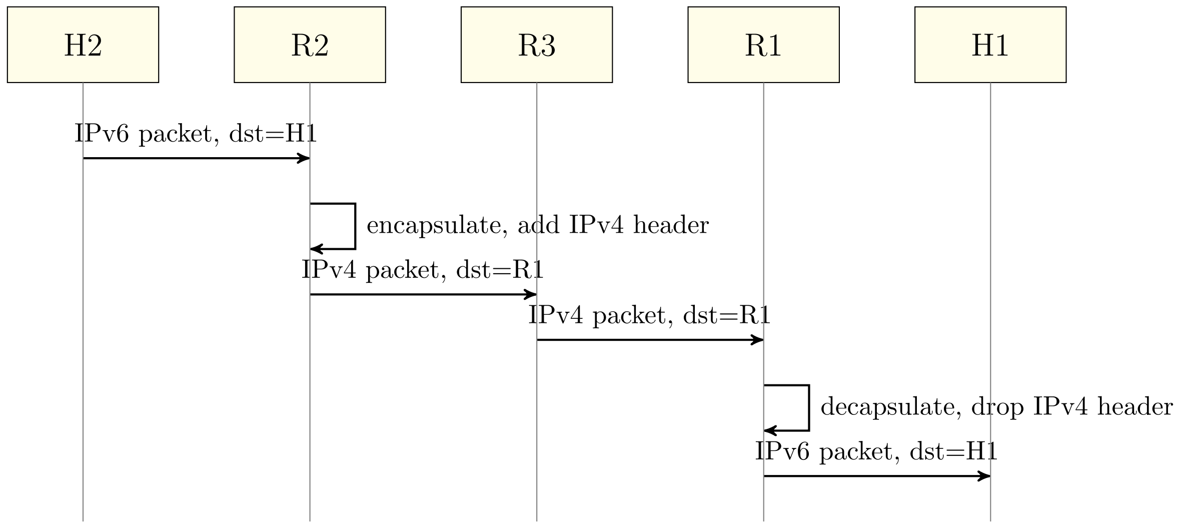

Figure B.23: Flow Diagram for the Scenario S.NG.09

Discussion

The existence of the 6in4 tunnel and that the tunnel is used for the traffic sent from H2 to H1 can be concluded from the Routing Table (I.NI.09) or the Destination Cache (I.NC.07) on R2 and the 6in4 tunnel configuration (I.N6.01 - I.N6.08). The access port of the sender and the receiver can be identified with their IP addresses and the process described in scenario S.NG.01.

This allows only retracing the path of the packet and that the tunnel was used. It does not allow retracing a specific packet (content and time). Only capturing the packet payload (I.NI.14) allows that.

Required Information I.NI.09, I.NC.07, I.N6.01, I.N6.02, I.N6.03, I.N6.04, I.N6.05, I.N6.06, I.N6.07, I.N6.08

Scenario Dependency S.NG.01, S.NG.03

B.4.10 Sending Traffic through a GRE Tunnel (S.NG.10)

Description

R1 and R2 are configured as tunnel endpoints for a GRE tunnel. The tunnel goes through R3 as the link L7 is disabled.

When R2 receives a packet addressed to H1 it does the normal routing lookup. It determines the tunnel interface as the outgoing interface. R2 encapsulates the IPv6 packet in a GRE/IPv4 packet addressed to R1 and injects it into the IPv4 routing process. R2 sends the encapsulated packet to R3, which forwards it to R1. R1 inspects the packet because it is address to itself. R1 detects the next header field GRE (47) that tells the router to decapsulate the packet and inject it back into the IPv6 packet routing. R1, finally, forwards the packet to H1.

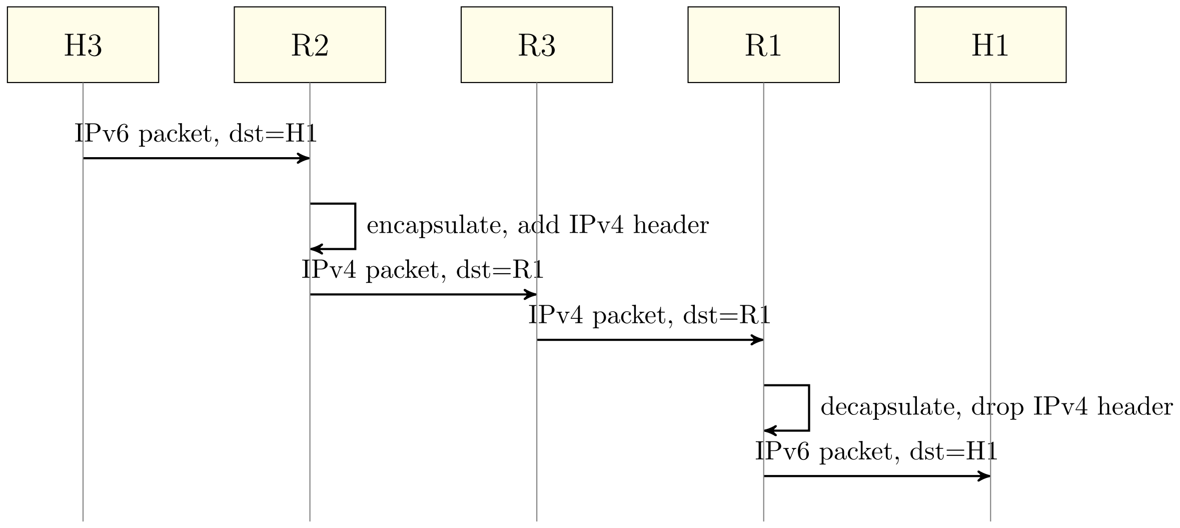

Figure B.24: Flow Diagram for the Scenario S.NG.10

Discussion

This scenario is almost identical to S.NG.09. The only difference is the protocol used to tunnel the traffic, which is in this scenario GRE. GRE leads to a slightly bigger protocol overhead (24 bytes at least) than 6in4 (20 bytes) and, therefore, to a smaller tunnel MTU. The existence of the tunnel and that the tunnel is used for the traffic sent from H3 to H1 can be concluded from the Routing Table (I.NI.09) or the Destination Cache (I.NC.07) on R2 and the GRE tunnel configuration (I.NG.01, I.NG.05 - I.NG.09). The access port of the sender and the receiver can be identified with their IP addresses and the process described in scenario S.NG.01.

This allows retracing the path of the packet and that the tunnel was used. However, it does not allow retracing a specific packet at a specific time. Only capturing the packet payload (I.NI.14) allows that.

Required Information I.NI.09, I.NC.07, I.NG.01, I.NG.05, I.NG.06, I.NG.07, I.NG.08, I.NG.09

Scenario Dependency S.NG.01, S.NG.03

B.4.11 Establishing Routing with RIPng (S.NG.11)

Description

The routers R1, R2 and R3 establish packet routing for the network with RIPng. Each router starts announcing the directly attached networks with global addresses to the multicast address ff02::9 on all interfaces. The routers receive the updates from their neighbours and add the new information to the routing table. They announce the new routing table in the next update.

They do not announce networks they have learned on a network interface back on the same interface (split-horizon) or announce them with the unreachable metric (poison-reverse).

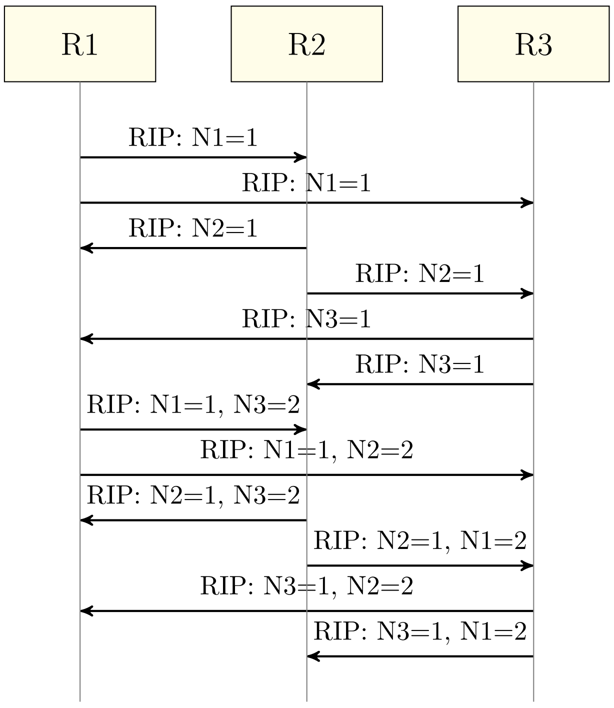

Figure B.25: Flow Diagram for the Scenario S.NG.11

RIP updates are also sent towards the networks N1-3 but are not shows in the flow diagram for simplicity.

Discussion

To retrace the scenario several pieces of information are needed: the routers running RIP (I.NR.02), the directly connected networks (I.NR.03), the Valid Neighbour List (I.NR.11) and the Filter List (I.NR.12). This allows retracing the router sending RIP updates and the routers receives and propagates them. Directly connect routers can be used to identify the IP address of the neighbour routers and the Neighbour Cache (I.NC.06) the corresponding data-link layer address. Both pieces of information can be verified with the interface addresses on the network layer (I.NI.08) and on the data-link layer (I.DE.03). The Filter Database (I.DB.05) allows identifying the access port where the routers are connected to the network.

Required Information I.NR.02, I.NR.03, I.NR.11, I.NR.12, I.NC.06, I.NI.08, I.DE.03, I.DB.05

Scenario Dependency S.NG.03

B.4.12 Establishing Routing with OSPFv3 (S.NG.12)

Description

The routers R1, R2 and R3 establish packet routing for the network with OSPF version 3. The neighbours establish a relationship, exchange the DB description and synchronise their LS databases. Afterwards, each router calculates the routing table with Shortest Path First algorithm.

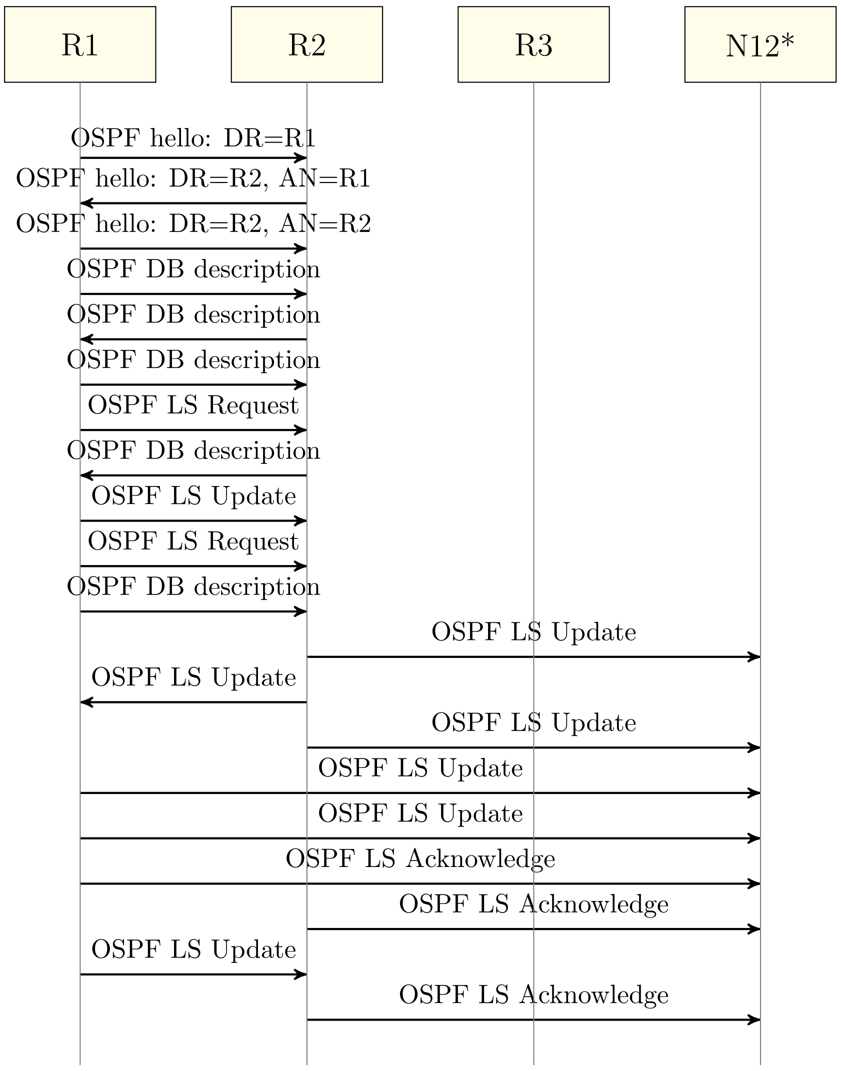

Figure B.26: Flow Diagram for the Scenario S.NG.12

The flow shows just the establishment of the neighbour relationship between R1 and R2. The reason for the messages being sent to N12 instead of to the neighbour is the DR/BDR role of R1 and R2.

Discussion

Retracing the neighbour relationships (I.NO.05) and the Link State Database (I.NO.06) of each router is possible. Furthermore, the result of the Shortest

Path First calculation, the Routing Table (I.NO.07, I.NI.08), is also available and allows showing the consistency of the different OSPF information (I.NO.01 - I.NO.13) with the resulting Routing Table (I.NI.08). This can be further verified with the Physical Layout (I.DE.08) of the network. The access ports of the routers can be identified with their IP addresses based on the neighbour table and the procedure described in the scenario S.NG.01.

Required Information I.NO.01, I.NO.02, I.NO.03, I.NO.04, I.NO.05, I.NO.06, I.NO.07, I.NO.08, I.NO.09, I.NO.10, I.NO.11, I.NO.12, I.NO.13, I.NI.08, I.DE.08

Scenario Dependency S.NG.01, S.NG.03

B.4.13 Sending a Packet with a Spoofed Source Address (S.NM.01)

Description

M2 sends an IP packet to H1. M2 spoofs the source address with the one of H2. For H1 it appears as the traffic originates from H2 and any reply traffic is sent to H2.

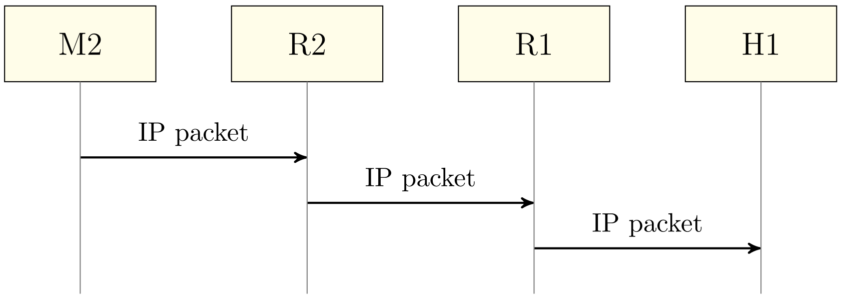

Figure B.27: Flow Diagram for the Scenario S.NM.01

Discussion

If M uses its own data-link layer source address (the other case is covered in scenario S.DM.04), the spoofing can be detected by a mismatch of the data-link layer and the network layer source address. Devices theoretically in a position to detect such a mismatch would be switches used to reach the default router (S2, S3) and the default router (R2). However, neither the switches nor the router has all the information necessary to do so readily available.

The switches could use the Filter Database (I.DB.05) combined with a mechanism that traces NS/NA messages or DAD messages (if SLAAC or DHCP is used) to map network layer addresses to the entries in the Filter Database to create valid triples of access port, source data-link layer address, and source network layer addresses. Some commercial switches support similar features (e.g. Cisco IPv6 ND Inspection [103]).

The other device able to detect a mismatch is R2. The router can compare the data-link layer address of the incoming packet (I.DE.01) with the data-link layer address stored in the Neighbour Cache (I.NC.06) for the network layer source address (I.NI.01) of the incoming packet. However, the Neighbour Cache might have no entry for be empty for the network layer address of the incoming packet (H2's address) and R2 need to send first an NS message to populate the Neighbour Cache. The additional delay might be undesirable and the attacker M might spoof the answer (see scenario S.NM.13 for the NS message spoofing scenario).

If M choses to use an IP address outside the network range of N2 as the source address, R2 could detect the malicious activity with a traffic filter on the incoming interface that logs traffic with source IP addresses outside the network range assigned to the interface. This feature is also known as RPF based filtering.

New software needs to be developed to detect the network layer address spoofing. See Section 9.4.6 for a discussion of such software.

Required Information I.DB.05, I.DE.01, I.NI.01, I.NC.03, I.NC.04, I.NC.06

Scenario Dependency S.NG.03

B.4.14 Circumventing Traffic Filtering with the Source Routing Header (S.NM.02)

Description

M2 circumvents the traffic filter with the source routing header (RFC 5095 [36] deprecates type 0 source routing headers). R2 filters traffic and allows only H2 sending traffic to H1. M2 circumvents the filter by sending an IP packet to H2 with a type 0 source routing header. H2 interprets the source routing header and forward the packet to H1. R2 does not filter the packet because it comes from H2.

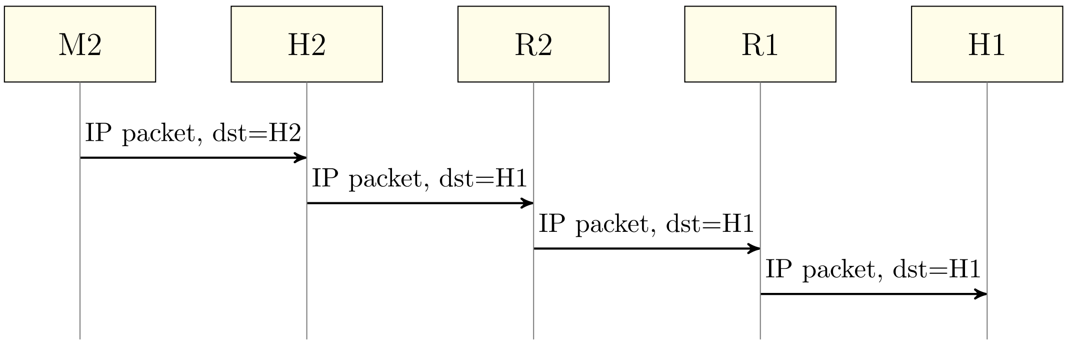

Figure B.28: Flow Diagram for the Scenario S.NM.02

Discussion

The Packet Routing Header (I.NI.03) is important to retrace the misuse of IP source routing. Further, the Protocol Parameters (I.NI.14) control if a node actually interprets the routing header (`accept_source_route`) and forwards packets (`forwarding`). The access port of the sender and the receiver can be identified with their IP addresses and the process described in scenario S.NG.01. In this scenario, the IP address of the sender is the source address in the Packet Header but the address of the receiver might be in the destination address of the Packet Header (I.NI.01) or the Packet Routing Header depending where the packet was captured on the network.

Devices able to retrace the scenario are such that are involved into the source routing, in that case H2, but also devices routing the packet. This is because H2 does not remove the source routing extension header but merely switches IP addresses from the extension header to the IP header and vice versa.

Required Information I.NI.01, I.NI.03, I.NI.14

Scenario Dependency S.NG.01, S.NG.03

B.4.15 Overlapping IP Fragments (S.NM.03)

Description

M2 sends overlapping IP packet fragments to H1. RFC 5722 [37] updates the handling of fragmented packets and explicitly forbids overlapping fragments. A host receiving such IP fragments should silently drop them.

If systems do not implement these recommendations, M2 could use overlapping IP packet fragments to circumvent traffic filtering on R2. For example, R2 does not allow establishing sessions sourcing in N2. Therefore, R2 filters TCP segments with the SYN flag set and the ACK flag not set (first packet in the TCP three-way handshake) on interface #0. On the other hand, R2 allows TCP segments with the SYN flag and the ACK flag set (second packet in the TCP three-way handshake). That way hosts in N2 can reply to TCP connection request but cannot initiate TCP connection requests.

M2 sends a first IP fragment with a Segment Header that has the SYN and the ACK flag set and a second fragment that has just the SYN flag set. The offset of the second packet is chosen that it overwrites the first Segment Header when H1 puts the two fragments together. If R2 only inspects the first fragment and uses the filter decision for all subsequent packets, M2 would be able to successfully establish a TCP session from N2.

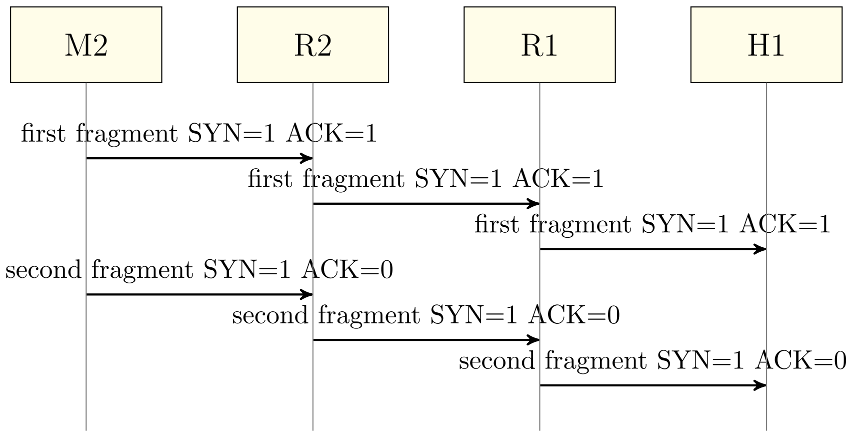
M2
R2
R1
H1
first fragment SYN=1 ACK=1
first fragment SYN=1 ACK=1
first fragment SYN=1 ACK=1
second fragment SYN=1 ACK=0
second fragment SYN=1 ACK=0
second fragment SYN=1 ACK=0

Figure B.29: Flow Diagram for the Scenario S.NM.03

Discussion

To retrace overlapping IP fragments the Packet Fragment Header (I.NI.04) is crucial. The only indication is in the offsets chosen in the two packet headers and how they are positioned to each other. The access port of the sender and the receiver can be identified with their IP addresses and the process described in scenario S.NG.01.

Devices in a good position to collect the information are R2, R1, and H1. This means either the host that receives the fragmented packet or any router that transports the fragmented packet.

Required Information I.NI.01, I.NI.04

Scenario Dependency S.NG.01, S.NG.03

B.4.16 DoS IP Fragment Reassembler (S.NM.04)

Description

M2 sends a packet to H1. M2 adds header extension to the packet and fragments it so that the first fragment does not contain a transport layer (TCP or UDP) header. R2, which filters based on the transport layer header, does not have the necessary information to come to a filter decision. It needs to wait at least for the second fragment and reassemble them to be able to filter the packet based on transport header fields. This consumes resources on R2. M2 can send a large number of first fragments without sending the second fragment to exhaust R2's memory and consume processor cycles.

If M2 knows that R2 reassembles the whole packet and not just waits until it sees the transport layer header, M2 can optimise the attack by sending several fragments to create a bigger packet and not send the last fragment that would allow R1 reassembling the whole packet.

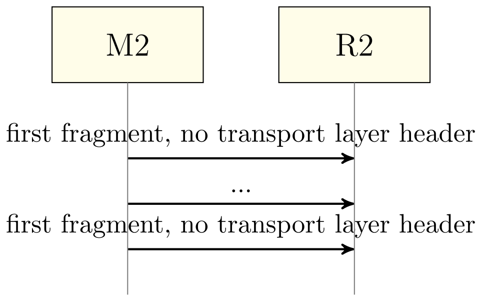
M2
R2
R1
H1
first fragment, no transport layer header
...
first fragment, no transport layer header

Figure B.30: Flow Diagram for the Scenario S.NM.04

Discussion

If a stream of IP fragments does not have a transport layer header in the first fragments, it is a strong indication that an IP fragment attack is in progress. The Packet Fragment Header (I.NI.04) allows determining that the packet is fragmented and the transport layer header (I.TT.01 and I.TU.01) allow showing the position of the headers in the fragments. The access port of the sender and the receiver can be identified with their IP addresses and the process described in scenario S.NG.01.

Required Information I.NI.01, I.NI.04, I.TT.01, I.TU.01

Scenario Dependency S.NG.01, S.NG.03

B.4.17 Unknown Extension Header (S.NM.05)

Description

Another way for M2 to hide a transport layer header is to put it behind an unknown extension header. Any intermediate system that needs to parse the transport layer header, for example, to filter network traffic, is unable to find the transport layer header because it does not know how to parse the unknown extension header.

M2 creates an IP packet with an unknown destination header options and sends it to H1.

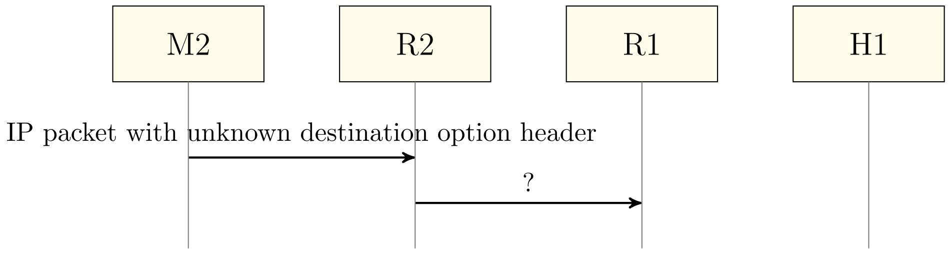

Figure B.31: Flow Diagram for the Scenario S.NM.05

Discussion

To retrace the scenario the actual Extension Header (I.NI.15) and the Packet Header (I.NI.01) are needed. The Extension Header shows the malicious intent and the Packet Header allows identifying the access port (S.NG.01).

Required Information I.NI.01, I.NI.15

Scenario Dependency S.NG.01, S.NG.03

B.4.18 DoS attack with excessive Hop-by-Hop Options (S.NM.06)

Description

M2 creates an IP packet with an excessive number of hop-by-hop options and sends it to H1. The number of options is not limited by RFC 2460 [35]. Each router needs to parse through each option, even if it does not understand them. This attack consumes processor cycles and memory on each router on the path of the packet.

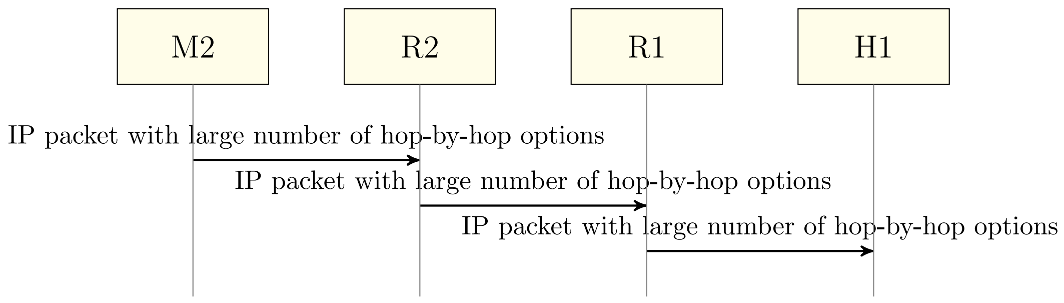
M2
R2
R1
H1
IP packet with large number of hop-by-hop options
IP packet with large number of hop-by-hop options
IP packet with large number of hop-by-hop options

Figure B.32: Flow Diagram for the Scenario S.NM.06

Discussion

To retrace the scenario, the Packet Header (I.NI.01) with the Hop-by-Hop Options Header (I.NI.02) and any Unknown Options (I.NI.16) needs to be analysed. The excessive number of hop-by-hop options in a single IP header identifies this attack. The access port of the sender and the receiver can be identified based on their IP address (S.NG.01).

Required Information I.NI.01, I.NI.02, I.NI.16

Scenario Dependency S.NG.01, S.NG.03

B.4.19 IP Options as a Covert Channel (S.NM.07)

Description

M2 sends IP packets to H1 and encodes covert messages into IP header fields such as the flow label, or the Pad1 or PadN option. Other header fields such as the traffic class or the hop limit could also be used. They are a bit more limiting in the number of covert bytes per IP packet that can be transported. The destination option header can be quiet easily be used to transmit covert messages. Almost all IP header fields and extension headers could be misused to transport covert messages.

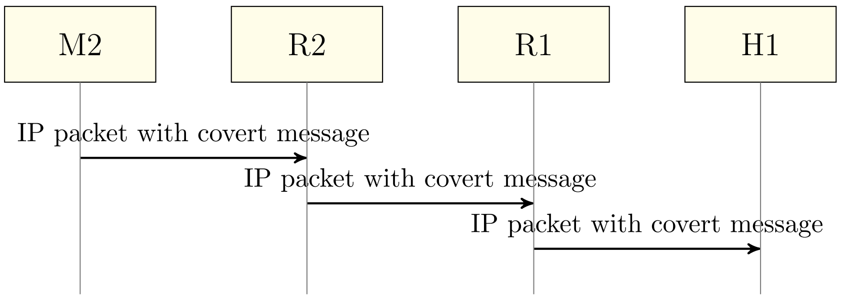
M2
R2
R1
H1
IP packet with covert message
IP packet with covert message
IP packet with covert message

Figure B.33: Flow Diagram for the Scenario S.NM.07

Discussion

Because almost all IP header fields and extension headers can be misused to transport covert messages all the IP header and all extension headers need to be available to retrace this scenario. The access port of the sender and the receiver can be identified based on their IP addresses (S.NG.01).

Required Information I.NI.01, I.NI.02, I.NI.03, I.NI.04, I.NI.05, I.NI.15, I.NI.16

Scenario Dependency S.NG.01, S.NG.03

B.4.20 Specially created Header Option (S.NM.08)

Description

M2 generates packets with a packet fuzzer and sends them to R2. Errors in the implementation of the IP packet parser might lead to the crash of the parsing process or even the whole operating system of R2.

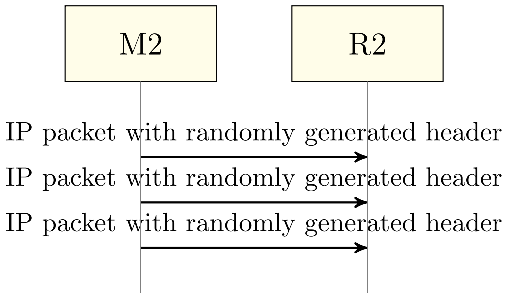
M2
R2
IP packet with randomly generated header
IP packet with randomly generated header
IP packet with randomly generated header

Figure B.34: Flow Diagram for the Scenario S.NM.08

Discussion

To retrace the scenario, the Packet Header (I.NI.01) and all extension headers are needed. The access port of the sender and the receiver can be identified based on their IP addresses (S.NG.01).

Required Information I.NI.01, I.NI.02, I.NI.03, I.NI.04, I.NI.05, I.NI.15, I.NI.16

Scenario Dependency S.NG.01, S.NG.02

B.4.21 Administrative Access (S.NM.09)

Description

M2 manages to get administrative access to R2 and full control over the router.

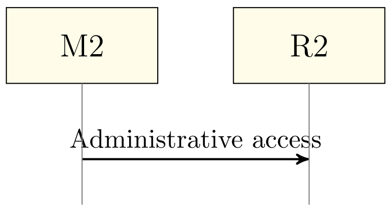

Figure B.35: Flow Diagram for the Scenario S.NM.09

Discussion

M2 has several means get administrative access to R2, for example, Simple Network Management Protocol, Telnet, or Secure Shell. These management protocols are out of the scope of this thesis and are ignored. However, administrative access on a router allows M2 manipulating the Routing Table and make changes to the routing protocols. Such changes would be reflected in I.NI.03, I.NR.04, I.NO.05 and I.NO.06.

Required Information -

Scenario Dependency -

B.4.22 DoS with the Router Alert Option (S.NM.10)

Description

M2 creates IP packets with the router alert option and sends them to H1. The transit routers R2 and R1 are forced to parse the packets past the Packet Header and examine the content more closely ([112]). The attack consumes processor cycles and memory on R2 and R1.

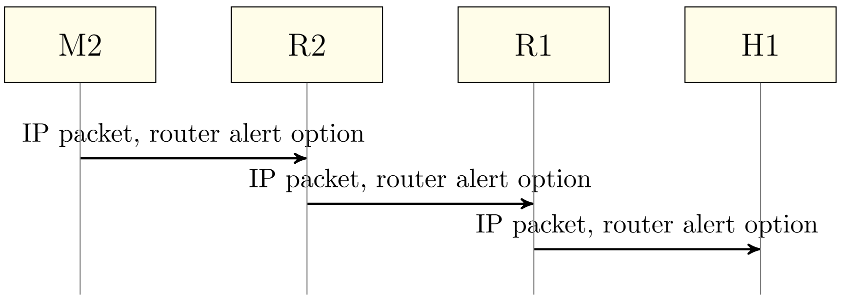
M2
R2
R1
H1
IP packet, router alert option
IP packet, router alert option
IP packet, router alert option

Figure B.36: Flow Diagram for the Scenario S.NM.10

Discussion

To retrace the scenario, the Packet Header (I.NI.01) and Packet Hop-by-Hop Options Header (I.NI.02) is needed. An excessive number of packets with the Router Alert Option indicates this scenario. The access port of the sender can be identified with the source IP address (S.NG.01). Devices that can collect the information are systems in the routing path of the packet.

Required Information I.NI.01, I.NI.02

Scenario Dependency S.NG.01, S.NG.03

B.4.23 DoS Last Hop Router (S.NM.11)

Description

M2 sends a large number of IP packets with destination addresses in the range of N1 but to addresses not used by any host. R1 sends three NS messages for each new destination address and waits for a reply or until it runs into the timeout for each message. This consumes memory and processor cycles and might lead to unresponsiveness of the router.

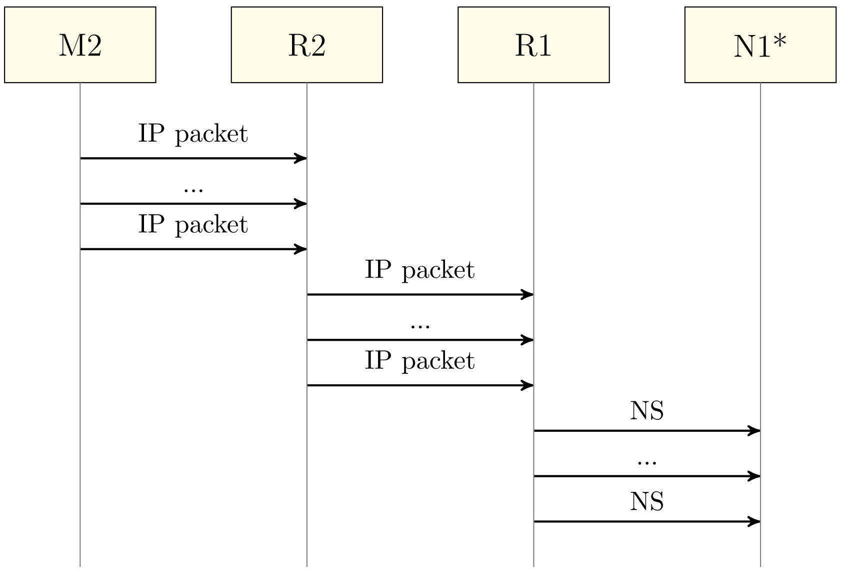

Figure B.37: Flow Diagram for the Scenario S.NM.11

Discussion

To retrace the scenario, the Neighbour Cache (I.NC.06) is needed. Each failed attempt to resolve the data-link layer address of a non-existing IP creates a negative entry. Someone might expect a corresponding number of ICMP Destination Unreachable messages being sent to M2. However, the router limits the number of ICMP error messages (see I.NC.34).

To retrace the source, at least the ICMP Destination Unreachable message (I.NC.28) needs to be retained. The payload contains the original IP packet that triggered the error message. It can be used to retrace the access port of the attacker (S.NG.01).

Required Information I.NC.06, I.NC.28, I.NI.01, I.NI.02

Scenario Dependency S.NG.01, S.NG.03

B.4.24 Spoof NA messages (S.NM.12)

Description

M2 sends an NA message to H2 that says to use the data-link layer address of M2 to reach the network layer address of R2. Consequently, H2 addresses its traffic for the default router R2 with the data-link layer address of M2. M2 has successfully redirected H2's traffic.

A variation of the attack can be used to redirect traffic to a DoS victim by creating NA messages with the data-link layer address of the DoS victim.

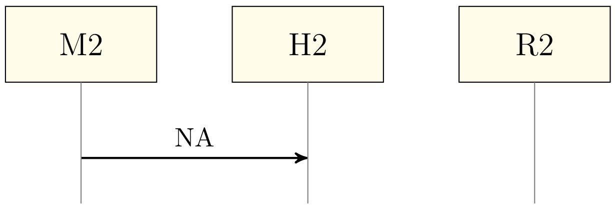

Figure B.38: Flow Diagram for the Scenario S.NM.12

Discussion

Spoofed NA messages can be detected by observing changes in the Neighbour Cache (I.NC.06). However, to retrace the attack the actual NA message (I.NC.04) including the Packet Header (I.NI.01) and the Frame Header (I.DE.01) is needed. With the data-link layer source address and the Filter Database (I.DB.05) the source port used in the attack can be determined.

Required Information I.NC.04, I.NC.06, I.NI.01, I.DE.01, I.DB.05

Scenario Dependency S.NG.02

B.4.25 Spoof NS messages (S.NM.13)

Description

M2 sends an NS message with the source address of R2 and a source data-link layer option to H2. Consequently, H2 updates its neighbour table and addresses its traffic for the default router R2 with the data-link layer address of M2. M2 has successfully redirected H2's traffic.

A variation of the attack can be used to redirect traffic to a DoS victim by creating NS messages with the source data-link layer option containing the address of the DoS victim.

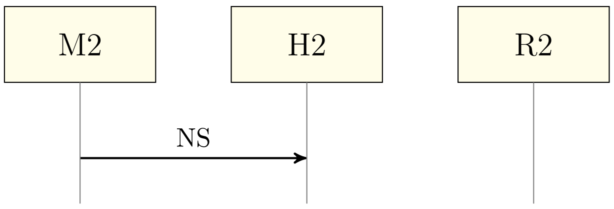

Figure B.39: Flow Diagram for the Scenario S.NM.13

Discussion

Spoofed NS messages can be detected by observing changes in the Neighbour Cache (I.NC.06). However, to retrace the attack the actual NS message (I.NC.03) including the Packet Header (I.NI.01) and the Frame Header (I.DE.01) is needed. With the data-link layer source address and the Filter Database (I.DB.05) the source port used in the attack can be determined.

Required Information I.NC.03, I.NC.06, I.NI.01, I.DE.01, I.DB.05

Scenario Dependency S.NG.02

B.4.26 DoS Neighbour Cache (S.NM.14)

Description

M2 sends NS messages with different source IP addresses and source data-link layer options to H2. H2 adds them to the neighbour cache. When they start to timeout, H2 sends NUD messages and M2 keeps replying to them to keep the entries active in the neighbour cache of H2.

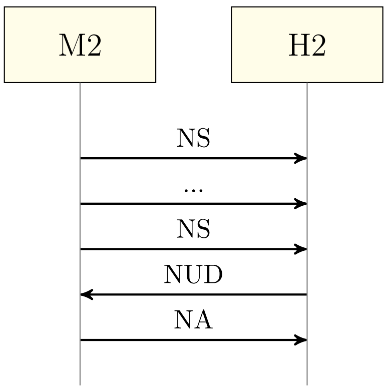

Figure B.40: Flow Diagram for the Scenario S.NM.14

Discussion

The flooding of the Neighbour Cache (I.NC.06) can be detected by simply monitoring it. However, to retrace the attack the actual NS message (I.NC.03) including the Packet Header (I.NI.01) and the Frame Header (I.DE.01) is needed. With the data-link layer source address and the Filter Database (I.DB.05) the source port used in the attack can be determined.

Required Information I.NC.03, I.NC.06, I.NI.01, I.DE.01, I.DB.05

Scenario Dependency S.NG.02

B.4.27 Spoof RA messages (S.NM.15)

Description

M2 sends RA messages into the network N2. H2 adds M2 to the default router list. Depending on the host operating system (RFC 4861 [56], p. 56) H2 uses M2 or R2 as the default router.

The attack can be combined with scenario S.NM.16 to make sure H2 uses M2 as its default router.

The attacker could also announce additional prefixes as on-link. A victim assumes then that these networks are directly reachable and sends NS messages to resolve the data-link layer address of hosts in these networks. The attacker can then reply with NA messages at will (S.NM.12).

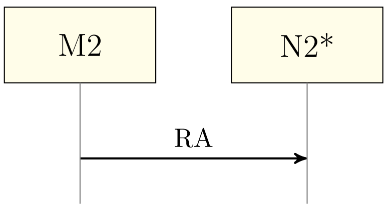

Figure B.41: Flow Diagram for the Scenario S.NM.15

Discussion

The spoofing of RA messages can be detected by monitoring the Default Router List (I.NC.09). However, to retrace the scenario also the Neighbour Cache (I.NC.06) and the Filter Database (I.DB.05) are needed. With the data-link layer address of the new default router in the Neighbour Cache and the Filter Database (I.DB.05), the access port of the attacker can be determined.

This scenario can be retraced with the current set of significant information.

Required Information I.NC.06, I.NC.09, I.NC.05

Scenario Dependency S.NG.02

B.4.28 Clean the Default Router List (S.NM.16)

Description

M2 sends a copy of the RA message R2 sends and changes the lifetime to 0. H2 removes R2 from the default router list. RFC 4862 [41] mitigates this attack by ignoring RA messages with a lifetime shorter than two hours that cannot be authenticated.

The attack can also be interesting without spoofing another RA message (S.NM.14) and adding a new router to the default router list, because RFC 2461 [118] states that if the default router list is empty, the sender should assume the target is on-link. However, that sentence was removed in RFC 4861 [56] (obsoletes RFC 2461 [118]), so it depends on the victims implementation of the protocol what happens in such a case. If the victim assumes all nodes are on-link if the default router list is empty, the attacker might just reply with NA messages for every target the attacker wants to redirect traffic for.

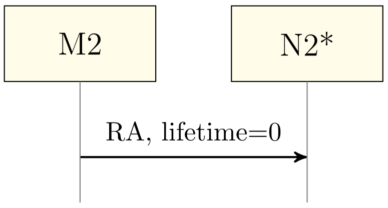

Figure B.42: Flow Diagram for the Scenario S.NM.16

Discussion

To detect the scenario, monitoring the Default Router List (I.NC.09) would suffice. To determine on which port the attacker sending the false RA messages is actually connected, the whole RA message (I.NC.02) including the Packet Header (I.NI.01) and the Frame Header (I.DE.01) is needed. With the data-link layer source address and the Filter Database (I.DB.05) it is possible to determine the port where the RA messages entered the network.

Required Information I.NC.09, I.NC.02, I.NI.01, I.DE.01, I.DB.05

Scenario Dependency S.NG.02

B.4.29 Flood the Default Router List (S.NM.17)

Description

M2 sends an excessive number of RA messages to flood the default router list of H2.

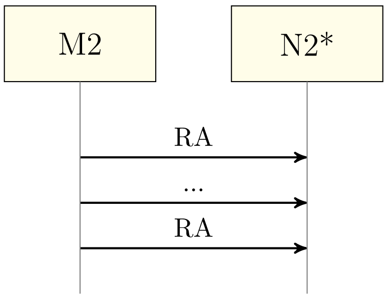

Figure B.43: Flow Diagram for the Scenario S.NM.17

Discussion

To detect the attack, the Default Router List (I.NC.09) would suffice. To determine the access port where the random RA messages enter the network, the whole RA message (I.NC.02) including the Packet Header (I.DE.01) and the Frame Header (I.DE.01), and the Filter Database (I.DB.05) are needed. The data-link layer source address allows, with the help of the Filter Database, to determine the port where the RA messages entered the network.

Required Information I.NC.09, I.NC.02, I.NI.01, I.DE.01, I.DB.05

Scenario Dependency S.NG.02

B.4.30 Lower Hop-Count (S.NM.18)

Description

M2 sends a copy of the RA message R2 sends but changes the hop-count to a very low number (e.g. 1). H2 uses the lower hop-count from the new RA message. Most of packets, created onwards, are dropped by a router because they reach a hop-count of zero before reaching their destination. The router dropping the packet informs the sender with an ICMP Time Exceeded (type 3, code 0) message about the reason for dropping the packet.

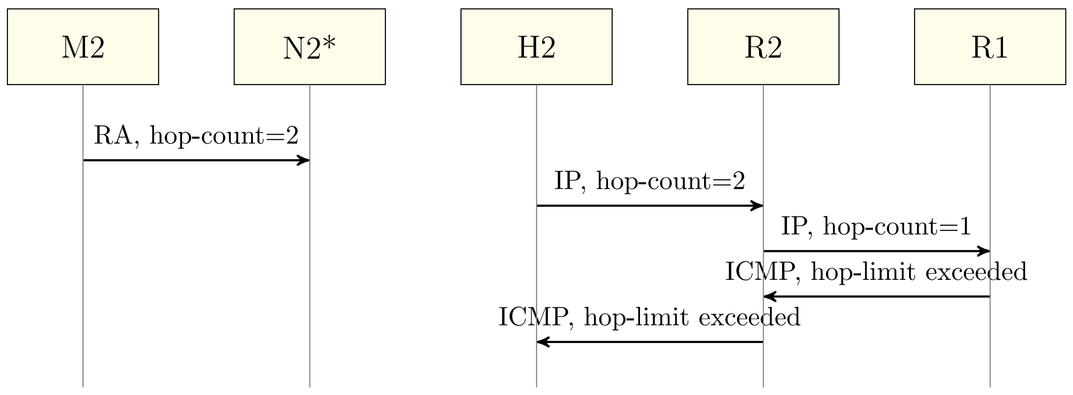

Figure B.44: Flow Diagram for the Scenario S.NM.18

Discussion

The scenario can be detected by monitoring the Default Router List (I.NC.09). To determine the access port where the malicious RA messages enter the network, the whole RA message (I.NC.02) including the Packet Header (I.NI.01) and the Frame Header (I.DE.01), and the Filter Database (I.DB.05) are needed. The data-link layer source address allows, with the help of the Filter Database, to determine the port where the RA messages entered the network.

Required Information I.NC.09, I.NC.02, I.NI.01, I.DE.01, I.DB.05

Scenario Dependency S.NG.02

B.4.31 DoS Auto-Configuration (S.NM.19)

Description

H2 has to do DAD after generating an IP address and before using it. H1 sends, therefore, and NS message for the generated address. When no other host replies, H2 is sure the address is not used already by another node in the same network. H1 does that for its link-local and any global address it generates by auto-configuration or addresses assigned by DHCP. If H2 receives an NA message during DAD, the IP operation should be disabled [41]. M2 interferes with the process by replying to any NS message H2 sends with an NA message. M2 is claiming that it is already using the address H2 tries to use.

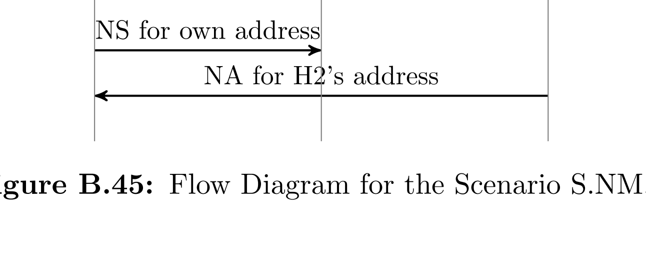

Figure B.45: Flow Diagram for the Scenario S.NM.19

Discussion

Detecting this scenario is harder than retracing it. An unusual high number of NS messages might indicate that the attack is taking place but the attack might be also just directed against a single host with a single NA message. In that case, an additional software that is able to verify NS/NA messages entering a switch port (see Section 9.4.6) is needed.

To retrace the attack NA messages (I.NC.04) need to be monitored. If one malicious NA messages including the Packet Header (I.NI.01) and the Frame Header (I.DE.01) is captured, it is possible to determine the access port the attacker used with the help of the data-link layer source address and the Filter Database (I.DB.05).

Required Information I.NC.04, I.NI.01, I.DE.01, I.DB.05

Scenario Dependency S.NG.02

B.4.32 Traffic Redirect with ICMP Redirect Messages (S.NM.20)

Description

M2 sends a spoofed ICMP Redirect messages to H2 to inform it that M2 is a better next-hop to reach a specific destination (e.g. fd00:0:0:666::/64). M2 spoofs the message so it appears to be coming from R2. H2 sends all subsequent IP packets for the same destination to M2 instead of R2.

M2 can even add in the target data-link layer option its own data-link layer address to optimise the attack. It saves H2 the step to determine the data-link layer address of M2 with an NS/NA message exchange. Further, M2 can reply to NUD messages to keep the redirect active on H2 or can send another redirect message later that informs H2 to use R2 again to conceal the attack. In addition, M2 can either pretend to be the destination network of the traffic and reply accordingly or forward the traffic to the original destination if M2 is only interested in observing the traffic.

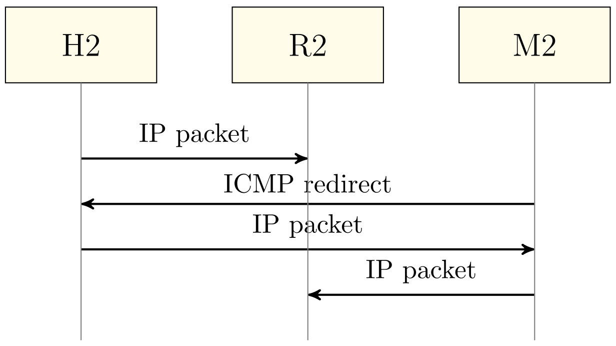

Figure B.46: Flow Diagram for the Scenario S.NM.20

Discussion

Malicious ICMP Redirect messages (I.NC.05) can be detected by monitoring them. In a typical access network with one default gateway it is not do be expected to see any ICMP Redirect messages. With the ICMP Redirect (I.NC.05) message including the Packet Header (I.NI.01) and the Frame Header (I.DE.01), and the Filter Database (I.DB.05) it is possible to determine the access port of the attacker.

Required Information I.NC.05, I.NI.01, I.DE.01, I.DB.05

Scenario Dependency S.NG.02

B.4.33 DoS with ICMP Packet Too Big messages (S.NM.21)

Description

IPv6 routers do not fragment IP packets. Instead, they drop the packet and inform the sender that the packet was too big to be sent to the next hop [35][55].

M2 creates ICMP Packet Too Big messages and sends them to H2 for packets H2 sent to H1. H2 is forced to fragment and resend the packet with the lower packet size. M2 has to guess or observe part of an original packet H2 sent to add it to the ICMP messages. Otherwise, H2 is not able to identify the connection that triggered the error and retransmit the appropriate packets.

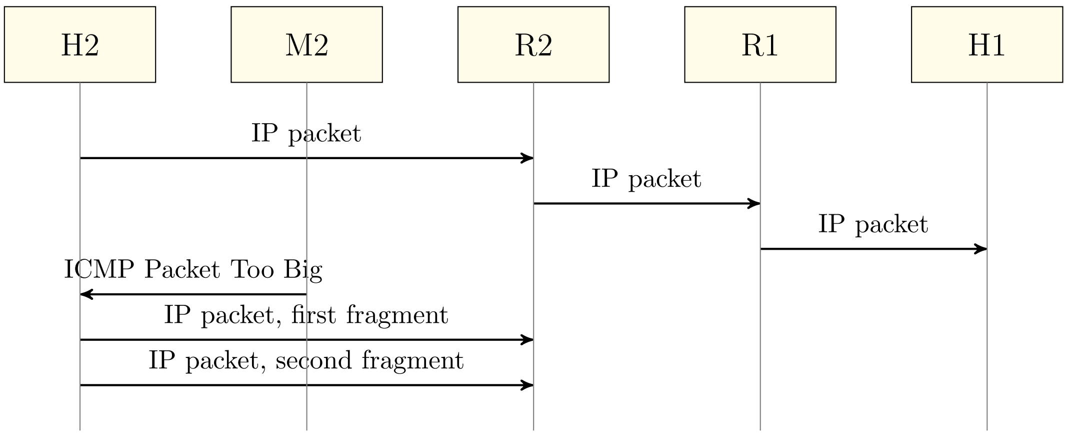

Figure B.47: Flow Diagram for the Scenario S.NM.21

Discussion

Detecting this scenario is quite difficult because the messages are a crucial part of how IPv6 works. Therefore, the mere presence of such messages is not an indication that the attack described in this scenario is in progress. Further, the attacker does not even need to spoof the data-link or network layer source address to make the attack successful because any router along the path of the packet could legitimately send the ICMP Packet Too Big message.

To detect the attack, the Routing Table (I.NI.09) of all routers is needed to determine the path the packet is supposed to take to its destination. This would generate a list of potential sources of such an ICMP messages. The next step is to verify the link MTUs (I.NC.22) of all links on the path to verify if the message was actually needed. If the source does not match with a potential source or packet is actually not too big to be transported, the ICMP Packet Too Big message is malicious. The actual ICMP Packet Too Big message (I.NC.29) including the Packet Header (I.NI.01 and the Frame Header (I.DE.01) combined with the Filter Database (I.DB.05) allows identifying the access port of the attacker.

Required Information I.NI.01, I.NI.09, I.NC.22, I.NC.29, I.DE.01, I.DB.05

Scenario Dependency S.NG.03

B.4.34 DoS with ICMP Error Messages (S.NM.22)

Description

An ICMP Error message can reset higher-layer network protocols. For example, an ICMP Port Unreachable message can reset a TCP connection to the referred port.

M2 sends a spoofed ICMP Port Unreachable message for a packet previously sent from H2 to H1. M2 has to guess or observe part of an original packet H2 sent to add it to the ICMP messages. Otherwise, H2 is not able to identify the connection that triggered the error.

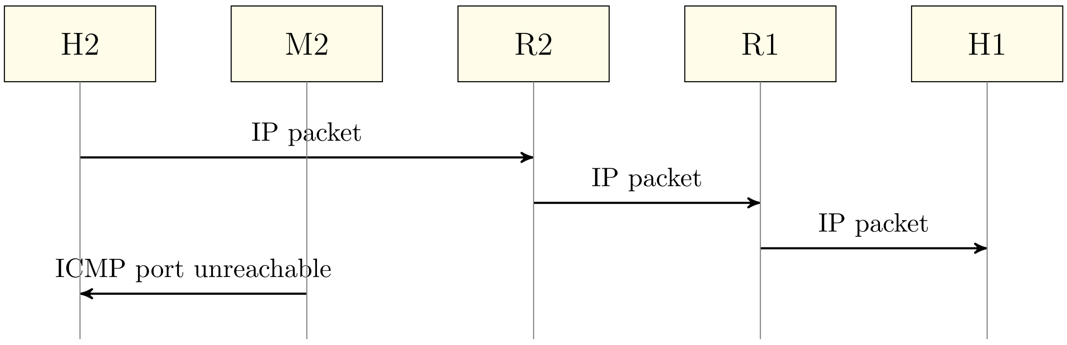

Figure B.48: Flow Diagram for the Scenario S.NM.22

Discussion

The mere present of the ICMP Port Unreachable message does not allow concluding that this scenario is in progress. Such ICMP error messages have legitimate uses and might appear in genuine connections as well.

An ICMP Port Unreachable messages is more suspicious if a session is fully established on the transport layer before the ICMP message was received. That means that the Filter Rules (I.NI.11) changed, the routing changed, or the application that established the session crashed. While all of that is possible, it is not likely that is happens very often.

To retrace the attack the actual ICMP messages is needed. Further, the attacker needs to spoof the source IP address of the ICMP message for a successful attack. The detection and retracing of source IP address spoofing is discussed in scenario S.NM.01.

Required Information I.NC.28, I.NC.29, I.NC.30, I.NC.31, I.NC.35, I.NI.11, I.TT.04, I.TU.04

Scenario Dependency S.NM.01, S.NM.03

B.4.35 Covert Channel with ICMP messages (S.NM.23)

Description

ICMP messages are crucial for the correct function of IP protocol. For example, ICMP Packet Too Big messages cannot be filtered without impairing the correct function of the IP protocol. Therefore, ICMP messages are ideal to transport covert messages. Because ICMP error message usually contain the original packet that triggered the error message, they can easily misused to tunnel IP packets by replacing the erroneous packet with packets that should be tunnelled.

M2 sends ICMP Port Unreachable messages to M1. The message should carry the packet that triggered the error but instead M2 adds a covert message as payload.

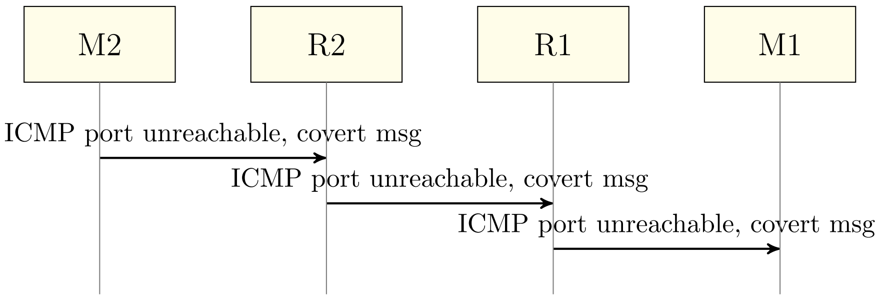
M2
R2
R1
M1
ICMP port unreachable, covert msg
ICMP port unreachable, covert msg
ICMP port unreachable, covert msg

Figure B.49: Flow Diagram for the Scenario S.NM.23

Discussion

While retracing a covert channel is relatively easy once the covert channel is detected, the detection is difficult. The detection is well discussed in current literature [119, 120, 121] and outside of the scope of this research.

To be able to retrace the attack, the actual ICMP message is needed to show the covert message and the Packet Header (I.NI.01) with the Frame Header (I.DE.01) to retrace where the message entered the network. Further, all Packet (I.NI.01), Segment (I.TT.01) and Datagram Headers (I.TU.01) in proximity of the event are need to show that the ICMP Port Unreachable message is not triggered by protocol event.

Required Information I.NI.01, I.DE.01, I.NC.01, I.NC.02, I.NC.03, I.NC.04, I.NC.05, I.NC.28, I.NC.29, I.NC.30, I.NC.31, I.NC.32, I.NC.33, I.TT.01, I.TU.01

Scenario Dependency S.NG.03

B.4.36 DoS with ICMP message flooding (S.NM.24)

Description

An attack might flood a victim with a large number of ICMP messages.

M2 sends an excessive number of ICMP Packet Too Big messages to H2.

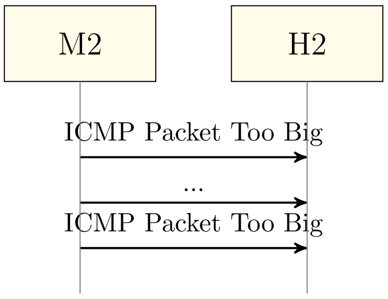

Figure B.50: Flow Diagram for the Scenario S.NM.24

Discussion

To retrace this scenario, the actual ICMP messages (I.NC.28 - I.NC.33) including the Packet Header (I.NI.01) and the Frame Header (I.DE.01), and the Filter Database (I.DB.05) are needed. The ICMP message shows the malicious behaviour and the different data-link layer source address can be used to locate the access port of the attacker.

Required Information I.NI.01, I.DE.01, I.NC.28, I.NC.29, I.NC.30, I.NC.31, I.NC.32, I.NC.33

Scenario Dependency S.NG.02

B.4.37 Spoof Tunnel Packet Source (S.NM.25)

Description

M2 sends a 6in4 packet to R2 with the source address of R1 in the outer IPv4 header. R2 exams the next header as the packet is address to itself and decapsulates the inner IPv6 packet and forwards it appropriately. M2 can choose whatever inner IPv6 packet suits the purposes, for example, a packet from H1 to H3. H3 has no mean to distinguish the packet from a genuine one.

The same attack can be run from M1. This makes it even harder for R2 to detect the malicious packet because the packet is originating in a different data-link layer.

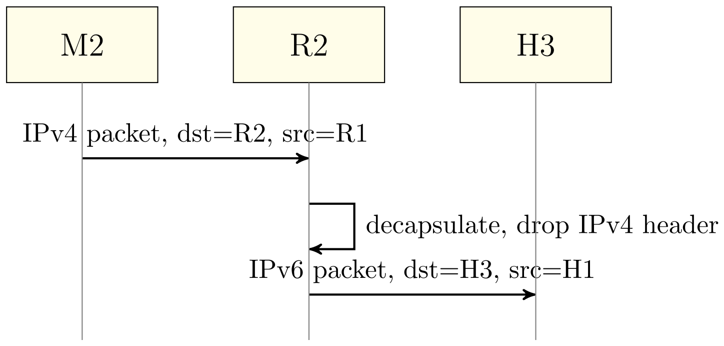
M2
R2
H3
IPv4 packet, dst=R2, src=R1
decapsulate, drop IPv4 header
IPv6 packet, dst=H3, src=H1

Figure B.51: Flow Diagram for the Scenario S.NM.25

Discussion

To detect a spoofed 6in4 packet, the tunnel endpoint needs to be able to detect IPv4 source spoofing. RPF allows doing that in the described scenario. However, if M1 runs the attack, RPF on R2 is not able to detect the attack and RPF also needs to be implemented on R3. This shows how difficult it is to detect such an attack if the tunnel is run over a public network such as the Internet.

To retrace the scenario, the Tunnel IP Header (I.N6.12, I.NG.02) is needed, including the Frame Header (I.DE.01), and the Filter Database (I.DB.05) is needed. To retrace the same attack originating from M1, the Tunnel IP Header does not suffice. The packet that arrives at R2 does not have the Frame Header as M1 created it. Therefore, RPF needs to be activated on all access-layer routers (I.N6.09) and violations against the RPF filtering need to be logged with the Tunnel IP Header and Frame Header.

Required Information I.N6.09, I.N6.12, I.NG.02, I.DE.01, I.DB.05

Scenario Dependency S.NG.01

B.4.38 IPv4 Reassembly Attack against the Tunnel Endpoint (S.NM.26)

Description

M1 sends a large number of fragmented 6in4 packet to the tunnel endpoint. Each chain of IPv4 fragments lacks the last fragment. Therefore, the tunnel endpoint R2 has to buffer all the other fragments until it runs into a timeout and drops the whole chain of received fragments.

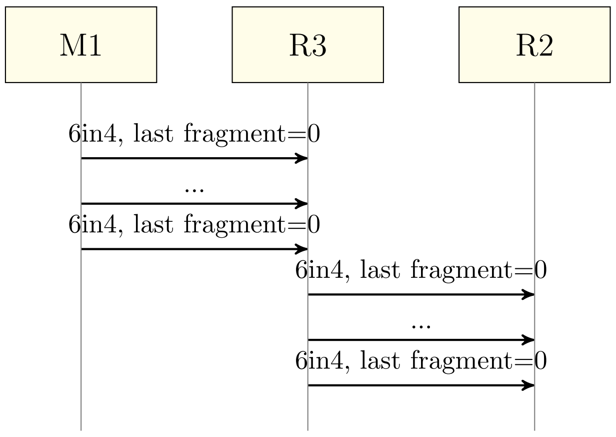

Figure B.52: Flow Diagram for the Scenario S.NM.26

Discussion

To retrace this scenario, the Tunnel IP Header (I.N6.12) is needed. The Tunnel IP Header shows the fragmentation and the missing last fragment. The underlying problem of the attack is also the IPv4 source address spoofing. The retracing of the attacker is similar as discussed in the previous scenario S.NM.25.

Required Information I.N6.12, I.NG.02

Scenario Dependency S.NG.03, S.NM.25

B.4.39 Extend Attack Reach with Encapsulated Packets (S.NM.27)

Description

M1 sends an encapsulated (6in4 or GRE) packet to R2 with the source spoofed so the packet appears to come from R1. The encapsulated packet contains a packet that would otherwise not reach N2. Possibilities are to send packets with a link-local addresses, the loopback address or any other invalid source addresses. Such packets would normally be filtered or at least not forwarded. Another option is to send packets with a hop count of 255 in the hope the tunnel endpoint does not decrement the hop limit. This opens the network directly behind the tunnel endpoint to a variety of attacks that would otherwise be evaded by relying on the hop count of 255 or link-local source and destination addresses.

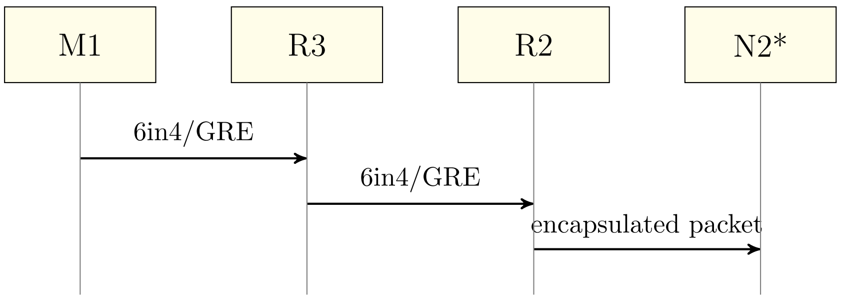

Figure B.53: Flow Diagram for the Scenario S.NM.27

Discussion

The extended reach attack uses also packets encapsulated in a tunnel header with a spoofed IPv4 source address. Therefore, the detection and retracing of the attack is identical to scenario S.NM.25. To show the malicious intent the actual packet including the Tunnel Header (I.N6.12, I.NG.02) and the Packet Header (I.NI.01) is needed.

A few tests with the scenario revealed that Linux correctly decrements the hop limit in encapsulated IPv6 packets and drops packets with a link-local address or the loopback address as the source address. However, packets with the source address of the router (global scope) are forwarded. This might enable attack vectors where traffic is filtered based on the IPv6 source address.

Required Information I.N6.12, I.NG.02, I.NI.01

Scenario Dependency S.NG.03, S.NM.25

B.4.40 Circumvent Traffic Filtering (S.NM.28)

Description

R2 filters traffic from M2 to H1. M2 circumvents the filtering by encapsulating the packet by itself, spoofs the source address as R2's and sends it to R1. Because R2 does not inspect encapsulated traffic and allows IPv4 traffic from R2 to R1, the traffic reaches R1. R1 decapsulates the traffic and forwards it to H1.

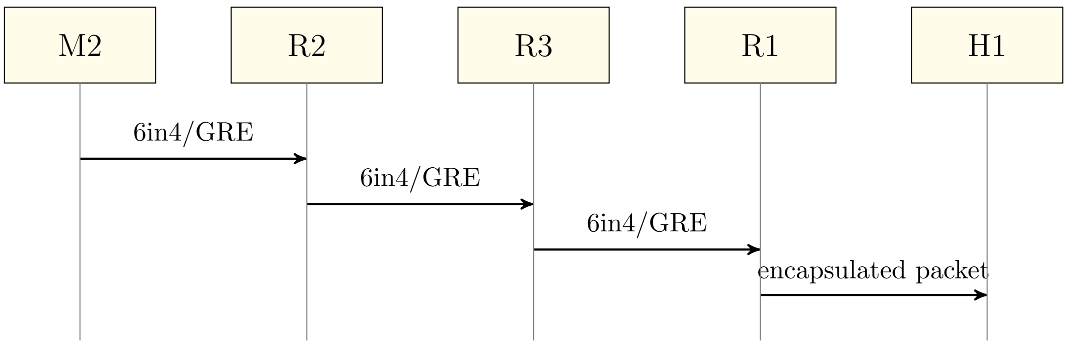

Figure B.54: Flow Diagram for the Scenario S.NM.28

Discussion

The circumvention of IPv6 traffic filtering by encapsulating the traffic into a 6in4 packet is also based on spoofing the source IPv4 address of the tunnel packet. Therefore, the detection and retracing is identical to scenario S.NM.25. To show the malicious intent the actual packet including the Tunnel Header (I.N6.12, I.NG.02) and the Packet Header (I.NI.01), and the Traffic Filter (I.NI.11) are needed.

Required Information I.N6.12, I.NG.02, I.NI.11

Scenario Dependency S.NG.03

B.4.41 DoS the Tunnel with Malicious Payload Addressing (S.NM.29)

Description

M1 creates a encapsulated packet (6in4 or GRE) that contains an ICMPv6 Echo Request message from H3 to H1. The addressing is chosen that the tunnel is used as often as possible. The outer IPv4 header uses the source address of R1 and the destination address of R2. When R2 receives the packet, it examines the destination address of the encapsulated IPv6 packet. R2 routes the packet back out on the tunnel interface because the IPv6 destination is on the other side of the tunnel. The tunnel works as an attack amplifier.

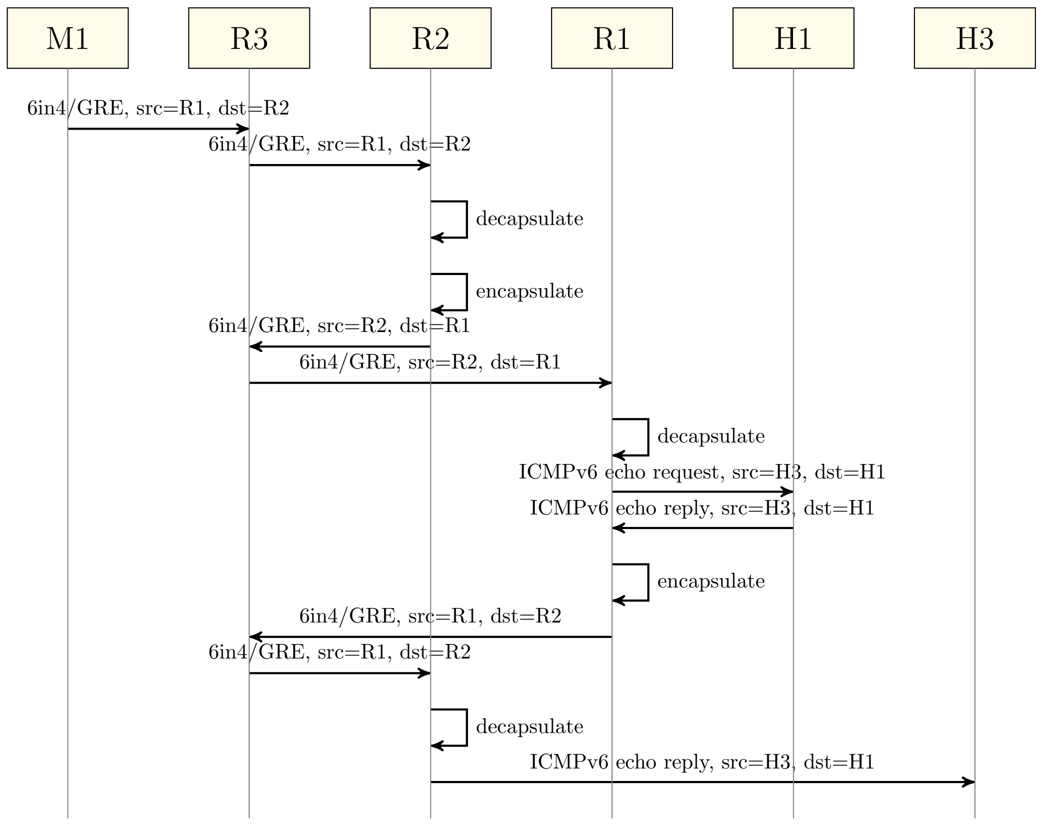

Figure B.55: Flow Diagram for the Scenario S.NM.29

Discussion

This attack can be detected and retraced when IPv4 address spoofing is detectable and retraceable. See scenario B.4.37 for a longer discussion of the detection and the retracing of source IP address spoofing.

This attack can also be detected with RPF based on the inner IPv6 Packet Header (I.NI.01). The tunnel endpoint could implement RPF on the tunnel interface and log any packet that is received on the tunnel interface that is sent out the tunnel interface again. To do so the router references the Routing Table (I.NI.09).

Required Information I.N6.12, I.NG.02, I.NI.01, I.NI.09

Scenario Dependency S.NG.03, S.NM.25

B.4.42 Lower Tunnel MTU (S.NM.30)

Description

If the tunnel (6in4 or MTU) uses Path MTU Discover to determine the MTU for the tunnel, it is vulnerable to an attack that lowers the tunnel MTU. M1 sends spoofed ICMPv4 Fragmentation Needed And Don't Fragment Bit Set message to R2. M1 sets the source of the message to the address of R3 and the destination to R2. R2 lowers the MTU according to the MTU in the ICMPv4 message.

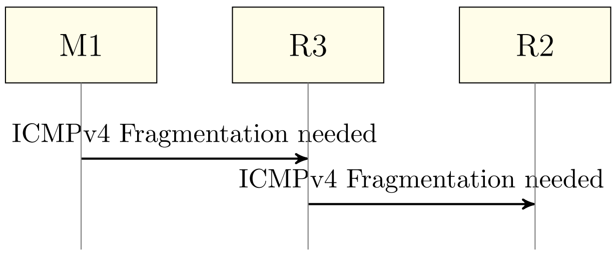

Figure B.56: Flow Diagram for the Scenario S.NM.30

Discussion

The attack requires the attacker to spoof the source address of the malicious ICMPv4 message. See scenario B.4.37 for a longer discussion of the detection and the retracing of source IP address spoofing.

The attacker can also choose not to spoof the source address. This is possible because the ICMPv4 message can originate from any router on the path between the tunnel endpoints. In that case, the whole ICMPv4 message (I.N6.14) including the IPv4 Header (I.N6.12) and the IPv4 Routing Table (I.N6.15) of all routers at the time of the attack is needed. The packet path can be reconstructed with the IPv4 Routing Tables. This allows creating a list of possible sources of the ICMPv4 error message. Further, with the Interface Link MTU (I.NC.22), it is possible to verify if the error message was correct or not.

Required Information I.N6.12, I.N6.14, I.N6.15, I.NC.22

Scenario Dependency S.NG.03, S.NM.25

B.4.43 DoS on GRE tunnels with Sequence Numbers (S.NM.31)

Description

GRE tunnel can be configured to use sequence numbers to ensure in sequence packet delivery. M1 sends spoofed GRE packets with wrong sequence numbers. M1 choses the next sequence number R2 expects. R2 drops the next packet arriving from R1 because, according to the sequence number, the packet was already received.

The attack is even easier if the victim uses a buffer to store packets with sequence numbers ahead of the counter. The attacker could place packets in the buffer that would be used later instead of the genuine packets from R1.

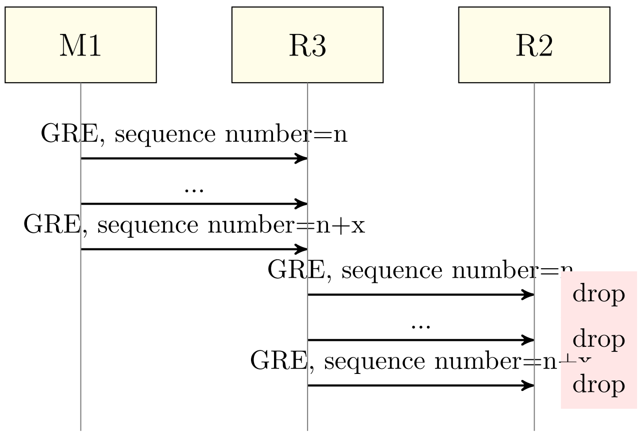

Figure B.57: Flow Diagram for the Scenario S.NM.31

Discussion

To detect this attack, the actual GRE Protocol Header (I.NG.02) and the Tunnel IP Header (I.N6.12) is helpful. It allows detecting that two packets with the same sequence number but different payload (IPv6 Packet Header I.NI.01) were received.

The attack can be retraced if the source address spoofing of the outer IPv4 header can be detected and retraced. See scenario B.4.37 for a longer discussion of the detection and the retracing of source IP address spoofing.

Required Information I.NG.02, I.N6.12, I.NI.01

Scenario Dependency S.NG.03, S.NM.25

B.4.44 Redirect Traffic with more Specific Routes (S.NM.32)

Description

RIPng does not provide a mechanism to authenticate peers. Instead, the protocol relies on IPSec for authentication. No hardware vendor supports currently to use IPSec for authentication [13]. Therefore, an attacker that is on-link with a RIPng router is in a position to inject malicious RIPng packets.

M2 sends a RIPng Update to R2 announcing its own address as the next-hop for the destination of choice (e.g. M2 announces the IP address range fd00:0:0:666::/64).

A variation of the attack uses the Routing Table Entry with the metric 0xff to specify the next-hop explicitly. This way, the attacker can redirect traffic to any on-link victim as a DoS attack.

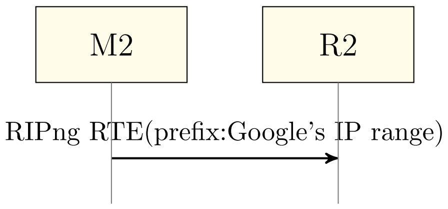

Figure B.58: Flow Diagram for the Scenario S.NM.32

Discussion

This attack can be detected by monitoring the Routing Table (I.NI.09) and compare it with the physical layout (I.DE.08) and the RIP routers (I.NR.02) in the network. In short, it means a router receiving a RIP packet on an interface where no RIPng router should be.

The attack can be retraced with the Routing Table (I.NI.09) because the attacker needs to use his/her own source IP address to receive the traffic. With the help of the Neighbour Cache (I.NC.06) the data-link layer address of the attacker can be determined and with the Filter Database (I.DB.05) the access port of the attacker.

If the attacker modifies the attack to a DoS attack by spoofing the source address of the RIPng packet, the actual RIPng packet (I.NR.13) including the Frame Header (I.DE.01) is needed. With the data-link layer source address of the frame and the Filter Database (I.DB.05) the access port of the attacker can be determined.

Required Information I.NI.09, I.DE.08, I.NR.02, I.NC.06, I.DB.05, I.NR.13

Scenario Dependency S.NG.02

B.4.45 Redirect Traffic with Lower Metric (S.NM.33)

Description

M2 sends a RIPng Update to R2 announcing its own address as the next-hop for N1. M2 uses the lowest possible metric and R2 choses the malicious update over the genuine one. This attack is a DoS attack for the network N2 because return traffic might not reach the original sender. An attacker might use the attack combined with an address spoofing attack (address in the range of N1) to be able to establish a two-way connection with a spoofed IP source address.

If the attacker is in a position to inject routes on multiple routers, he/she might create intentionally a routing loop as a DoS attack.

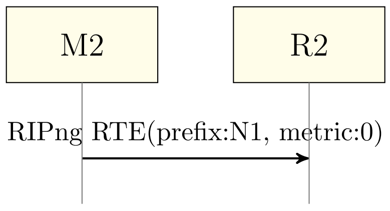

Figure B.59: Flow Diagram for the Scenario S.NM.33

Discussion

This attack can be detected with the Routing Table (I.NR.04). However, to retrace the port where the malicious RIPng packet entered the network the whole RIPng Packet (I.NR.13) including the Packet Header (I.NI.01) and Frame Header (I.DE.01), and the Filter Database (I.DB.05) is needed.

Required Information I.NR.04, I.NR.13, I.NI.01, I.DE.01, I.DB.05

Scenario Dependency S.NG.02

B.4.46 DoS RIPng Router with Requests (S.NM.34)

Description

RIPng router response quickly to request messages with the current routing table.

M2 sends an excessive number of RIPng requests to R2 for the complete routing table (one entry in the list of requested prefixes uses a prefix length of zero). Because creating the response uses more computing power than creating the request and needs more network bandwidth than the request it is an ideal attack vector for a DoS attack. The attacker might choose to spoof the source address of the request to avoid receiving the responses.

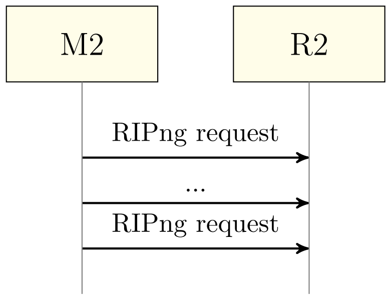

Figure B.60: Flow Diagram for the Scenario S.NM.34

Discussion

To detect this attack, the actual RIPng packets (I.NR.13) are needed. To retrace the attack at least one RIPng packet needs to be captured including the Packet Header (I.NI.01 and the Frame Header (I.DE.01). With the help of the Filter Database (I.DB.05) it is possible to retrace the switch port where the malicious packets entered the network.

Required Information I.NR.13, I.NI.01, I.DE.01, I.DB.05

Scenario Dependency S.NG.02, S.NM.01

B.4.47 DoS RIPng Routers with Triggered Updates (S.NM.35)

Description

RIPng routers might implement triggered updates to advertise changes in the network faster. RFC 2080 [59] suggests a random timeout between two triggered updates of 1 to 5 seconds.

M2 sends updates of new networks in short intervals to trigger updates throughout the RIPng domain. This consumes bandwidth through the network and processor cycles an all RIPng routers. The effectiveness of the attack depends of the timer implementation of the routing daemon.

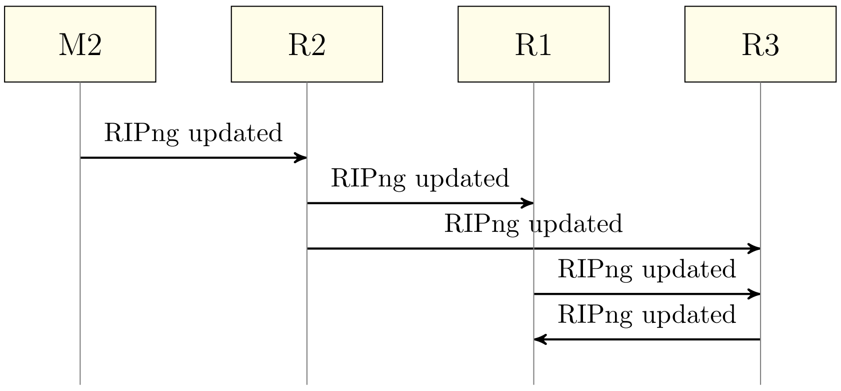

Figure B.61: Flow Diagram for the Scenario S.NM.35

Discussion

To detect this attack, the actual RIPng packets (I.NR.13) are needed. Very frequent changes in the Routing Table (I.NR.04) also indicate the attack. To retrace the attack at least one RIPng packet needs to be captured including the Packet Header (I.NI.01) and the Frame Header (I.DE.01). With the help of the Filter Database (I.DB.05) it is possible to retrace the switch port where the malicious packets entered the network.

This scenario cannot be retraced with the current set of significant information. Therefore, the RIPng packet (I.NR.13) is added to the set of significant information.

Required Information I.NR.13, I.NR.04, I.NI.01, I.DE.01, I.DB.05, I.NR.14

Scenario Dependency S.NG.02

B.4.48 Redirect Traffic with more Specific Routes (S.NM.36)

Description

If OSPF is not configured to use IPSec for authentication, an attacker on-link with an OSPF router is in a position to form a neighbour relationship and to inject malicious OSPF packets.

M2 forms a neighbour relationship with R2 and sends LSAs stating links behind M2 (e.g. M2 announces the IP range fd00:0:0:666::/64).

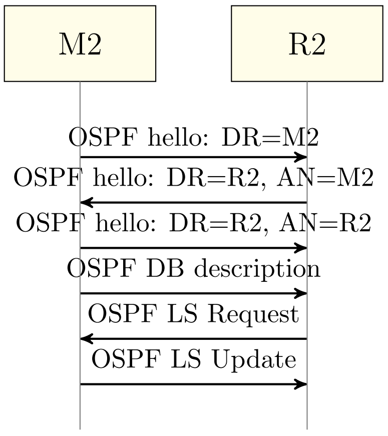

Figure B.62: Flow Diagram for the Scenario S.NM.36

Discussion

This attack can be detected by monitoring the Neighbour Table (I.NO.05). However, the forming of the neighbour relationship is based on the content of the hello packets and they are sent to a multicast IP address. An attacker might spoof his/her own source IP address in order to hide his/her identity. Spoofing the source IP for a full OSPF database exchange is difficult but not impossible if the attacker has knowledge of the network topology.

If the attacker wants to redirect traffic to himself he/she has to use his/her own IP address. Otherwise, the traffic is sent to the spoofed IP address and the attack is not successful. The access port of the attacker can be retraced base on the IP address in the Neighbour Table and the procedure described in scenario S.NG.01.

When the attacker uses a spoofed source, at least one OSPF Packet (I.NO.10) including the Packet Header (I.NI.01) and the Frame Header (I.DE.01) needs to be captured. With the help of the Filter Database (I.DB.05) it is possible to retrace the switch port where the malicious packets entered the network.

Required Information I.NO.05, I.NO.10, I.NI.01, I.DE.01, I.DB.05

Scenario Dependency S.NG.01, S.NG.02

B.4.49 DoS OSPF Router with Topology Changes (S.NM.37)

Description

OSPF routers recalculate the routing table upon changes in the link state database with the Shortest Path First algorithm. A malicious router able to form a neighbour relationship with one router would be able to inject new LSAs to keep the routers in the same area constantly recalculating their routing table.

M2 creates a neighbour relationship and keeps injecting new LSAs forcing the routers to keep recalculating the routing table. Quagga defines several timers that define how long a Shortest Path First calculation should be delayed when a topology change occurs. An attacker could optimise the attack by waiting slightly longer than the delay timer.

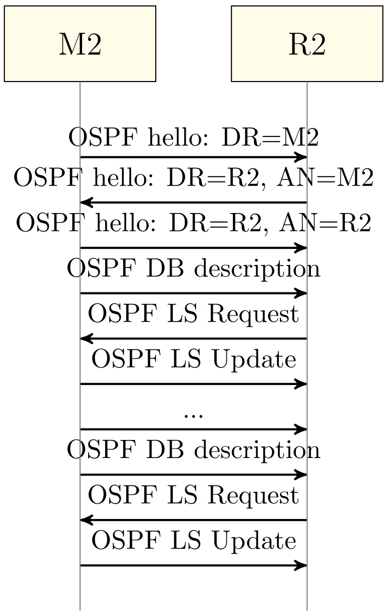

Figure B.63: Flow Diagram for the Scenario S.NM.37

Discussion

This scenario is similar to the previous one but the attacker injects and withdraws LS messages. This could be detected by the frequency changes in the Routing Table (I.NO.07).

Because the goal of the attack is only to keep the other routers in the network calculating their Routing Table, the attacker could use a spoofed IP source address. Therefore, the actual OSPF packet (I.NO.10) including the Packet Header (I.NI.01) and the Frame Header (I.DE.01) needs to be captured. With the help of the Filter Database (I.DB.05) it is possible to retrace the switch port where the malicious packets entered the network.

Required Information I.NO.07, I.NO.10, I.NI.01, I.DE.01, I.DB.05

Scenario Dependency S.NG.02

B.4.50 DoS OSPF Router by Flooding Hello Packets (S.NM.38)

Description

OSPF routers form a neighbour relationship before starting exchange topology information. An attacker might flood a network segment with OSPF hello packets, each packet from a different source. The genuine routers in the same segment would use processor cycles and network bandwidth to reply to the hello packets and running the DR/BDR election.

M2 creates OSPF hello packets with different source parameters and different DR/BDR settings.

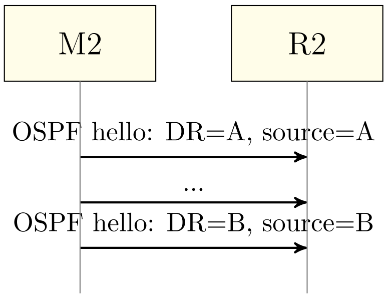

Figure B.64: Flow Diagram for the Scenario S.NM.38

Discussion

As never a neighbour relationship is successfully established, the Neighbour Table (I.NO.05) does not allow detecting or retracing an OSPF Hello packet flooding attack. To detect and retrace the attack the actual OSPF Hello packets (I.NO.10) including the Packet Header (I.NI.01) and the Frame Header (I.DE.01) are needed. With the data-link layer source address and the Filter Database (I.DB.05) the port where the malicious OSPF Hello packets entered the network can be retraced.

Required Information I.NO.10, I.NI.01, I.DE.01, I.DB.05

Scenario Dependency S.NG.02

B.4.51 DoS OSPF Router with Spoofed Hello Packets (S.NM.39)

Description

OSPF routers form neighbour relationships before starting to exchange topology information. The attacker might flood a network segments with OSPF hello packets, each packet from a different source. If the parameter changes in the Hello packet, the neighbour relationship is reset.

Mx connects to the link L7. Mx sends spoofed OSPF hello packets to R2 with slightly modified parameters (e.g. disable the V6 option).

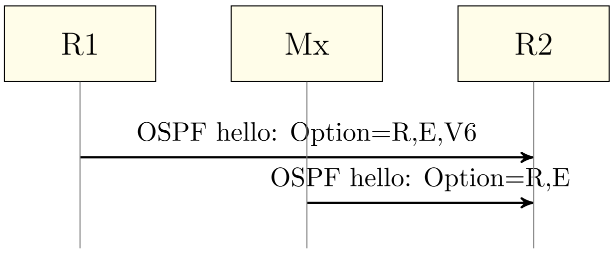

Figure B.65: Flow Diagram for the Scenario S.NM.39

Discussion

This attack can be detected by monitoring the Neighbour Table (I.NO.05). In addition, the attaching of the attacker's system into L7 can be detected by observing the link operation status (I.DE.06). To be able to retrace the reason for the flapping neighbour relationship the actual OSPF packet (I.NO.10) including the Packet Header (I.NI.01) and the Frame Header (I.DE.01) is needed. With the data-link layer source address and the Filter Database (I.DB.05) the port where the malicious OSPF Hello packets entered the network can be retraced.

Required Information I.NO.05, I.NO.10, I.NI.01, I.DE.01, I.DE.06, I.DB.05

Scenario Dependency S.NG.02

B.5 Transport Layer Scenarios

B.5.1 Send a UDP Segment (S.TG.01)

Description

H2 sends an UDP datagram from port 2000 to H1 port 1000. H1 has a process listening on port 1000.

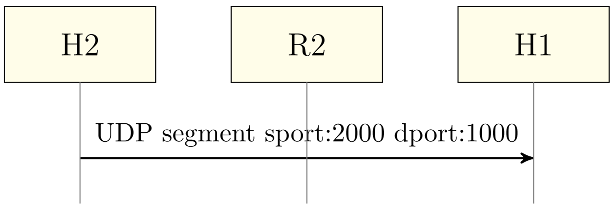

Figure B.66: Flow Diagram for the Scenario S.TG.01

Discussion

Without actually capturing the datagram in front of H2 and H1 it is not possible to prove that a specific datagram was sent from H2 to H1. However, it is possible to show that there was communication between H2 and H1 on a specific UDP port at a specific time. The client programme on H2 creates a socket to send the datagram. This socket is shown in the Active Sockets

(I.TU.03). The traffic filter R2 keeps track of the UDP connection in the Traffic Filter Sessions (I.TU.05) and the host H1 opens first a listening socket and creates later an established socket with H2, both are shown in the Active Sockets. The timestamp when the socket changes from a listening socket to an established socket is when the UDP packet was received by H1. This should correlate with the creation of a new Traffic Filter Session. The scenario S.NG.01 describes how to retrace the access ports based on the systems IP addresses.

This scenario is not fully retraceable without the Datagram Payload (I.TU.02). However, it is not added to the set of significant information because it is likely that it triggers events on H1 when the packet is passed to the next higher layer. This renders the information redundant.

Required Information I.TU.03, I.TU.05

Scenario Dependency S.NG.01, S.NG.03

B.5.2 Send a UDP Segment to a Closed Port (S.TG.02)

Description

H2 sends an UDP datagram from port 2000 to H1 port 1000. Because H1 has no process listening on port 1000, it replies with an ICMP Destination Unreachable (Port Unreachable) message.

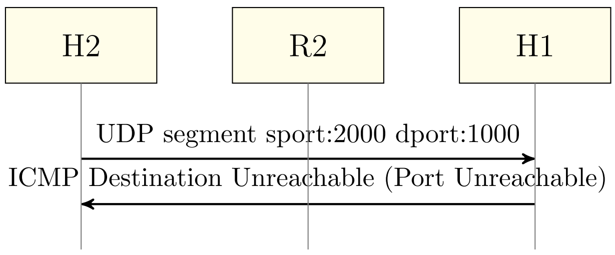
H2
R2
H1
UDP segment sport:2000 dport:1000
ICMP Destination Unreachable (Port Unreachable)

Figure B.67: Flow Diagram for the Scenario S.TG.02

Discussion

The Active Sockets (I.TU.03) show the socket used to send the datagram as before and the Traffic Filter Sessions (I.TU.05) show the connection. H1 does not show any Active Sockets in this scenario. However, the ICMP Destination Unreachable message (I.NC.28) is a good indication that this scenario is in progress. Further, inspection of the payload of the ICMP message allows retracing the involved ports and the involved systems (IP addresses). The scenario S.NG.01 describes how to retrace the access ports based on the systems IP addresses.

Required Information I.TU.03, I.TU.05, I.NC.28

Scenario Dependency S.NG.01, S.NG.03

B.5.3 Establish a UDP Stream (S.TG.03)

Description

H2 and H1 exchange several UDP datagrams between each other.

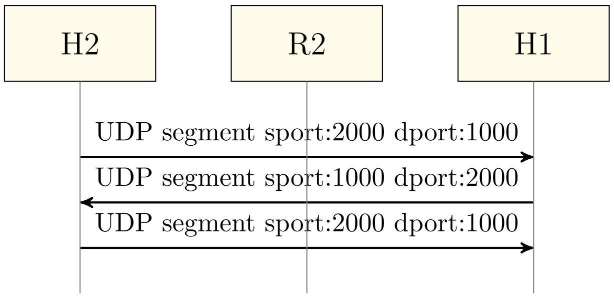
H2
R2
H1
UDP segment sport:2000 dport:1000
UDP segment sport:1000 dport:2000
UDP segment sport:2000 dport:1000

Figure B.68: Flow Diagram for the Scenario S.TG.03

Discussion

This scenario is very similar to the scenario S.TG.01. The only difference in retracing it is that R2 move the state of the traced UDP connection (I.TU.05) from flagged as “unreplied”, after to first packet, to no flag, after the first reply packet, to flagged as “assured”, after receiving the second return packet in either direction. If the connection is flagged as “assured” there were at least three packets exchanged. Unfortunately, the session tracking does not show traffic counter. The reason for this is that the feature was not enabled when the kernel was compiled. This would allow retracing the amount of data transported using a specific session. The scenario S.NG.01 describes how to retrace the access ports based on the systems IP addresses.

Required Information I.TU.03, I.TU.05, I.NC.28

Scenario Dependency S.NG.01, S.TG.01

B.5.4 Establish a TCP Session (S.TG.04)

Description

H2 initiates a TCP session with H1. Sends a few octets data and closes the session. H1 closes the session afterwards.

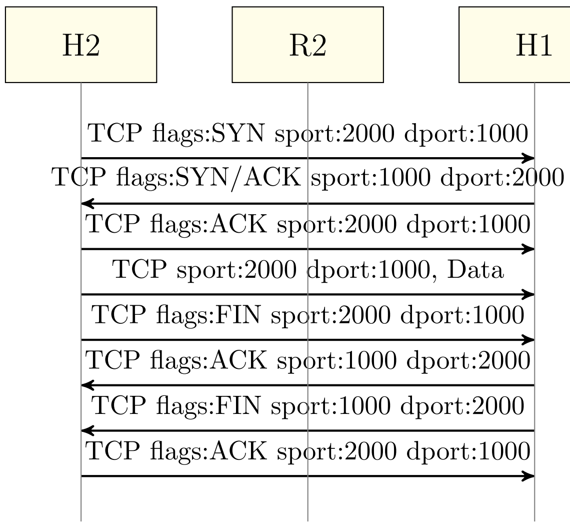
H2
R2
H1
TCP flags:SYN sport:2000 dport:1000
TCP flags:SYN/ACK sport:1000 dport:2000
TCP flags:ACK sport:2000 dport:1000
TCP sport:2000 dport:1000, Data
TCP flags:FIN sport:2000 dport:1000
TCP flags:ACK sport:1000 dport:2000
TCP flags:FIN sport:1000 dport:2000
TCP flags:ACK sport:2000 dport:1000

Figure B.69: Flow Diagram for the Scenario S.TG.04

Discussion

Without actually capturing the segments in front of H2 and H1 it is not possible to retrace the TCP stream. The client programme on H2 creates a socket to send the stream. This socket is shown in the Active Sockets (I.TT.03). The traffic filter R2 keeps track of the TCP session in the Traffic Filter Sessions (I.TT.05) and the host H1 opens first a listening socket and creates later an established socket with H2, both are shown in the Active Sockets. The timestamp when the socket changes from a listening socket to an established socket is when the TCP three-way handshake was completed. This should correlate with the Traffic Filter Session on R2 moving through the different TCP states.

This scenario is not fully retraceable without the Segment Payload (I.TT.02). However, it is not added to the set of significant information because it is likely that the TCP stream triggers events on H1 when it is passed to the next higher layer. This renders the information redundant.

Required Information I.TT.03, I.TT.05

Scenario Dependency SDG.01

B.5.5 Initiate a TCP Session to a Closed Port (S.TG.05)

Description

H2 initiates a TCP session with H1. Because H1 has no process listening on that port, it replies with an ICMP Destination Unreachable (Port Unreachable) message.

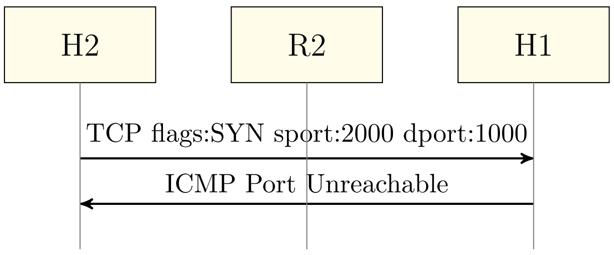

Figure B.70: Flow Diagram for the Scenario S.TG.05

Discussion

The Active Sockets (I.TT.03) show the socket used to establish the session in the state “SYN_SENT” and the Traffic Filter Session (I.TT.05) moves through the states “NEW (SYN_SENT)” and “DESTROYED”. H1 does not show any Active Sockets in this scenario. However, the ICMP Destination Unreachable message (I.NC.28) indicates that this scenario is in progress. Further, inspection of the payload of the ICMP message allows retracing the involved ports and the involved systems (IP addresses). The scenario S.NG.01 describes how to retrace the access ports based on the systems IP addresses.

Required Information I.TT.03, I.TT.05, I.NC.28

Scenario Dependency S.NG.01, S.NG.03

B.5.6 UDP Flooding Attack (S.TM.01)

Description

M2 sends UDP datagrams from a random source port and the source IP address of H2 to H1 random ports of H1. Either H1 replies with an ICMP error message or if an application is listening on the chosen destination port, with a response from the application. In both cases, the return traffic is sent to H2.

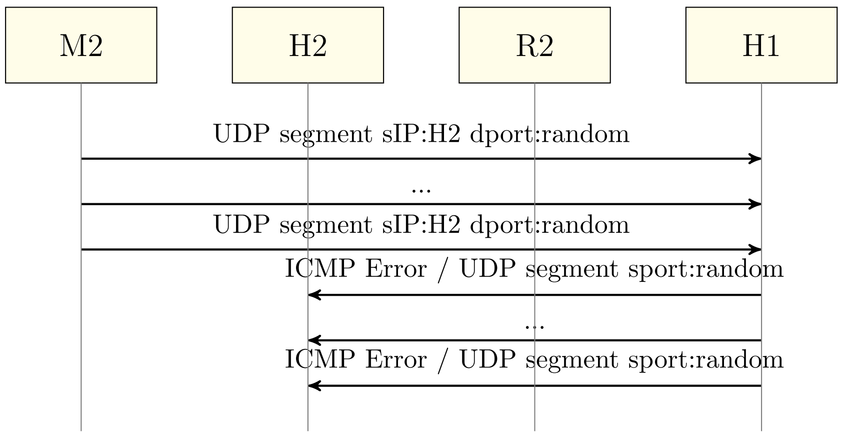
M2
H2
R2
H1
UDP segment sIP:H2 dport:random
...
UDP segment sIP:H2 dport:random
ICMP Error / UDP segment sport:random
...
ICMP Error / UDP segment sport:random

Figure B.71: Flow Diagram for the Scenario S.TM.01

Discussion

The attack can be retraced with the UDP sessions created on R2 (I.TU.05) and by the number of ICMP Destination Unreachable messages (I.NC.28) sent from H1 to H2. However, both pieces of information would indicate H1 as the source of the attack. To fully retrace the attack, IP address spoofing must be retraceable (see scenario S.NM.01).

Required Information I.TU.05, I.NC.28

Scenario Dependency S.NG.03, SNM.01

B.5.7 Endless Stream (S.TM.02)

Description

M2 send a spoofed UDP datagram (source port 7, source address H2) to initiate a connection to port 7 (echo service) of H1. H1 replies with the payload M2 added to the spoofed datagram. H2 replies to H1 with the same payload and so on. The two echo services continue to send each other the spoofed payload and create an endless UDP stream that uses all available network bandwidth. The attack can also use the Chargen service that just replies with 0 to 512 randomly selected characters.

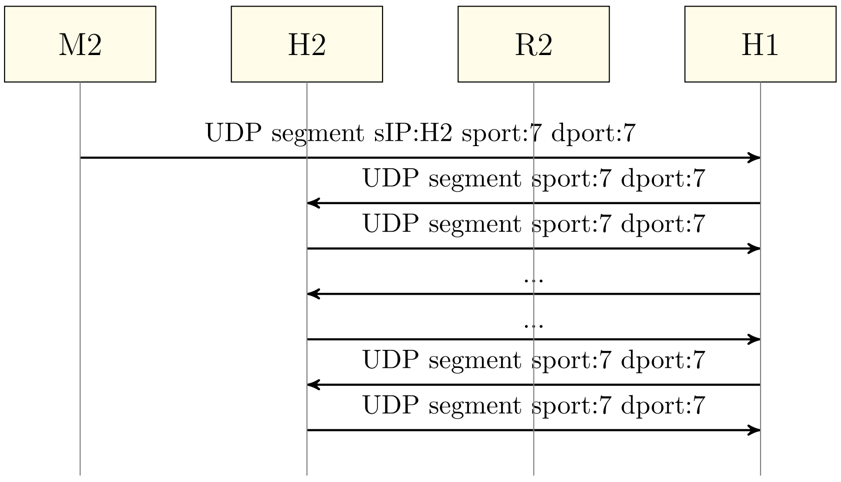
M2
H2
R2
H1
UDP segment sIP:H2 sport:7 dport:7
UDP segment sport:7 dport:7
UDP segment sport:7 dport:7
...
...
UDP segment sport:7 dport:7
UDP segment sport:7 dport:7

Figure B.72: Flow Diagram for the Scenario S.TM.02

Discussion

The attack is hard to detect as the initiating packet goes to a legitimate port with a service running on. However, the source port to be the same as the destination port is an indication that this scenario is in progress.

To retrace the attack the Datagram Header (I.TU.01) including the Packet Header (I.NI.01) is needed. The source IP address in the Packet Header indicates H2 as source of the attack. Therefore, to fully retrace the attack, IP address spoofing must be retraceable (see scenario S.NM.01).

Required Information I.TU.01, I.NI.01

Scenario Dependency S.NG.03, SNM.01

B.5.8 Crash UDP Parser (S.TM.03)

Description

M2 sends UDP datagrams to H2 with the length set to a smaller number than the minimum of eight [66] and with the checksum set to zero. The UDP checksum is mandatory for IPv6 [35].

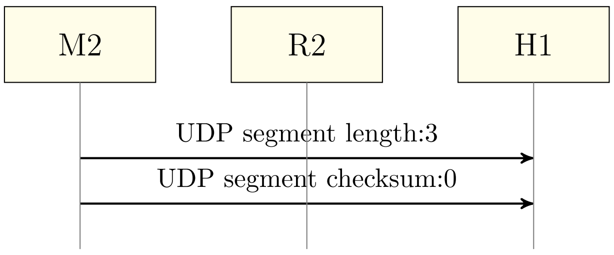

Figure B.73: Flow Diagram for the Scenario S.TM.03

Discussion

To retrace the attack the Datagram Header (I.TU.01) including the Packet Header (I.NI.01) is needed. Further, the Linux kernel logs both events in the kernel log (I.TU.08). While it records the IP addresses and ports of the connection for the short UDP packet, it does not do that for the UDP packet with the checksum set to zero. The scenario S.NG.01 describes how to retrace the access ports based on the systems IP addresses.

Required Information I.TU.01, I.TU.08, I.NI.01

Scenario Dependency S.NG.01, S.NG.03

B.5.9 Hide sending Process (S.TM.04)

Description

UDP allows using the source port 0 [66]. M2 sends a UDP datagram to H1 with the source port set to 0.

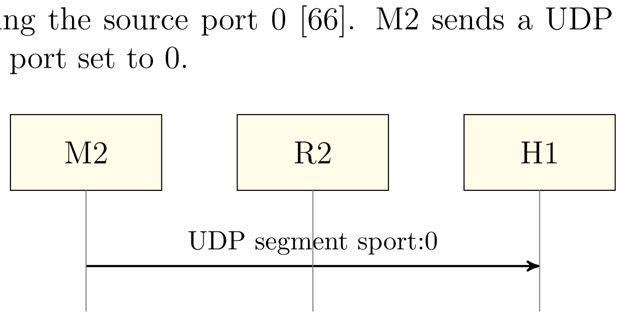

Figure B.74: Flow Diagram for the Scenario S.TM.04

Discussion

To retrace the attack the Datagram Header (I.TU.01) including the Packet Header (I.NI.01) is needed. The Datagram Header shows the malicious source port and the Packet Header allows determining the source port of the attack (see scenario S.NG.01). With a source port set to zero, it is not possible to identify the sending process on the source node.

Required Information I.TU.01, I.NI.01

Scenario Dependency S.NG.01, S.NG.03

B.5.10 Fingerprinting with Header Fields (S.TM.05)

Description

Different TCP implementations react differently to special combinations of header fields. For example, the combination of flags, the source or destination port is set to zero, which TCP options are active by default, the initial TCP window, and the initial sequence number.

M2 sends a TCP SYN segment to H1 with the destination port set to zero. Based on the respond of H1, M2 may be able do conclude the operating system H1 is running.

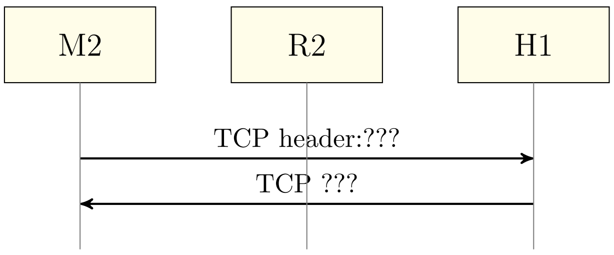

Figure B.75: Flow Diagram for the Scenario S.TM.05

Discussion

To retrace the attack the Segment Header (I.TT.01) including the Packet Header (I.NI.01) is needed. The Segment Header shows the malicious source port and the source IP address in the Packet Header allows retracing the packet to the access port where it entered the network (see scenario S.NG.01).

Required Information I.TT.01, I.NI.01

Scenario Dependency S.NG.01, S.NG.03

B.5.11 Fingerprinting by Retransmission Timeout Sampling (S.TM.06)

Description

Different TCP implementations use different retransmission timers. By measuring the time until the other side retransmits a segment, it is possible to do remote OS fingerprinting.

M2 establishes a TCP session with H1 and triggers a response of H1. Even though M2 receives the segments from H1 it does not acknowledge them. Instead, it records the time between the retransmissions. Based on these times, M2 tries to conclude what operating system is running on H1.

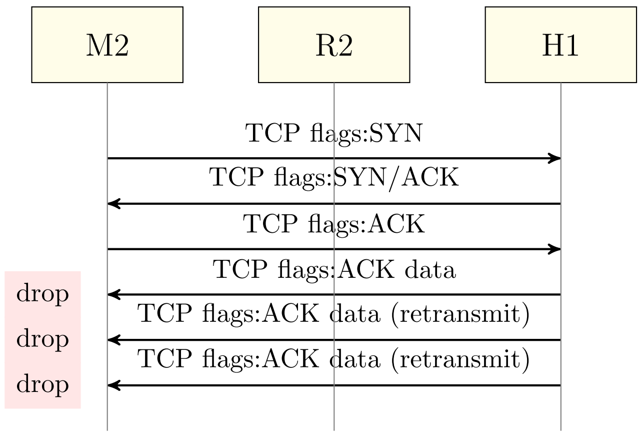

Figure B.76: Flow Diagram for the Scenario S.TM.06

Discussion

This attack is very hard to detect because it runs within the normal protocol behaviour. Exact the same packet exchange would happen if M2 would just lose the network connection instead of deliberately ignoring the responses it receives from H1. The malignity of the activity can only be shown if M2 repeats the process very often or does the same with many other systems.

Retracing the attack is possible with the Segment Header (I.TT.01) including the Packet Header (I.NI.01). The Segment Headers show the malicious activity and the source IP address in the Packet Header allows retracing the packet to the access port where it entered the network (see scenario S.NG.01).

Required Information I.TT.01, I.NI.01

Scenario Dependency S.NG.01, S.NG.03

B.5.12 DoS with Push Flag (S.TM.07)

Description

M2 establishes a TCP session with H1. Afterwards M2 keeps sending TCP segments with the Push Flag set but without any payload. This causes the receiving application on H1 to be triggered, which consumes memory and processor cycles.

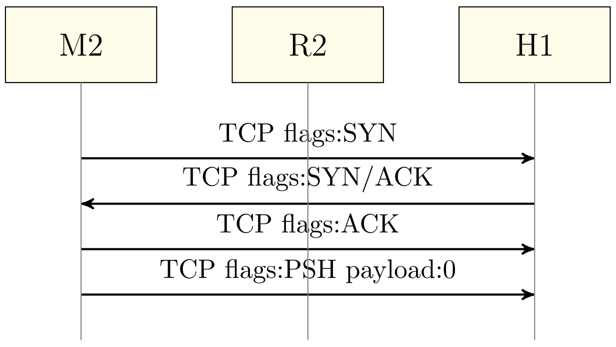

Figure B.77: Flow Diagram for the Scenario S.TM.07

Discussion

The connection tracking on R2 (I.TT.05) does not track the PSH flag. However, with the Segment Headers (I.TT.01) including the Packet Headers (I.NI.01) it is possible to retrace the scenario. The Segment Headers show the malicious activity and the source IP address in the Packet Header allows retracing the packet to the access port where it entered the network (see scenario S.NG.01).

Required Information I.TT.01, I.NI.01

Scenario Dependency S.NG.01, S.NG.03

B.5.13 SYN Flooding Attack (S.TM.08)

Description

M2 sends a TCP connection requests (SYN flag set) with spoofed source IP addresses to H1. H1 replies to with a TCP segment with the SYN and ACK flag set. The segment is sent to the spoofed IP address. If the address is used by a system, the system replies with an RST as it does not have a corresponding TCP session. If the address is unreachable, H1 waits until it runs into the timeout for the last segment of three-way handshake.

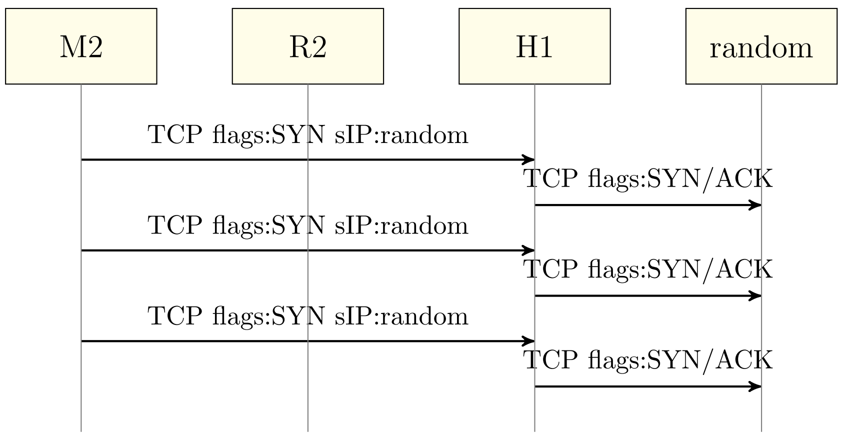

Figure B.78: Flow Diagram for the Scenario S.TM.08

Discussion

The attack can be detected with the large number of initiated but not established TCP sessions. To retrace the attack the Segment Header (I.TT.01) including the Packet Header (I.NI.01) is needed. The Segment Headers show the malicious activity and the source IP address in the Packet Header allows identifying the sender. It would indicate the random IP address as the source of the attack. Therefore, to fully retrace the attack, IP address spoofing must be retraceable (see scenario S.NM.01).

Required Information I.TT.01, I.NI.01

Scenario Dependency S.NG.03, SNM.01

B.5.14 Connection Forgery Attack (S.TM.09)

Description

H1 allows TCP session to be established only from H2. M2 disables H2, for example, with a SYN Flood Attack (S.NM.08), and establishes a TCP session with H1 with the source IP of H2. Without disabling H2 first, H2 would send RST segments in response to the TCP segments it receives from H1. M2 needs to guess the sequence numbers H1 is using (X in the flow diagram) as M2 does not receive any return traffic from H1 (H1 is blind).

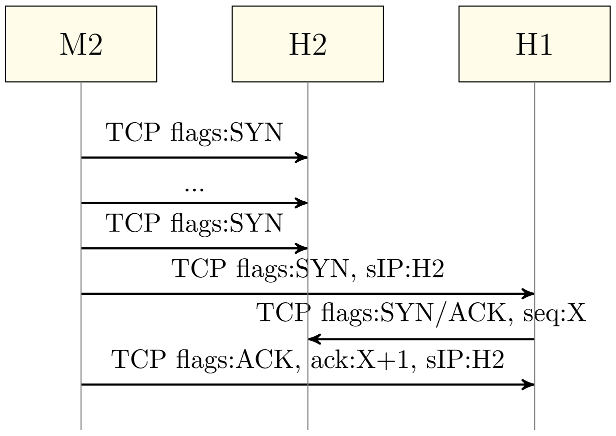

Figure B.79: Flow Diagram for the Scenario S.TM.09

Discussion

The scenario can be retrace with Segment Headers (I.TT.01) and the Packet Headers (I.NI.01). They show the first part of the attack, the SYN flooding, and the second part of the attack, the connection forgery. The source IP address of the second part implies H2 as the sender of the traffic. Therefore, to fully retrace the attack, IP address spoofing must be retraceable (see scenario S.NM.01).

Required Information I.TT.01, I.NI.01

Scenario Dependency S.NG.03, SNM.01, S.TM.08

B.5.15 Connection Flooding Attack “Naptha” (S.TM.10)

Description

M2 establishes an excessive number of TCP session with H1 just to abandon them after the handshake. M2 is not closing them with a FIN, nor with an RST segment. Each established session on H1 runs into the timeout eventually. Until then, the session consumes memory and processor cycles on H1.

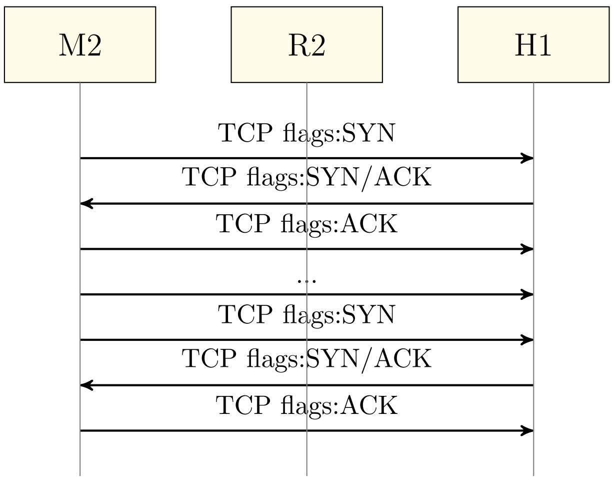

Figure B.80: Flow Diagram for the Scenario S.TM.10

Discussion

A spike in the number of established TCP sessions (I.TT.03) is an indication of this scenario. To retrace the attack the Segment Header (I.TT.01) including the Packet Header (I.NI.01) is needed. The Segment Headers show the establishing and abandoning of the session and the source IP address in the Packet Header allows retracing the packet to the access port where it entered the network (see scenario S.NG.01).

Required Information I.TT.01, I.TT.03, I.NI.01

Scenario Dependency S.NG.01, S.NG.03

B.5.16 Overlong TCP Option (S.TM.11)

Description

M2 sends a TCP segment to H1 with an unknown option that is longer than the actual TCP segment. The goal is to crash the TCP implementation of H1.

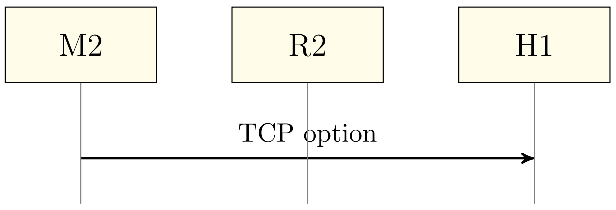

Figure B.81: Flow Diagram for the Scenario S.TM.11

Discussion

To retrace the attack the Segment Header (I.TT.01) including the Packet Header (I.NI.01) is needed. The Segment Headers show the malicious activity and the source IP address in the Packet Header allows retracing the packet to the access port where it entered the network (see scenario S.NG.01).

Required Information I.TT.01, I.NI.01

Scenario Dependency S.NG.01, S.NG.03

B.5.17 Blind In-Window Attack (Timestamp) (S.TM.12)

Description

M2 sends a spoofed TCP segment to H1 with the timestamp option set. M2 use H2's address as source IP and sets the timestamp option higher than what H2 recently used. Any segment H2 sends later has a smaller timestamp than the one M2 sent. Therefore, H1 drops them and return an ACK with the last correctly received segment [67], which is in this case the malicious one of M2. The TCP session freezes in the direction from H2 to H1.

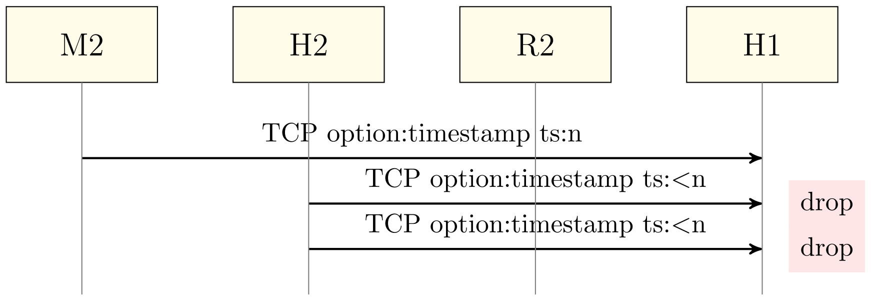
M2
H2
R2
H1
TCP option:timestamp ts:n
TCP option:timestamp ts:<n
drop
TCP option:timestamp ts:<n
drop

Figure B.82: Flow Diagram for the Scenario S.TM.12

Discussion

The scenario can be retrace with Segment Headers (I.TT.01) and the Packet Headers (I.NI.01). The Segment Headers have a jump in the timestamp (TSval) and segments after the jump segments use a lower timestamp value. The source IP address in the Packet Header implies H2 as the sender of the traffic. Therefore, to fully retrace the attack, IP address spoofing must be retraceable (see scenario S.NM.01).

Required Information I.TT.01, I.NI.01

Scenario Dependency S.NG.03, SNM.01

B.5.18 FIN-WAIT-2 Flooding Attack (S.TM.13)

Description

M2 establishes a TCP session with H1 and close the session directly afterwards with a FIN segment. H1 acknowledges the FIN segment and send one on its own to close the session in the other direction. However, M2 does not acknowledge the receiving of the FIN segment. H1 is stuck in the LAST_ACK state until it runs into the timeout.

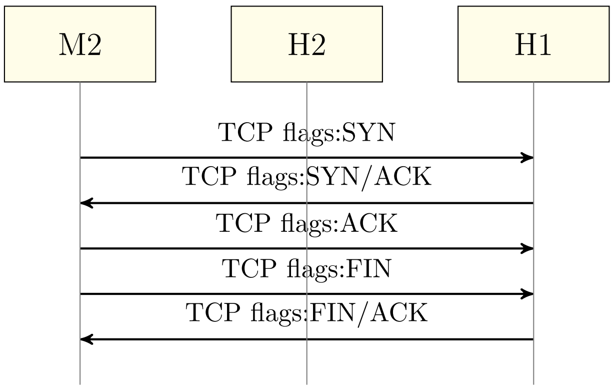

Figure B.83: Flow Diagram for the Scenario S.TM.13

Discussion

The scenario can be retrace with Segment Headers (I.TT.01) and the Packet Headers (I.NI.01). The Segment Headers show the malicious activity and the source IP address in the Packet Header allows retracing the packet to the access port where it entered the network (see scenario S.NG.01).

Required Information I.TT.01, I.NI.01

Scenario Dependency S.NG.01, S.NG.03

B.5.19 Resource Exhaustion Attack "Netkill" (S.TM.14)

Description

M2 establishes a TCP session with H1 and then brings H1 to send data (e.g. HTTP GET / HTTP/1.0). Then, the client abandons the connection and ignores any further packets related to this connection. Because the data is already in the TCP send buffer of H1, it keeps trying to deliver the data to M2 and run eventually into a timeout and gives up. Until then, system memory and processor cycles are consumed. M2 could further optimise the attack by advertising a zero-byte TCP window before abandoning the connection.

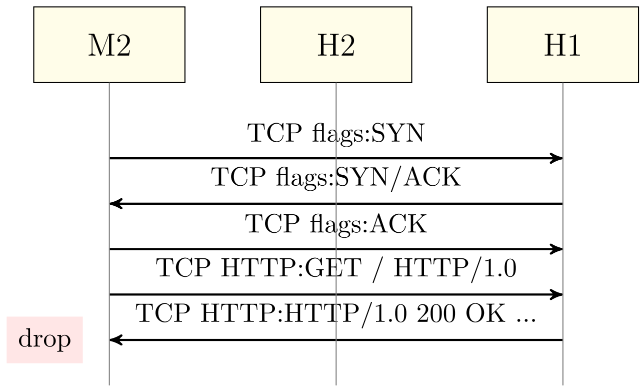

Figure B.84: Flow Diagram for the Scenario S.TM.14

Discussion

A spike in the number of established TCP session (I.TT.03) is an indication of this scenario. To retrace the attack the Segment Header (I.TT.01) including the Packet Header (I.NI.01) is needed. The Segment Headers show the malicious activity and the source IP address in the Packet Header allows retracing the packet to the access port where it entered the network (see scenario S.NG.01).

Required Information I.TT.01, I.TT.03, I.NI.01

Scenario Dependency S.NG.01, S.NG.03

B.5.20 TCP Reassembly Buffer Attack (S.TM.15)

Description

M2 sends a sequence of TCP segments to H1. M2 deliberately drops one of the segments and does not send it to H1. Afterwards M2 abandons the session and repeats the process in an excessive manner. H1 keeps the rest of the stream in the memory waiting for the missing segment to arrive or to timeout.

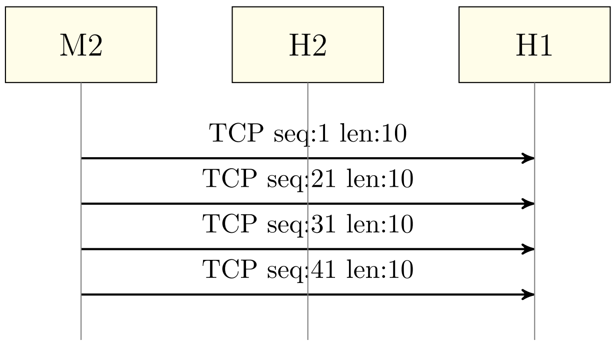

Figure B.85: Flow Diagram for the Scenario S.TM.15

Discussion

To retrace the attack all the Segment Headers (I.TT.01) including the Packet Headers (I.NI.01) are needed. The Segment Headers show the missing segment and the abandoning of the session. Further, the source IP address in the Packet Header allows retracing the packet to the access port where it entered the network (see scenario S.NG.01).

Required Information I.TT.01, I.NI.01

Scenario Dependency S.NG.03

B.5.21 Overlapping TCP segments (S.TM.16)

Description

M2 sends two TCP segments. The sequence number of the second one is smaller than the next expected sequence number. Therefore, the second segment overwrites data of the first segment.

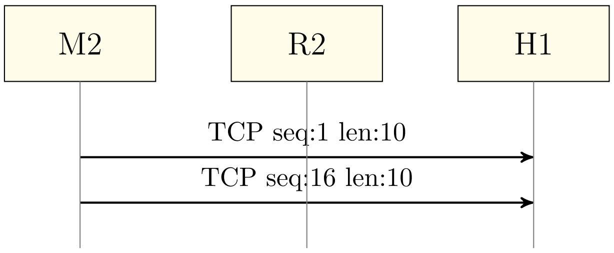

Figure B.86: Flow Diagram for the Scenario S.TM.16

Discussion

The scenario can be retrace with Segment Headers (I.TT.01) and the Packet Headers (I.NI.01). The Segment Headers show that the TCP sequence number of the second segment is smaller than the sequence number of the first segment plus its payload length. Further, the source IP address in the Packet Header allows retracing the packet to the access port where it entered the network (see scenario S.NG.01).

Required Information I.TT.01, I.NI.01

Scenario Dependency S.NG.01, S.NG.03

B.5.22 ACK Division Attack (S.TM.17)

Description

TCP increases the congestion window based on the number of ACKs received. To speed up the increasing of the congestion window, M2 sends multiple ACKs for one segment it receives.

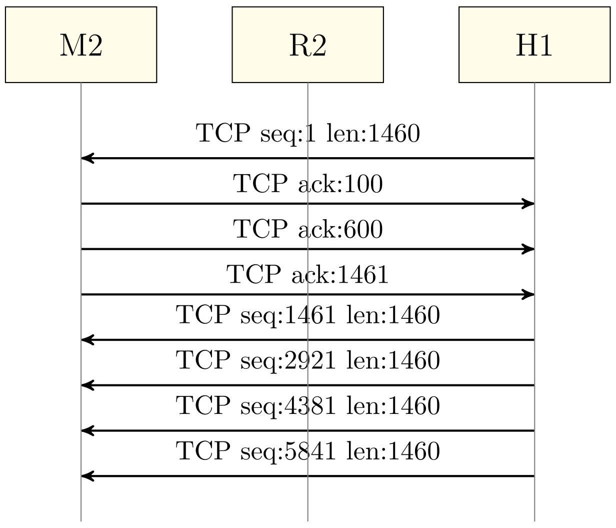

Figure B.87: Flow Diagram for the Scenario S.TM.17

Discussion

To retrace the attack all the Segment Headers (I.TT.01) including the Packet Headers (I.NI.01) are needed. The Segment Headers show how the ACK was divided and the source IP address in the Packet Header allows retracing the packet to the access port where it entered the network (see scenario S.NG.01).

Required Information I.TT.01, I.NI.01

Scenario Dependency S.NG.01, S.NG.03

B.5.23 Duplicated ACK Forgery (S.TM.18)

Description

TCP goes into Fast Recovery when three duplicated ACKs are received. To force a faster increasing of the congestion window, M2 sends multiple duplicated ACKs for one single segment it receives. The first three ACKs trigger Fast Recover and the following ACKs inflate the congestion windows of the sender. M2 ends up with an increased data transmission rate.

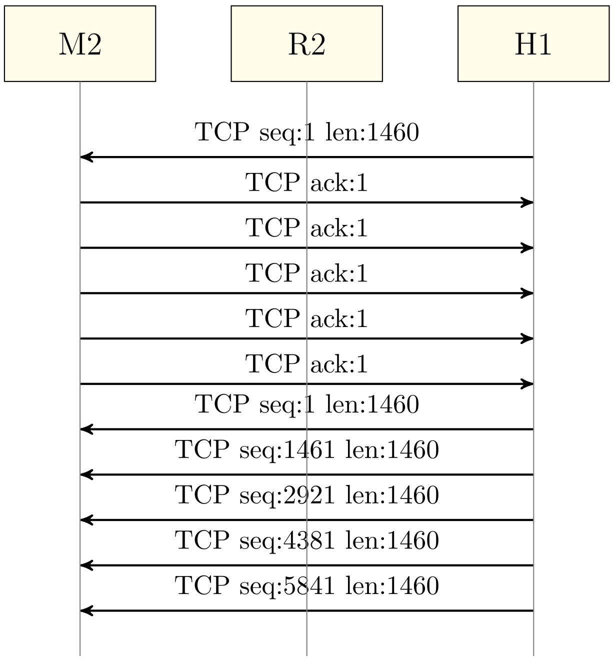

Figure B.88: Flow Diagram for the Scenario S.TM.18

Discussion

To retrace the attack all the Segment Headers (I.TT.01) including the Packet Headers (I.NI.01) are needed. The Segment Headers show how the ACK sent multiple times and the source IP address in the Packet Header allows retracing the packet to the access port where it entered the network (see scenario S.NG.01).

Required Information I.TT.01, I.NI.01

Scenario Dependency S.NG.01, S.NG.03

B.5.24 Optimistic ACKing (S.TM.19)

Description

M2 guesses what segments might by already be sent by H1 and in flight. M2 sends then the corresponding ACKs before actually receiving the corresponding segment.

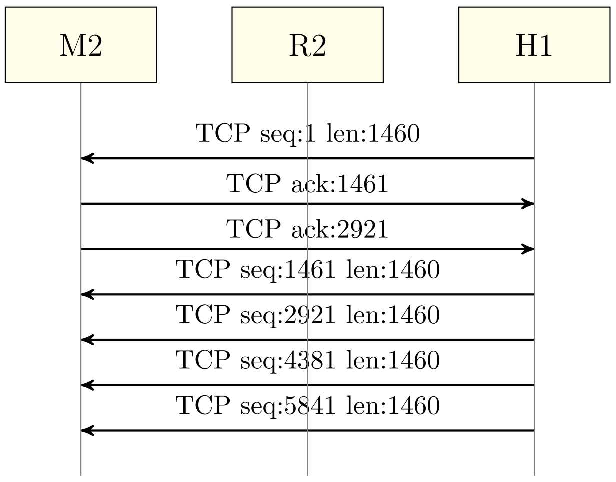

Figure B.89: Flow Diagram for the Scenario S.TM.19

The flow diagram is slightly misleading, it does not show how the different segments cross each other on the network.

Discussion

To retrace the attack all the Segment Headers (I.TT.01) including the Packet Headers (I.NI.01) are needed. The Segment Headers show how the ACK sent ahead of receiving the data and the source IP address in the Packet Header allows retracing the packet to the access port where it entered the network (see scenario S.NG.01). The success of retracing the scenario highly depends where exactly the network traffic is captured.

Required Information I.TT.01, I.NI.01

Scenario Dependency S.NG.01, S.NG.03

B.5.25 Blind Throughput-Reduction Attack (S.TM.20)

Description

When TCP receives segments that are not in the window of expected sequence numbers, it replies with an ACK. Therefore, it is possible for an attacker to trigger ACK messages by injecting segments in an established TCP session. The attacker has to guess the four tuple (source IP, source port, destination IP, destination port) of an existing connection.

M2 triggers four duplicated ACK messages from H1 to H2 by sending spoofed TCP segments that are out of the window of expected sequence numbers. H2, receiving three duplicated ACKs, retransmits the “lost” segment and goes into Fast Recovery.

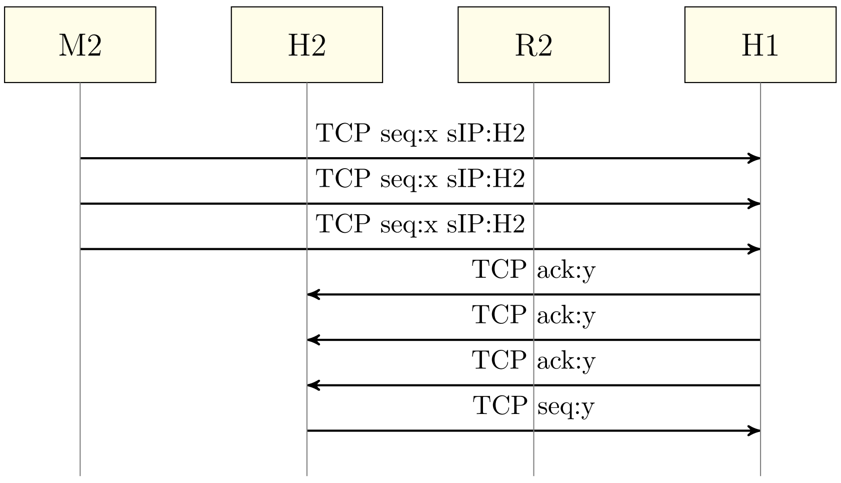

Figure B.90: Flow Diagram for the Scenario S.TM.20

Discussion

Retracing the attack is possible with the Segment Header (I.TT.01) including the Packet Header (I.NI.01). The sequence and acknowledgement numbers in the segments show the three duplicated ACKs sent from M2 to H1 and the ACKs sent from H1 to H2 in response to these. The source IP address in the Packet Header implies H2 as the sender of the traffic. Therefore, to fully retrace the attack, IP address spoofing must be retraceable (see scenario S.NM.01).

Required Information I.TT.01, I.NI.01

Scenario Dependency S.NG.03, SNM.01

B.5.26 Blind Flooding Attack (S.TM.21)

Description

When TCP receives segments that are not in the window of expected sequence numbers, it replies with an ACK. Therefore, it is possible for an attacker to trigger ACK messages by injecting segments in an established TCP session. The attacker has to guess the four tuple (source IP, source port, destination IP, destination port) of an existing connection.

M2 creates a long burst (instead of four in scenario S.TM.20) of duplicated ACK messages from H2 to H1 by sending spoofed TCP segments that are out of the window of expected sequence numbers. H1, receiving the duplicated ACKs, retransmits the “lost” segment and goes into Fast Recovery.

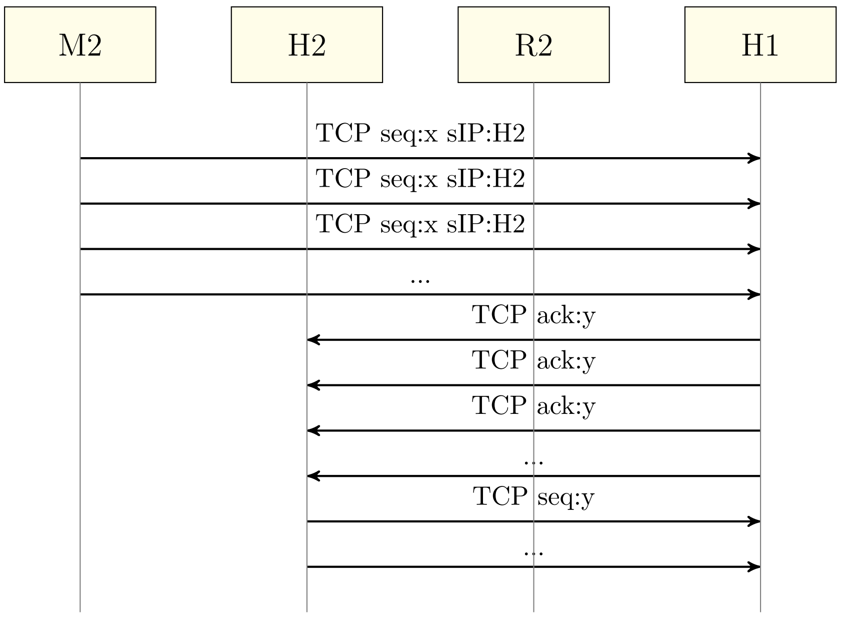

Figure B.91: Flow Diagram for the Scenario S.TM.21

Discussion

Retracing the attack is possible with the Segment Header (I.TT.01) including the Packet Header (I.NI.01). The sequence and acknowledgement numbers in the segments show the excessive number of duplicated ACKs sent from M2 to H1 and the ACKs sent from H1 to H2 in response to these. The source IP address in the Packet Header implies H2 as the sender of the traffic. Therefore, to fully retrace the attack, IP address spoofing must be retraceable (see scenario S.NM.01).

Required Information I.TT.01, I.NI.01

Scenario Dependency S.NG.03, SNM.01

B.5.27 Blind Connection Reset with RST Flag (S.TM.22)

Description

M2 resets the TCP connection between H1 and H2 with a spoofed TCP segment with the RST flag set. M2 has to guess the current sequence number and the four tuple (source IP, source port, destination IP, destination port) used in the TCP connection between H1 and H2 to successfully reset the session.

A variation of the attack does not send a segment with the RST flag but with the SYN flag set. If a SYN segment is received within the expected sequence number window, an RST segment should be sent in response [67]. That way, M2 triggers H1 to send an RST to H2.

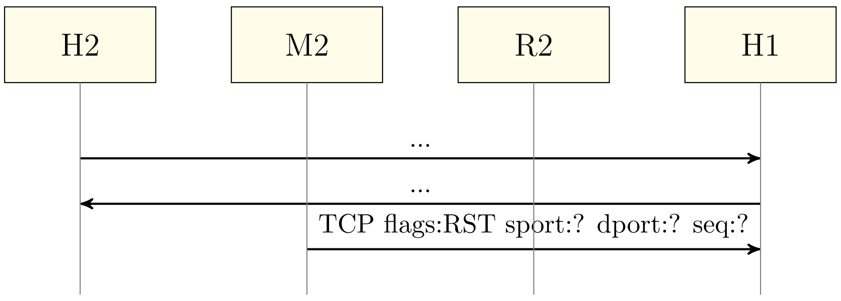

Figure B.92: Flow Diagram for the Scenario S.TM.22

Discussion

Retracing the attack is possible with the Segment Header (I.TT.01) including the Packet Header (I.NI.01). The RST flag in the Segment Header indicates the attack. The source IP address in the Packet Header implies H2 as the sender of the traffic. Therefore, to fully retrace the attack, IP address spoofing must be retraceable (see scenario S.NM.01).

Required Information I.TT.01, I.NI.01

Scenario Dependency S.NG.03, SNM.01

B.5.28 Blind Data-Injection Attack (S.TM.23)

Description

M2 inject data into the TCP session between H1 and H2 with a spoofed TCP segment. M2 has to guess the current sequence number and the four tuple (source IP, source port, destination IP, destination port) used in the TCP connection between H1 and H2.

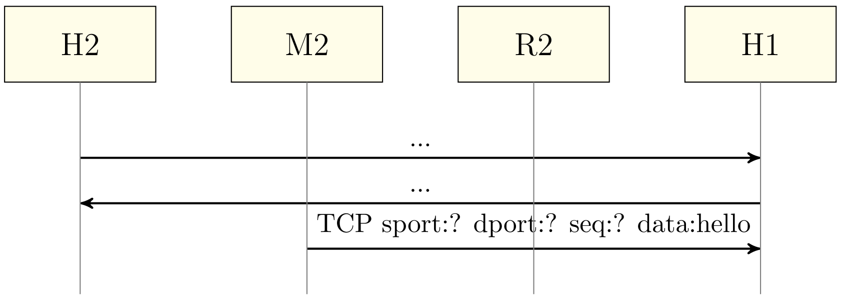

Figure B.93: Flow Diagram for the Scenario S.TM.23

Discussion

Depending on the size of the next segment sent by H2 after the attack, the attack triggers many retransmissions and the TCP session gets stale or the TCP session fully recovers from the attack.

The first case happens if the first segment after the attack sent by H2 to H1 is smaller than the injected segment. In that case, H1 replies to the segment with a duplicated ACK to acknowledge the injected segment. Simplified, H1 tells H2 that it already got the data. This confuses H1 because the internal sequence counter shows that the segment has not been transmitted yet. H1 continues to retransmit the segment in the timeout intervals and H2 keeps acknowledging the injected segment. H2 is not able to send any further data of the TCP session.

In the second case, the first segment after the attack is larger than the injected segment. In that case, H2 appends the received data to the data already received by the injected segment and replies with an ACK. After that exchange both, H1 and H2, have again corresponding sequence counters and the TCP session keeps intact.

Retracing the attack is possible with the Segment Header (I.TT.01) including the Packet Header (I.NI.01). The Segment Headers show that the TCP sequence numbers of all received TCP segments overlap in one part. The source IP in the Packet Header indicates H2 as source of the attack. Therefore, to fully retrace the attack, IP address spoofing must be retraceable (see scenario S.NM.01).

Required Information I.TT.01, I.NI.01

Scenario Dependency SDG.01

B.5.29 TCP Port Scan (S.TM.24)

Description

There exist roughly six methods to figure if a system has a TCP port in the listen state.

The first method uses a normal TCP three-way handshake while the second one sends an RST after the SYN/ACK from the server to avoid the logging of a newly establish connection on the server.

The third method is to use an unusual combination of TCP flags (e.g. FIN, NULL, or XMAS) scan.

The Maimon scan, the fourth method, uses a bug in old TCP implementations that silently drops a segment with the FIN and the ACK flag set instead of returning an RST. The fifth method uses a difference (non zero for RSTs of on open ports) in the size of the window field in the response to an out of sequence window received ACK. The final method is aimed to determine if an intermediate system filters the segment or the end system. An end node would reply to an ACK segment on an open port with an RST while a packet filter might just silently drop the packet.

M2 starts a TCP three-way handshake with H1 but sends an RST when it receives the SYN/ACK from H1.

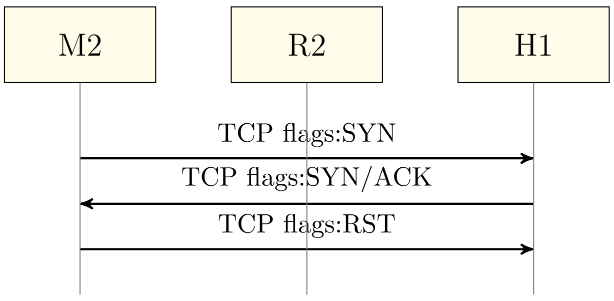

Figure B.94: Flow Diagram for the Scenario S.TM.24

Discussion

The first method, the three-way handshake, can be retraced with the Segment Headers (I.TT.01) and the Packet Headers (I.NI.01). However, it is almost not distinguishable from an genuine TCP session. A way to see if it was a genuine connection would be to retrace on the application layer if some meaningful data was exchanged.

The second method can be recognised base on that the reset of TCP session by M2 after receiving the SYN/ACK from H1. The third and fourth methods can be identified based on the TCP flag combinations. The fifth and sixth methods can be recognised based on the ACK segment not belonging to any active TCP session (I.TT.05).

For all methods, the destination port in the segments can be correlated with the Active Sockets I.TT.03 on H1 to determine if the segment was sent to an actual active socket or just used to determine open ports on the end system.

Required Information I.TT.01, I.TT.03, I.TT.05, I.NI.01

Scenario Dependency S.NG.03

B.5.30 Blind Performance-Degrading Attack with ICMP (S.TM.25)

Description

A TCP stream is sent as a sequence of IP packets. Ideally, each packet has the size of the Path MTU. In that case, every intermediate system is able to forward the packets. If a packet is bigger than the MTU to the next-hop, the system drops the packet and sends the sender an ICMP Packet Too Big message. The sender has to adjust the size of the IP packet to the MTU mentioned in the ICMP Packet Too Big message and resend the packet.

This attack is similar to the scenario S.NM.21 but it focuses on TCP. TCP makes the attack even harder as on top of the IP packet in the payload of ICMP message, the attacker needs also to guess the TCP ports and sequence numbers for a successful attack.

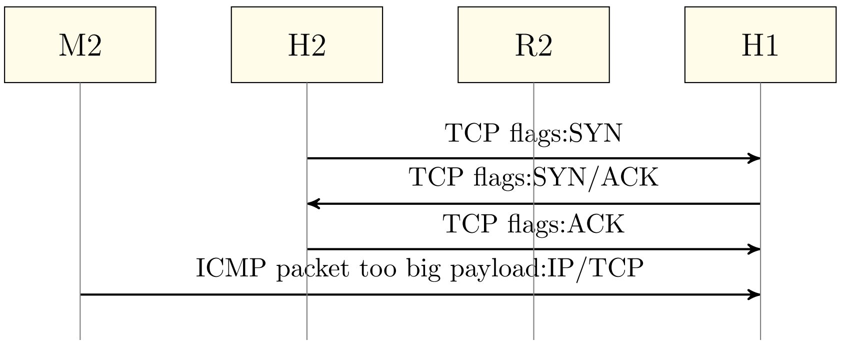

Figure B.95: Flow Diagram for the Scenario S.TM.25

Discussion

To retrace the attack the ICMP Packet Too Big message (I.NC.29) is needed. The payload of the ICMP message allows identifying the corresponding TCP session. The source IP address of the ICMP message allows retracing the access port (see scenario S.NG.03) of the attacker. Scenario S.NM.21 discusses how to verify if the ICMP message is genuine or malicious.

There is no need for the attacker to spoof the source IP address of the malicious ICMP message because it is not possible to determine if the source is actually a router on the path between H1 and H2 and a legitimate sender of an ICMP Packet Too Big message. However, an attacker might choose to spoof the source IP address anyway. In that case, to fully retrace the attack, IP address spoofing must be retraceable (see scenario S.NM.01).

Required Information I.NC.29, I.TT.01, I.TT.05, I.NI.01

Scenario Dependency S.NG.03, SNM.01, SNM.21

B.5.31 Blind Connection-Reset Attack with ICMP (S.TM.26)

Description

If a node receives an ICMP hard error (type 1 destination unreachable, code 1 communication with destination administratively prohibited or code 4 port unreachable) it aborts the corresponding connection.

M2 spoofs an ICMP hard error and sends it to H1 to reset an existing TCP session between H1 and H2. Depending on the TCP stack of H1, it either aborts the connection or just records the error [71]. M2 has to guess the four tuples (source IP, source port, destination IP, destination port) and the current sequence number to successfully reset the TCP session.

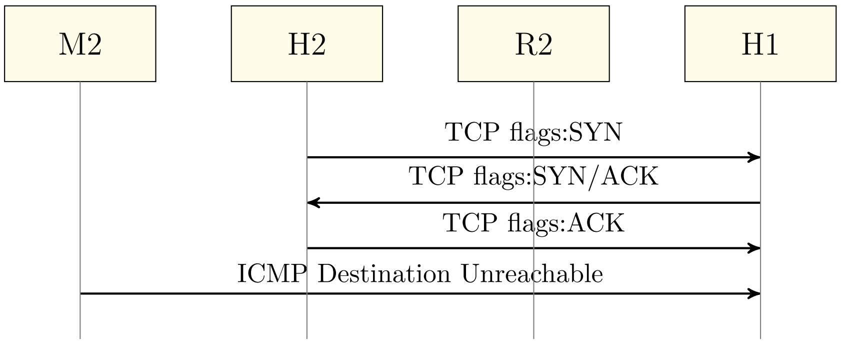

Figure B.96: Flow Diagram for the Scenario S.TM.26

Discussion

Retracing the attack is possible with the ICMP message (I.NC.28). The payload contains the TCP segment that triggered the error. The attack can be identified if the other side keeps sending segments (I.TT.01) after the ICMP message. In that case, it is obvious the there was no error. The IP address in the Packet Header (I.NI.01) of the ICMP message might identify the attacker (see scenario S.NG.01). On the other hand, the attacker might spoof the source IP of the message. Therefore, to fully retrace the attack, IP address spoofing must be retraceable (see scenario S.NM.01).

Required Information I.NC.28, I.TT.01, I.NI.01

Scenario Dependency S.NG.01, S.NG.03, SNM.01

B.5.32 TCP-based Traceroute (S.TM.27)

Description

Traceroute uses typically ICMP Echo Request or UDP packets with growing Hop Limit values to retrace the path packets take through the network. In some cases, the stateless nature or a filter device prevents the concept from working properly. Therefore, an attacker establishes a TCP session with the other node and starts sending TCP segments with increasing Hop Limit values. This scenario is similar to scenario S.NG.06.

M2 establishes a TCP session with H1. Afterwards, M2 sends further TCP segments but starts with a Hop Limit of one and increments it every TCP segment by one. Based on the ICMP Time Exceeded messages, M2 reconstructs the path the segments took through the network.

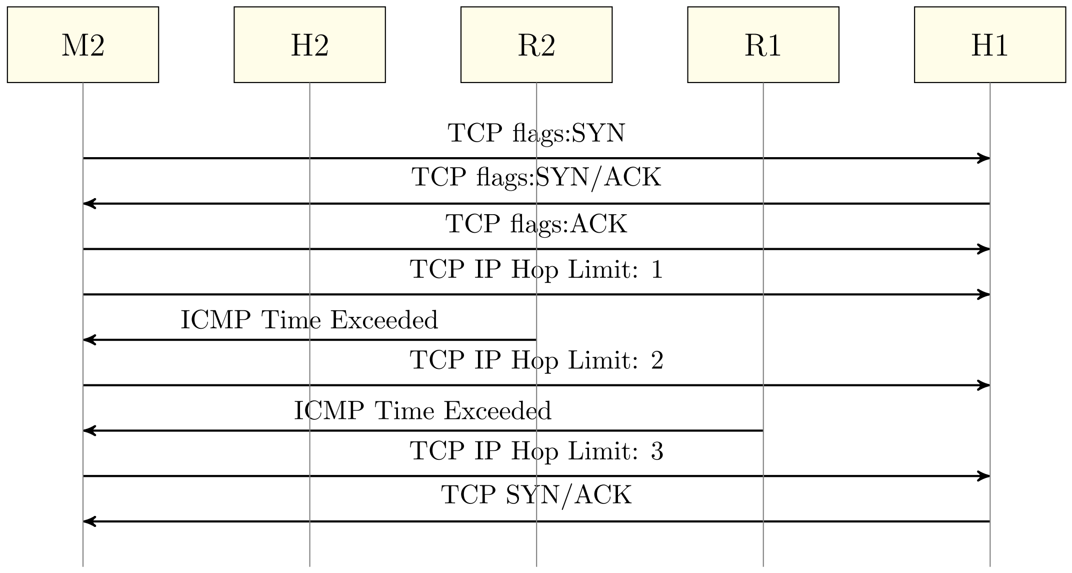

Figure B.97: Flow Diagram for the Scenario S.TM.27

Discussion

To retrace the attack all the Segment Headers (I.TT.01) including the Packet Headers (I.NI.01) and the ICMP Time Exceeded messages (I.NC.30) are needed. The Segment Headers with the increasing Hop Limit values and the corresponding ICMP messages show the attack. The source IP address in the Packet Header allows retracing the packet to the access port where it entered the network (see scenario S.NG.01).

Required Information I.NC.30, I.TT.01, I.NI.01

Scenario Dependency S.NG.03, S.NG.06

B.6 Application Layer Scenarios

B.6.1 Resolve a Name from a Local Zone (S.AG.01)

Description

H2 sends a DNS query to I1 to resolve the IP address of *H3.lab*. I1 is the authoritative name server for the zone. I1 receives the query, looks up the hostname and replies with the IP address of H1.

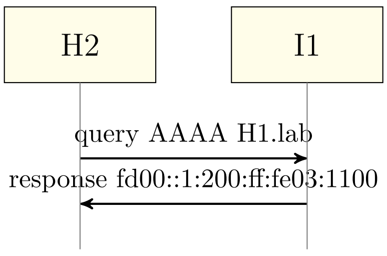

Figure B.98: Flow Diagram for the Scenario S.AG.01

Discussion

To retrace a DNS client that resolves a hostname that is in the local zone, the DNS Server (I.AN.04), the DNS Server Log (I.AN.07), and the DNS Cache (I.AN.06) is needed. The DNS Server identifies the system that handles the query and allows determining how the query was handled based on the server's configuration. The DNS Server Log allows determining the exact timestamp when the query was received, the IP address of the sender of the query and the content of the query, whether the Recursion Desired flag was set and on what IP address the query was received on. The answer, the DNS server replied with, is not shown in the log. The DNS Cache allows filling this blank. If the resource record is in the cache when the client sends the query, that is the answer the client received. If the resource record is in the cache after the query is received, that is the answer the client received. If the resource record is not in the cache after the query was received, the client received a reply without an answer (not found). The scenario S.NG.01 describes how to retrace the access ports based on the systems IP addresses. The actually data exchanged between the client and server (I.AN.01) would simplify the retracing and make the evidence more reliable but it is not needed to retrace the scenario.

Required Information I.AN.04, I.AN.06, I.AN.07

Scenario Dependency STG.01, S.NG.01

B.6.2 Resolve a Name from a Remote Zone (S.AG.02)

Description

H2 sends a DNS query to I1 to resolve the IP address of *ipv6.google.com.* I1 is not the authoritative name server for the zone. I1 receives the query from H2 and decides based on the hostname that it is not the authoritative name server. Because the DNS server is setup as a forwarder, it sends the query to D, a DNS server that can resolve the query outside of the lab network. D replies with a CNAME and the IP address of the CNAME. However, I1 sends another query for www.l.google.com, the CNAME previously replied, and D replies with the IP address of the host. I1 forwards then the CNAME and the IP address back to H2.

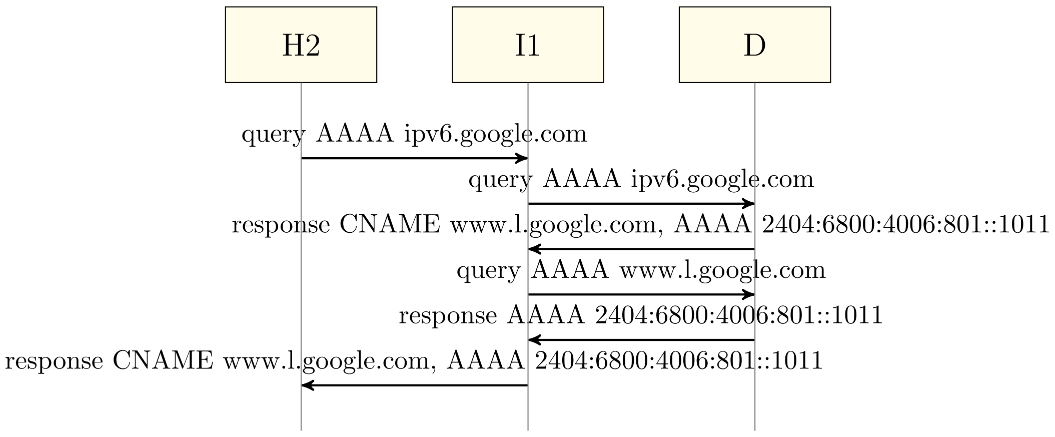
H2
I1
D
query AAAA ipv6.google.com
query AAAA ipv6.google.com
response CNAME www.l.google.com, AAAA 2404:6800:4006:801::1011
query AAAA www.l.google.com
response AAAA 2404:6800:4006:801::1011
response CNAME www.l.google.com, AAAA 2404:6800:4006:801::1011

Figure B.99: Flow Diagram for the Scenario S.AG.02

Discussion

This scenario is almost identical to scenario S.AG.01, however, this time the client sends a request to resolve a hostname in a non-local zone (e.g. ipv6.google.com). The retracing is also almost identical but to retrace how I1 resolved the name the actual DNS traffic (I.AN.01) is required. Determining the access port based on a system's IP address is discussed in scenario S.NG.01.

Required Information I.AN.01, I.AN.04, I.AN.06, I.AN.07

Scenario Dependency STG.01

B.6.3 Request Zone Transfer (S.AG.03)

Description

H2 establishes a TCP session with I1 and requests a zone transfer for the zone *lab*. I1 receives the request and replies with all resource records in the zone *lab*. If the host H2 is not supposed to initiate a zone transfer, this scenario is a reconnaissance attack. For the discussion of this scenario, H2 is allowed to initiate zone transfers.

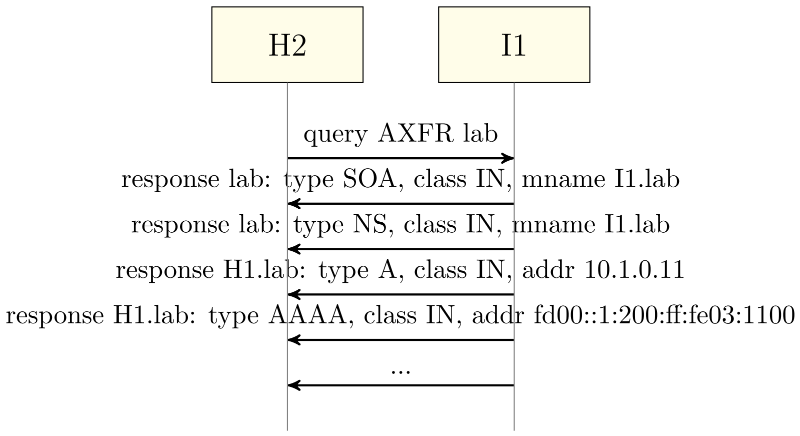
H2
I1
query AXFR lab
response lab: type SOA, class IN, mname I1.lab
response lab: type NS, class IN, mname I1.lab
response H1.lab: type A, class IN, addr 10.1.0.11
response H1.lab: type AAAA, class IN, addr fd00::1:200:ff:fe03:1100
...

Figure B.100: Flow Diagram for the Scenario S.AG.03

Discussion

The DNS Server Log (I.AN.07) allows retracing the timestamp, the client's IP and port, the zone the client requested and the IP the client connected to. The configuration of the DNS server (I.AN.04) allows determining the content of the zone the client received. On top of that, the actual DNS Traffic (I.AN.01) can be used as a secondary source.

Required Information I.AN.01, I.AN.04, I.AN.07

Scenario Dependency S.NG.01 , STG.04

B.6.4 DHCP Information Request (S.AG.04)

Description

H2 requests the address of the recursive DNS server and the Domain Search List. I1 receives the request and replies with appropriate parameter. H2 sends the request to the all DHCP server multicast group (ff02::1:2), the reply is directly addressed to the link-local address of H2.

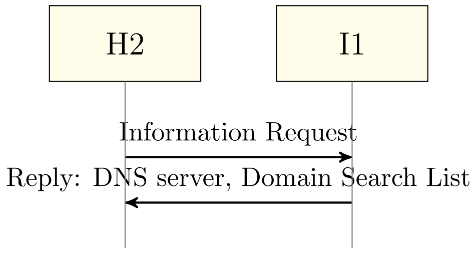

Figure B.101: Flow Diagram for the Scenario S.AG.04

Discussion

To retrace a client requesting stateless information from a DHCP server, the DHCP Server Log (I.AH.11) and the DHCP Server Configuration (I.AH.04) are very useful. The DHCP Server Log shows the messages received including a timestamp, the source IP and port where they were sent from, and the transaction ID. The log also contains the messages the server sends. However, the log does not contain the content of the message that was sent to the client. This needs to be reconstructed from the configuration of the DHCP server. The actual DHCP Traffic (I.AH.01) can be used as a secondary source but is not needed to retrace the scenario. The scenario S.NG.01 describes how to retrace the access ports based on the systems IP addresses.

The scenario can be slightly modified to include a DHCP Relay Agent (I.AH.05). In this case, the DHCP client runs on H1 and the DHCP Relay Agent on R1. The retracing is almost identical as before but the DHCP Relay Agent also creates log messages. The log messages contain the IP address and the interface the request was received, the server the message was relayed to and the receiving and forwarding of the server's reply. The content of the reply message is not in the log. Therefore, the DHCP Server Configuration is still needed to retrace the scenario.

Required Information I.AH.04, I.AH.05, I.AH.07, I.AH.11, I.AH.12

Scenario Dependency STG.01 , S.NG.01

B.6.5 DHCP Address Request (S.AG.05)

Description

H2 requests an IP address and the address of the recursive DNS server and the Domain Search List. I1 receives the request and replies with appropriate parameter. H2 searches for all available DHCP server with a Solicit message. The available DHCP servers reply with Advertise messages. H2 choses the best reply and requests the address offered in the Advertise message. The server sends with a Reply message to allow H2 using the address. When the address starts to timeout, H2 sends a Renew message to keep the address longer. I1 replies with a Reply message if it agrees with the renewal.

H2 sends all messages to the all DHCP server multicast group (ff02::1:2), the replies are directly address to the link-local address of H2.

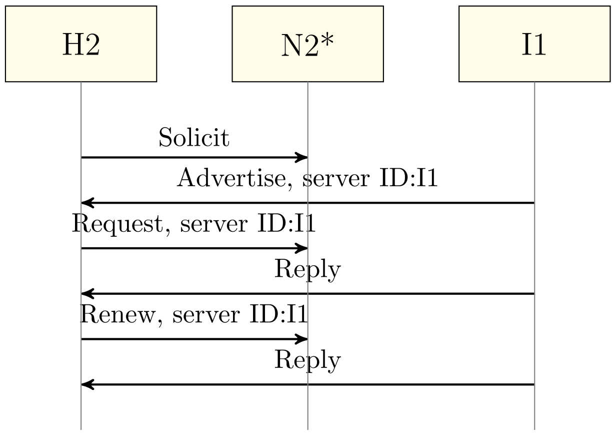

Figure B.102: Flow Diagram for the Scenario S.AG.05

N2* means that the packet is sent as a multicast packet.

Discussion

The four messages of a DHCP address assignment can be retraced with the DHCP Server Log (I.AH.11). It contains the timestamps and the kind of message that was received or sent and the IP and port the message was received from or sent to. The transaction ID of the received messages is also in the log. The first time the server assigns an IP address to the client, the address appears in the log. Unfortunately, the message stating the server chosen a certain IP address to assign to a client can only by matched to a clients Solicit message by time proximity. There is no other way to say what IP address was assigned to a specific client with a certain link-local address or client ID. The Binding Database (I.AH.07) can be used do determine what client ID had what IP address or addresses assigned at a specific point in time. Again, it is only possible to match the address in the Binding Database with the log entries about messages sent and received from a specific link-local address by time proximity.

An additional clue might be that the IP link-local address and the client ID (I.AH.08) are both often generated based on the data-link layer address of the client. Therefore, it is likely that the log entries with the link-local addresses can be correlated with the addresses in the Binding Database based on the data-link layer address that was used to generate both. However, this is not a necessity of the protocol but merely a coincidence.

Retracing of this setup with a DHCP Relay Agent is almost identical. The only difference is that the relay agent also log the receiving and forwarding of the DHCP message. The log is not detailed enough to retrace the scenario without the DHCP Traffic.

Because of the relative weak link between the Server Log and the Binding Database the DHCP Traffic (I.AH.01) is needed to retrace this scenario.

Required Information I.AH.01, I.AH.04, I.AH.05, I.AH.07, I.AH.08, I.AH.11, I.AH.12

Scenario Dependency STG.01

B.6.6 DNS Reply Spoofing (S.AM.01)

Description

M2 spoofs DNS replies for requests that H2 sends. The attack is quite difficult because M2 has to know the source port M2 and the transaction ID M2 uses. Further, M2 needs to know the name, type and class of the resource record H2 is asking for. On top of that, the reply has to receive the victim before the reply of the genuine DNS server does. This attack is likely to be possible if the attacker is in a position to capture traffic from the victim but not to intercept it. The attacker might start a DoS attack towards the server beforehand to increase the change of delivering the spoofed reply before the genuine one. A special variation of the attack is to reply with a Name Error instead of an IP address. In that case, the attack changes to a Dos attack.

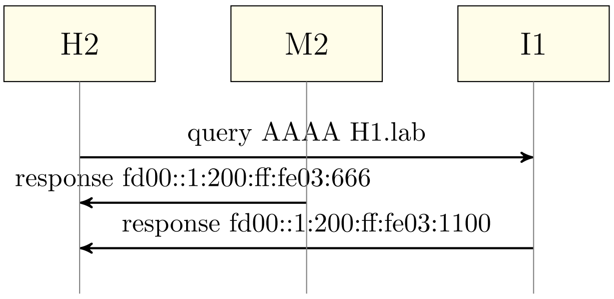
H2
M2
I1
query AAAA H1.lab
response fd00::1:200:ff:fe03:666
response fd00::1:200:ff:fe03:1100

Figure B.103: Flow Diagram for the Scenario S.AM.01

Discussion

To retrace the scenario, the actual DNS Traffic (I.AN.01) is helpful. The attack is based on IP spoofing. Therefore, to detect and retrace the attack IP address spoofing needs to be detectable and retraceable (see scenario S.NM.01). Further, the actual DNS traffic shows the double response (from the attacker and the genuine server) and the resource records the attacker spoofed.

Required Information I.AN.01

Scenario Dependency STG.01, SNM.01

B.6.7 DNS Cache Poisoning (S.AM.02)

Description

Cache poisoning works by appending malicious resource records to genuine DNS replies. The attacker tricks the victim to resolve a hostname in a zone under the control of the attacker (e.g. by a link in an email). That means the victim resolves a hostname from a DNS server under the control of the attacker. When the victim resolve the hostname in the link, for example *www.unsuspicious.com*, the attacker replies with the IP address of the server but also with an additional, malicious resource record that states, for example that *ipv6.google.com* resolves to an IP address under the attacker's control.

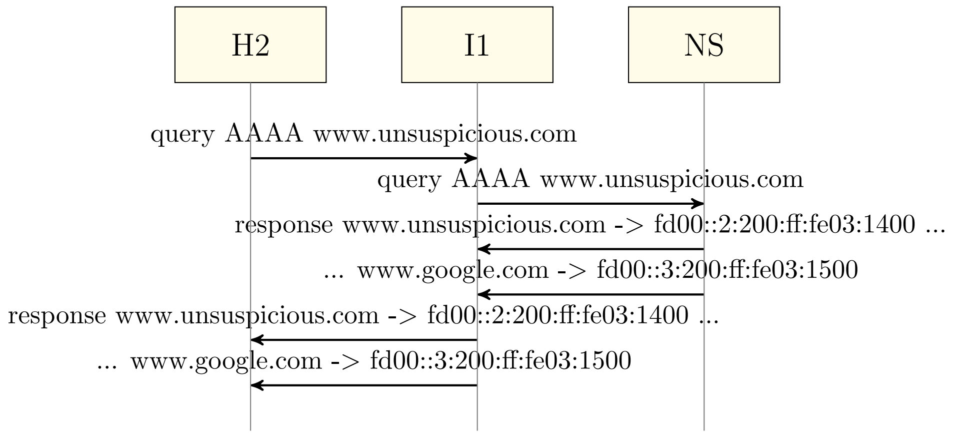
H2
I1
NS
query AAAA www.unsuspicious.com
query AAAA www.unsuspicious.com
response www.unsuspicious.com -> fd00::2:200:ff:fe03:1400 ...
... www.google.com -> fd00::3:200:ff:fe03:1500
response www.unsuspicious.com -> fd00::2:200:ff:fe03:1400 ...
... www.google.com -> fd00::3:200:ff:fe03:1500

Figure B.104: Flow Diagram for the Scenario S.AM.02

NS is a DNS server outside the lab in the Internet.

Discussion

DNS Cache Poising is particularly hard to detect and retrace. The DNS queries sent by the DNS recursive resolver have to be matched against the replies received from the different name servers. If a reply contains resource records that were not asked for in the query, it is possible that this scenario is in progress. However, related replies such as an additional A record for the CNAME record the original query resolved to are fine and not necessarily malicious.

Retracing the scenario is possible with the DNS Traffic (I.AN.01). The sender of the malicious replies can be identified by its IP address but the actual access port is not possible as the server is outside of the lab network in the Internet.

Required Information I.AN.01

Scenario Dependency STG.01

B.6.8 DNS Reflector Attack (S.AM.03)

Description

The DNS protocol can be misused for DoS attacks. It works particularly well because a small query can create a huge reply. The reflector attack uses this amplification vector to create a DoS attack against a DNS client.

M2 creates DNS queries with the intent to receive large answers (e.g. TXT). M2 puts multiple of these queries into one request and sends it to a recursive DNS server. M2 spoofs the source address in the query with H2's IP address. H2 receives the replies for the query M2 sent. With the second spoofed query, the recursive name server does not even need to resolve the names but directly replies from the cache. This generates even more traffic towards H2.

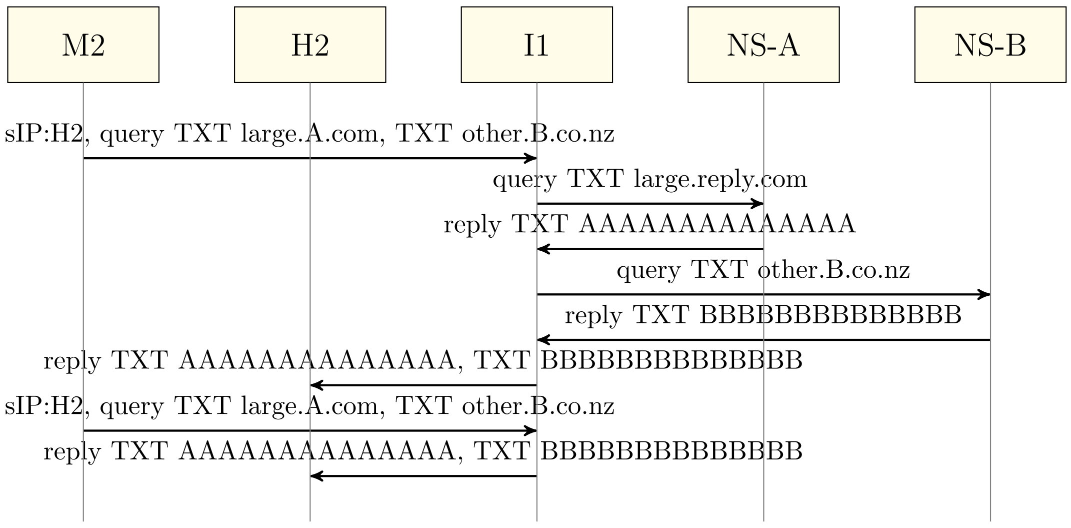
M2
H2
I1
NS-A
NS-B
sIP:H2, query TXT large.A.com, TXT other.B.co.nz
query TXT large.reply.com
reply TXT AAAAAAAAAAAAAA
query TXT other.B.co.nz
reply TXT BBBBBBBBBBBBBB
reply TXT AAAAAAAAAAAAAAA, TXT BBBBBBBBBBBBBB
sIP:H2, query TXT large.A.com, TXT other.B.co.nz
reply TXT AAAAAAAAAAAAAAA, TXT BBBBBBBBBBBBBB

Figure B.105: Flow Diagram for the Scenario S.AM.03

NS-A and NS-B are DNS servers outside the lab in the Internet.

Discussion

A DNS Reflector Attack can be detected and retrace with the DNS Traffic (I.AN.01). A query with a group of highly unrelated resource records and that result in large replies are typical for the attack. The attack is only successful if the attacker is able to spoof the IP address of the DNS query that starts the attack. Therefore, it must also be possible to detect and retrace IP address spoofing (see scenario S.NM.01). Only then, it is possible to retrace this attack.

Required Information I.AN.01

Scenario Dependency STG.01, SNM.01

B.6.9 Rogue DHCP Server (S.AM.04)

Description

DHCP clients send the Solicit message to a multicast address. M2 runs a DHCP server and replies to the Solicit message with an Advertise message. If M2 advertises with the preference of 255 (highest possible), it is likely that the client M2 uses this configuration. If the RA messages in the network do not have the M flag set (no managed address configuration), the attacker can decide to send his/her own RA messages (see scenario S.NM.15) to get the nodes in the networks to start using DHCP for the address configuration. The attacker can then decide to suggest stateless (just the O flag) or stateful (M flag) configuration for the clients.

Once the victim uses the attacker's DHCP server, the attacker can send the client configuration information at will. For example, the attacker could send the client an extremely long Domain Name Search List. This leads to delays on the client when it tries to resolve a non-fully qualified domain name. If the attacker does that to enough clients, it could be considered a DoS attack against the DNS server [94]. Another malicious option is to send the client very large DHCP messages or reply with an Advertise message for an address already in use.

A variation of the attack is to run a DHCP server blindly if the attacker is not in a position to receive the DHCP Solicit messages from the client. In that case, that attacker needs to guess the transaction ID [93] and the source port used by the client. This makes the attack much more difficult.

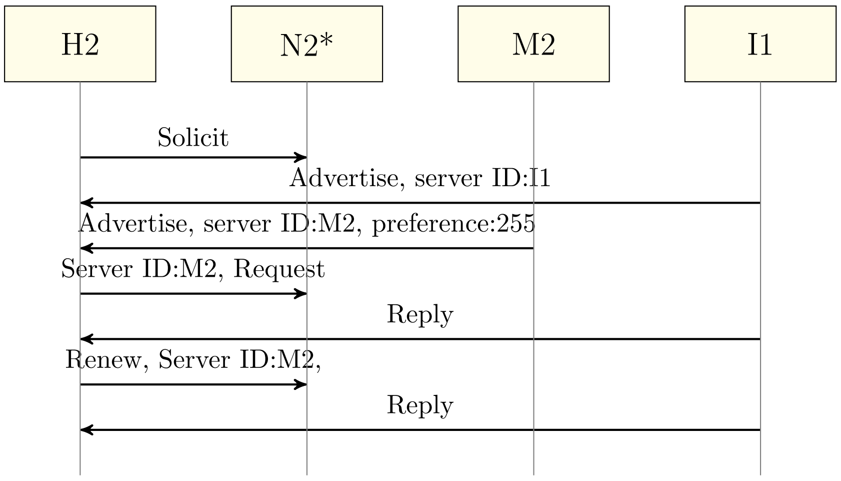

Figure B.106: Flow Diagram for the Scenario S.AM.04

Discussion

A rogue DHCP server can be detected and retrace with the DHCP Traffic (I.AH.01). The access port of the rogue DHCP server can be retraced with the source IP address (link-local) the server uses to send its messages. The scenario S.NG.01 describes how to retrace the access ports based on the systems IP addresses. If the attacker spoofs the source address and the capture DHCP traffic contains the Frame Header (I.DE.01), the access port can be determined by the Filter Database (I.DB.05).

Genuine DHCP servers are also in a good position to detect the attack. This is because the DHCP client sends all messages (Solicit, Request) to the All-DHCP-Server multicast group. That means that all DHCP servers receive the Request message the clients sends to the rough DHCP server. If the genuine DHCP server holds a list of the server IDs of all genuine DHCP servers in the network, it could create at least a log message stating a Request message for an unknown DHCP server was received. When the attacker decides to spoof the DHCP server ID with one of a genuine DHCP server, at least this server is able to detect that the Request message the client sends does not match any Advertise message previously sent by the server. However, this only allows retracing the attack if the attacker uses its own link-local IP address.

Required Information I.AH.01, I.AH.08, I.DE.01, I.DB.05

Scenario Dependency STG.01, S.NG.01

B.6.10 Spoof DHCP Release Message (S.AM.05)

Description

A DHCP client releases the IP address it requested previously from the DHCP server. An attacker could spoof such a release message. In the worst case, the server assigns the address to a new client at a later state while the other client is still using the address. In that case, the new client is unable to use the address because DAD fails.

M2 spoofs a DHCP Release message and sends it to I1. I1 marks the address as released and reassigns it to a new client at a later stage. M2 does not need to spoof the link-local address for a successful attack.

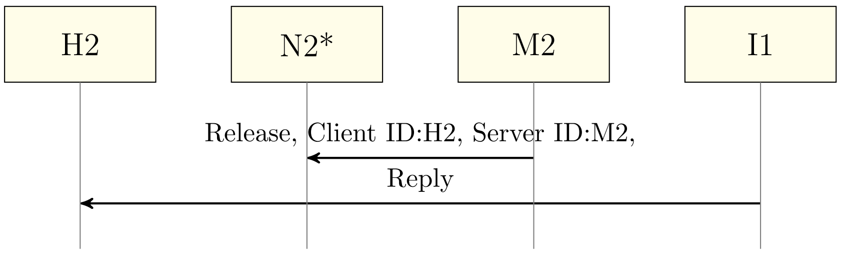

Figure B.107: Flow Diagram for the Scenario S.AM.05

Discussion

To retrace a spoofed DHCP Release message, the actual DHCP Traffic (I.AH.01) is needed. The attack can be detected by comparing the link-local IP address of the sender with the client ID (I.AH.08) in the actual message. If it changes during the lifetime of the DHCP lease, this scenario is ongoing. When the attacker decides to spoof the sender's IP address too, the retracing depends on the ability to detect and retrace IP address spoofing (see scenario S.NM.01).

Required Information I.AH.01, I.AH.08

Scenario Dependency STG.01, SNM.01

B.6.11 DoS DHCP Server (S.AM.06)

Description

A DHCP server uses a pool of addresses that are assigned to clients asking for one. An attacker could keep asking for new addresses until the pool is depleted. The attacker might need to spoof the link-local source address to trick the server into assigning a new IP address for every new request. Depending on the server implementation, changing the client ID might suffice. With access networks recommended to have /64 subnet masks [122] the attack can be made much harder by allowing the server to assign an relatively large pool.

M2 keeps sending DHCP Solicit message, followed by a Request message once the Advertise message from the server is received. M2 changes the client identifier for each request.

If the attacker is not in the same data-link layer as the DHCP server or in a network with a DHCP relay agent, the attacker could spoof a DHCP relay agent message. To do this, the attacker needs to know the IP address of a valid DHCP relay agent. The advantage of the attack is that the attacker does not receive the replies for the DHCP server.

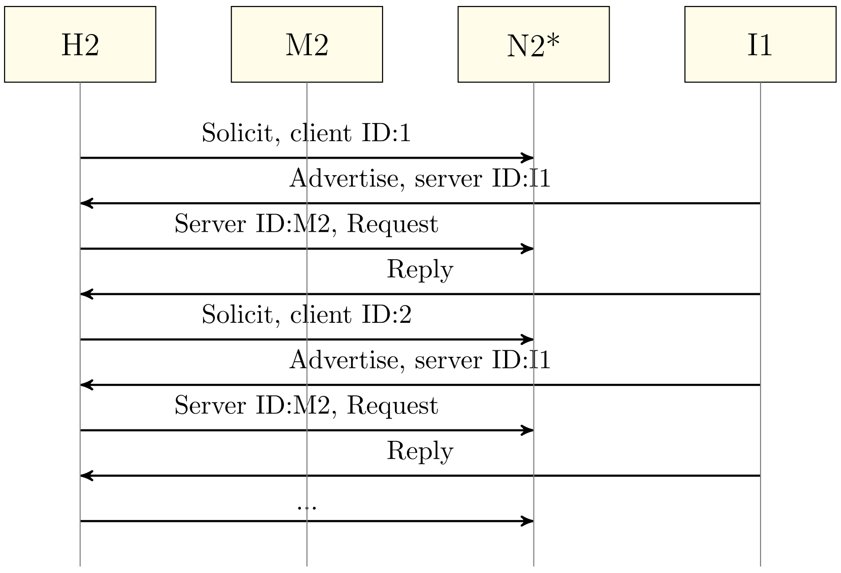

Figure B.108: Flow Diagram for the Scenario S.AM.06

Discussion

To detect a DoS attack where the attacker tries to deplete the address pool the DHCP Server Log (I.AH.11) and the DHCP Binding Database (I.AH.07) are useful. Based on the log a address assigning rate can be determined and the with the Binding Database the current level of usage. To retrace the port of the attacker the actual DHCP messages with the Frame Header (I.DE.01) are needed. Either with the source IP address (see scenario S.NG.01) or the data-link source address and the Filter Database (I.DB.05) the access port of the attacker can be determined.

Required Information I.AH.11, I.AH.07, I.DE.01, I.DB.05

Scenario Dependency STG.01, S.NG.01

B.6.12 DoS DHCP Client after Solicit Message (S.AM.07)

Description

A DHCP server sends an Advertise message with the status code *NoAddrAvail* if the server is not willing to assign an address to the client. An attacker can misuse this mechanism by spoofing an answer from the DHCP server. The attacker claims that the server is not willing to assign the client an address.

H2 sends a DHCP Solicit message. When M2 receives the message, it immediately replies with a spoofed Advertise message with the status code *NoAddrAvail*. The genuine Advertise message from I1 reaches H2 shortly after.

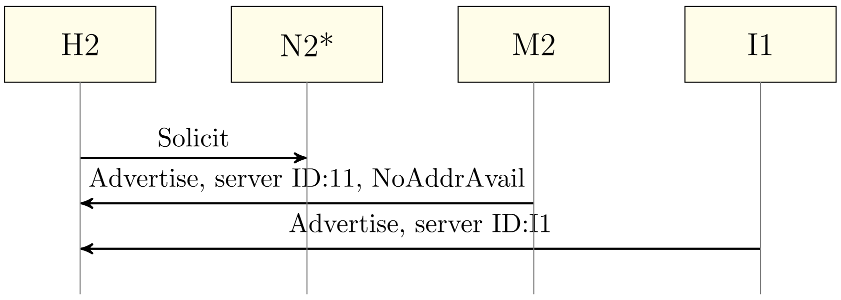

Figure B.109: Flow Diagram for the Scenario S.AM.07

Discussion

The DoS attack with a spoofed Advertise message with the *NoAddrAvail* option set is easy to detect with the DHCP Traffic (I.AH.01). Two Advertise message with the same server and client ID with in the same few seconds indicate this scenario is ongoing. Retracing this scenario is only possible if IP address spoofing is detectable and retraceable (see scenario S.NM.01).

Required Information I.AH.01, I.AH.07, I.DE.01, I.DB.05

Scenario Dependency STG.01, SNM.01

B.6.13 DoS DHCP Client after Renew Message (S.AM.08)

Description

A DHCP client renews its address every Renew Timeout (T2). An attacker can wait for a client renewing its address and immediately reply that the address is not valid anymore. This can be done by the status code *NoBinding* or by setting the lifetime of the address to zero in the reply.

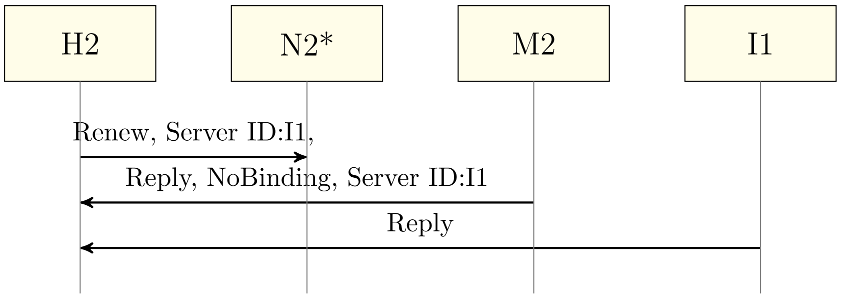

Figure B.110: Flow Diagram for the Scenario S.AM.08

Discussion

The detection and retracing for a DoS attack while a DHCP client is renewing its address is identical to the scenario S.AM.07. In this scenario, however, the detection is based on two Reply messages with the same server and client ID but different options within a few seconds. Retracing this scenario is only possible if IP address spoofing is detectable and retraceable (see scenario S.NM.01).

Required Information I.AH.01, I.AH.07, I.DE.01, I.DB.05

Scenario Dependency STG.01, SNM.01