Department of Networked Systems and Services
Faculty of Electrical Engineering and Informatics
Budapest University of Technology and Economics

Security Analysis of IPv4-as-a-Service IPv6 Transition Technologies

Doctoral School of Informatics

Ph.D. Dissertation
of

Ameen Al-Azzawi

Supervisor: Dr. Gábor Lencse

Budapest, Hungary
2025

Abstract

The proliferation of IoT (Internet of Things) devices and the increasing demand for network resources have led to a critical need for IPv6 transition technologies that enable seamless integration with existing IPv4 infrastructure. This thesis presents a comprehensive security analysis of five prominent IPv6 transition technologies 464XLAT (Combination of Stateful and Stateless Translation), DS-Lite (Dual-Stack Lite), Lw4o6 (Light weight 4over6), MAP-E (Mapping of Address and Port using Encapsulation) and MAP-T (Mapping of Address and Port using Translation). By examining the security threats inherent in each technology, a thorough assessment of their vulnerabilities and potential attack vectors is conducted. Furthermore, this research establishes practical test beds to emulate real-world scenarios, allowing for the execution of multiple attack scenarios against the infrastructure of each transition technology.

The security analysis reveals several critical threats, including DoS (Denial of Service) attacks, spoofing, source port exhaustion, man-in-the-middle attacks, information disclosure, etc. By simulating these attacks, the vulnerabilities and weaknesses in the infrastructure of each technology are systematically identified and analyzed. Subsequently, this research proposes effective mitigation methods to address and mitigate these security risks.

The mitigation methods developed in this study focus on enhancing the security posture of each IPv6 transition technology. The proposed measures include packet rate limiting, proper payload encryption, robust filtering techniques, traffic analysis, anomaly detection, etc. These solutions aim to protect against the identified attack vectors and ensure the integrity, confidentiality and availability of network communications.

The proposed mitigation methods are thoroughly validated through extensive testing and evaluation. The effectiveness of the techniques is measured by analyzing their impact on security and performance. The results demonstrate that the proposed measures significantly improve the security of the IPv6 transition technologies without compromising their operational efficiency.

By addressing the security challenges of 464XLAT, DS-Lite, Lw4o6, MAP-T and MAP-E, this thesis contributes to the existing body of knowledge on IPv6 transition technologies. The findings provide valuable insights for network administrators, service providers and researchers, aiding in the secure deployment and management of these technologies. Additionally, this research sheds light on the potential implications of implementing the proposed mitigation methods within real-world network environments.

In conclusion, this thesis offers a comprehensive analysis of the security landscape of IPv4-as-a-Service IPv6 transition technologies. Through the identification of vulnerabilities and the proposal of effective mitigation techniques, this research facilitates the adoption of these technologies in a secure and resilient manner. Ultimately, this work contributes to the advancement of secure network communication protocols and the successful transition from IPv4 to IPv6 in contemporary network architectures.

Contents

List of Figures

List of Tables

Acknowledgements

For the test-bed implementations, the resources of NICT StarBED, Japan, were used for conducting the experiment. The author thanks Shuuhei Takimoto for the opportunity to use these resources.

The author thanks Max Rottenkolber and Ian Farrer from Snabb's community for their valuable input regarding Snabb's configuration in order to build the lw4o6 test-bed, which we present in Chapter 8.

The author thanks Sándor Répás, András Gerendás and Omar D'yab for their reading and commenting on the manuscript of one of the papers that we published [7], which I built Chapter 7 upon.

The author thanks Professor Youki Kadobayashi, Laboratory for Cyber Resilience, Nara Institute of Science and Technology, Japan and Bertalan Kovács from Budapest University of Technology and Economics for reading and commenting on the manuscript of the paper where Chapter 9 is based upon.

The author would like to thank his wife (Anastasia Kiseleva) for her valuable support in proofreading two papers that were submitted throughout our research path, where we based Chapter 8 and 9 upon them.

Declaration

I, Ameen Al-Azzawi, declare that this doctoral dissertation is entirely my own work, utilizing exclusively the provided sources. I have diligently identified and distinguished any sections borrowed from other sources, whether verbatim or paraphrased and have cited their origins appropriately.

Name: Ameen Al-Azzawi

Date: 13 April 2025

Chapter 1

Introduction

1.1 Background

The rapid growth of the Internet has led to the exhaustion of available IPv4 addresses, necessitating the transition to IPv6, the next generation Internet Protocol. IPv6 offers a significantly larger address space compared to its predecessor, enabling the connectivity of an enormous number of devices and accommodating the future growth of the Internet. While the transition to IPv6 is crucial for the continued expansion and innovation of the Internet, it presents numerous challenges, particularly in terms of security.

IPv6 transition technologies play a critical role in facilitating the coexistence and interoperability between IPv4 and IPv6 networks during the transition period. These technologies enable organizations and service providers to gradually adopt IPv6 while maintaining connectivity with the existing IPv4 infrastructure. Some commonly used IPv6 transition mechanisms include Dual Stack, Tunneling and Translation techniques.

Dual Stack allows both IPv4 and IPv6 protocols to coexist on the same network infrastructure, providing a seamless transition path for organizations. It involves configuring networking devices, hosts and applications to support both IPv4 and IPv6 simultaneously [66].

Moreover, the tunneling mechanism is adopted by other IPv6 transition technologies to encapsulate IPv6 packets within IPv4 tunnel packets, facilitating the connection of IPv6 islands (6in4). This mechanism typically employs static pre-configured tunnels between two nodes [66]. However, this solution is not widely utilized by the average Internet user. On the other hand, the 4in6 encapsulation, as defined by RFC 2473 [19], is also not widely adopted among average Internet users. Nonetheless, it serves as an essential building block for the following three solutions: DS-Lite [23], lw4o6 [21] and MAP-E [80].

The translation technique, as implied by its name, involves translating IP addresses from one version to another, enabling communication between IPv4 hosts over an IPv6 island. This can be achieved through a single translation method, such as the combination of DNS64 [13] and NAT64

(Network Address Translation) [12], or a double translation method using two translation routers, as seen in 464XLAT [61].

However, as with any new technology, the deployment of IPv6 transition mechanisms introduces potential security vulnerabilities and risks that must be thoroughly understood and addressed. Security concerns related to IPv6 transition technologies arise from the complexity of the transition process and the differences between IPv4 and IPv6 protocols. IPv6 introduces new addressing schemes, network management protocols and extension headers, which require careful analysis to ensure a secure transition. Moreover, the coexistence of IPv4 and IPv6 networks introduces various challenges, such as address translation mechanisms, tunneling protocols and dual-stack implementations, which can create potential security gaps.

Understanding the security implications of IPv6 transition technologies is of paramount importance to ensure the integrity, confidentiality and availability of network services and data during the transition process. Security vulnerabilities can expose networks to threats such as unauthorized access, data leakage, network disruptions and DoS attacks. Therefore, a comprehensive security analysis of IPv6 transition technologies is necessary to identify potential risks, develop mitigation strategies and design secure transition architectures.

In recent years, researchers and practitioners have made significant efforts to investigate the security aspects of IPv6 transition technologies. Several studies have identified vulnerabilities and weaknesses in various transition mechanisms, highlighting the need for robust security measures.

It is also worth mentioning that there are more than 30 IPv6 transition technologies [51]. However, we have enlisted only the five most important IPv4aaS (IPv4-as-a-Service) technologies, because in the current stage of transitioning the Internet from IPv4 to IPv6, the typical challenge for the service providers is to use only IPv6 in their access and core networks to save costs (the operation of a dual-stack network is costly) while maintaining the opportunity for their customers to do the followings:

- uses legacy applications that can use only IPv4, and to access service,
- accesses legacy services that are available only over IPv4.

In this thesis, we aim to provide a detailed examination of the security aspects of the five most important IPv4aaS IPv6 transition technologies: 464XLAT, DS-Lite, Lw4o6 and MAP-T and MAP-E, then assess their security implications. Our research will encompass both theoretical analysis and practical evaluations to identify vulnerabilities.

1.2 Motivation

The transition from IPv4 to IPv6 is not just a technical necessity but a critical step towards ensuring the future viability and scalability of the Internet. With the exhaustion of available public IPv4 addresses in 2011 [50], organizations and service providers are increasingly adopting IPv6 to

accommodate the growing number of connected devices and to enable the deployment of emerging technologies such as the IoT, cloud computing and mobile networks [44]. However, alongside the benefits of IPv6 come significant security challenges that must be addressed to ensure a secure and reliable transition.

The motivation behind this research stems from the pressing need to comprehensively analyze the security aspects of IPv6 transition technologies. While the adoption of IPv6 is essential, it introduces new attack vectors and potential vulnerabilities that may be exploited by malicious actors. The complexity of the transition process, the coexistence of IPv4 and IPv6 networks and the differences in protocol implementations necessitate a thorough investigation of the security risks associated with IPv6 transition technologies.

By conducting a detailed analysis of the security implications, we aim to contribute to the development of robust and secure transition strategies. The outcomes of this research will not only enhance the understanding of security challenges in the context of IPv6 transition but will also provide valuable insights and recommendations for organizations, service providers and policymakers involved in the transition process.

Moreover, the lack of comprehensive studies and evaluations of the security measures implemented in various IPv6 transition technologies highlights the need for in-depth research in this area. Although there have been notable efforts to propose security enhancements and countermeasures, their effectiveness, practicality and potential trade-offs require further investigation.

Through this research, we seek to address the following key motivations:

- Identifying vulnerabilities and risks through highlighting the potential security vulnerabilities and risks introduced by IPv6 transition technologies. By thoroughly examining the various transition mechanisms, their implementations and the interactions between IPv4 and IPv6 networks, we aim to uncover potential attack surfaces and weaknesses that could be exploited by adversaries.
- Implement several attacking scenarios against the infrastructure of the IPv6 transition technologies to test their already emplaced security measures and to identify the vulnerabilities.
- Developing mitigation strategies: Once the vulnerabilities and risks are identified, the next motivation is to propose effective mitigation strategies. These strategies may include the development of secure implementation guidelines or the enhancement of existing security mechanisms.
- Evaluating the impact of each attacking scenario on the targeted transition technology, this evaluation will be based on several factors, including the severity of the attack's impact on the targeted machine and the complexity of mitigating such an attack.
- Contributing to secure transition frameworks: Ultimately, this research aims to contribute to the development of secure transition frameworks and guidelines for organizations and service providers. By identifying best practices, proposing effective security measures and

highlighting potential pitfalls, we aim to assist stakeholders in achieving a smooth and secure transition to IPv6.

This thesis analyzes IPv6 transition technologies and their security implications through empirical evaluations, theoretical analyses and practical assessments. Our goal is to enhance the understanding of secure IPv6 transition, contributing to a more secure and reliable future Internet.

1.3 Applied Methods

There are several methods to conduct every research plan. In our case, we have applied two methods which will be explained in the upcoming Chapters: -

- A method called STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege), which is a threat modeling framework that is commonly used in the field of computer security. It helps identify and categorize potential threats or risks to a system or application [78]. Therefore, the application of STRIDE to the analysis of security vulnerabilities in IPv6 transition technologies allows for a thorough examination of potential threats across a wide range of these technologies (464XLAT, DS-Lite, lw4o6 and MAP-E / MAP-T). This approach greatly simplifies the process of conducting a comprehensive threat analysis for almost all IPv6 transition technologies. For more details regarding STRIDE, please refer to Section 3.1.
- Another method presented by Marius Georgescu that deals with all IPv6 transition technologies in a way it classifies the high number of IPv6 transition technologies into a low number of categories and handles the categories [34]. Furthermore, this method was later amended by Gábor Lencse [50], where the author suggested conducting a comprehensive examination at two distinct levels: the individual IPv6 transition technologies and the level of their most important implementations. For more information, please refer to Section 3.2.

1.4 The structure of the dissertation

In Chapter 2, we define five IPv4aaS technologies, discuss their respective topologies, detail the packet journey from IPv4-only client to IPv4-only server across their infrastructure and explore the novel contributions they bring to the scientific community. Chapter 3 delves into the threat modeling techniques applied in our security analysis, including the use of STRIDE. In Chapter 4, we present prior research efforts related to each of the five targeted IPv4aaS technologies.

Furthermore, Chapters 6, 7, 8 and 9 examine the security analysis of 464XLAT, DS-lite, lw4o6 and MPA-T/MAP-E, respectively. In these chapters, we employ the STRIDE method to evaluate each technology, highlight potential attack vectors, implement various attacking scenarios and subsequently propose mitigation strategies. Finally, in Chapter 10, we propose a new security analysis methodology aims to refine and enhance existing methods.

Chapter 2

The 5 IPv4aaS technologies

The general purpose of the IPv4aaS technologies is to provide the home network with IPv4 connectivity across an IPv6-only access network.

2.1 464XLAT

464XLAT has a double translation mechanism, where its main structure is shown in Fig. 1, it is divided into two sides; CLAT (Customer-Side Translator) & PLAT (Provider-Side Translator).

2.1.1 CLAT

It algorithmically translates 1:1 private IPv4 addresses to global IPv6 addresses and vice versa [61]. It acts as an IPv6 router, DNS (Domain Name System) proxy and DHCP (Dynamic Host Configuration Protocol) server for local clients as well. Normally, CLAT must know its own prefix and PLAT side prefix in order to use it as a destination for its outgoing packets [61].

As for the IPv6 client, its own packets will pass through the CLAT without the need for any translation process and will be forwarded to the PLAT directly.

2.1.2 PLAT

It translates N:1 global IPv6 addresses with the previously set CLAT prefix to public IPv4 addresses and vice versa [61], it actually implements a stateful NAT64 gateway as described in RFC 6146 [12]. We give an easy-to-understand introduction to explain the operation of 464XLAT by Fig. 2. The client in the bottom left-hand side corner of the figure (using private IPv4 address 192.168.1.2) wants to connect to the server in the top left-hand side corner (using public IPv4 address 198.51.100.1). The prefix at the CLAT side is 2001:8db:aaaa::/96, while the prefix at the PLAT is 2001:8db:1234::/96. The CLAT translates the IPv4 packet into an IPv6 packet, in which the source address will be 2001:db8:aaaa::192.168.1.2 and the destination address will be 2001:db8:1234::198.51.100.1.

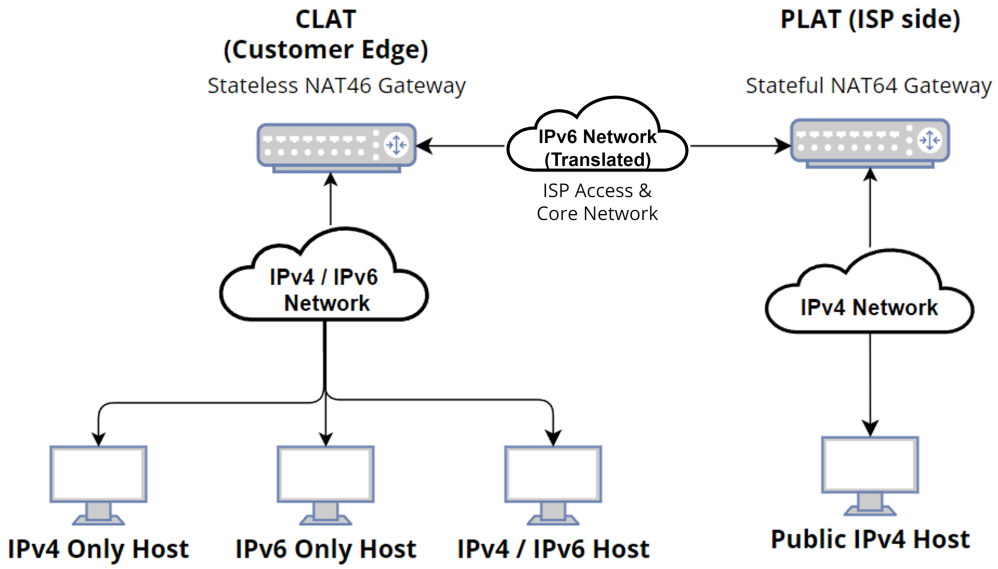

Figure 1: Overview of 464XLAT Architecture

At the PLAT side, the 2001:db8:1234::/96 prefix is discovered in the destination address, and an IPv4 packet is built using the embedded 198.51.100.1 IPv4 address as the destination address and the source IPv4 address is chosen from the pool of 192.0.2.1-192.0.2.100 (this time it happened to be 192.0.2.1). The source port is also replaced when needed, and the connection is registered into the state table of the NAT64 translator to be able to perform the stateful translation in the reverse direction, too. For further details of the stateful NAT64 translation, please refer to RFC 6146 [12]. Besides double translation, there are two other possible scenarios. If both the client and the server have IPv6 addresses, then there is no translation at all, but native IPv6 is used. If the client has an IPv6 address, but the server has only an IPv4 address, then there are two possible modes of operation:

- If DNS64 is configured, then the DNS64 server returns an IPv4-embedded IPv6 address, and only a single translation happens at the PLAT, which is the DNS64 + NAT64 solution [77].
- If no DNS64 is configured, then the client uses IPv4 and double translation happens as described above.

Furthermore, certain devices (clients) are directly linked to the ISP (Internet Service Provider) without traversing through the home gateway, owing to their capacity to function as CPE (Customer Premises Equipment).

2.2 Dual-Stack Lite

The general purpose of DS-Lite is to provide the home network with IPv4 connectivity across an IPv6-only access network. DS-Lite was presented in RFC 6333 [23], where it consists of two main parts: B4 (Basic Bridging Broadband) router and AFTR (Address Family Transition Router).

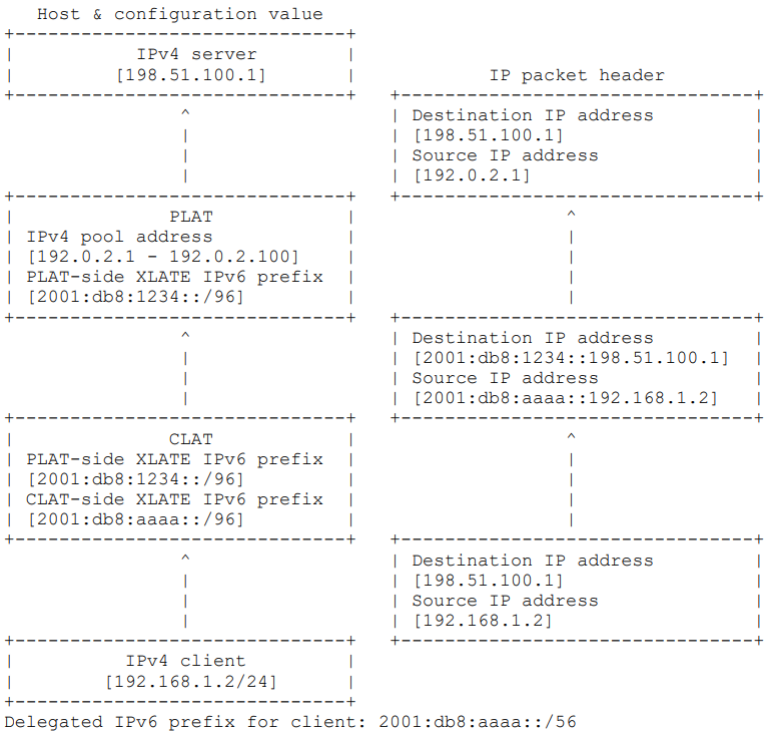

Figure 2: 464XLAT Packet Processing [61]

Fig. 3 illustrates the infrastructure of DS-Lite and how it functions (with some simplification, please refer to the operation of the AFTR below).

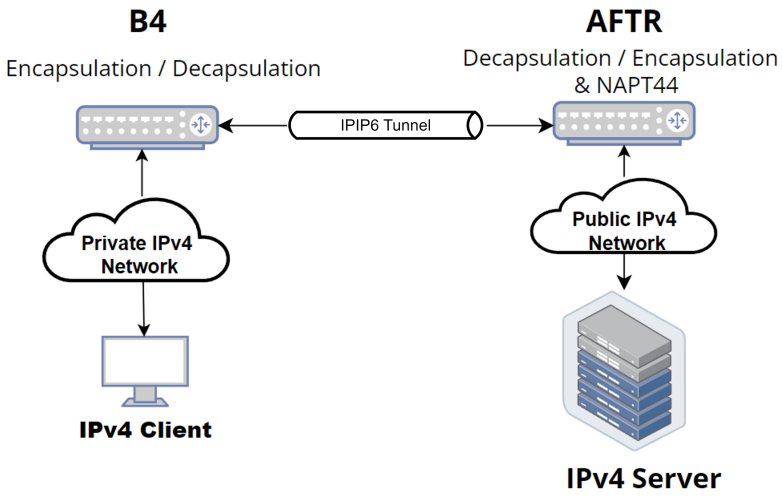

Figure 3: Overview of the DS-Lite architecture [53].

2.2.1 B4 (Basic Bridging Broadband)

This router is responsible for encapsulating IPv4 packets into IPv6 ones and then sending them over the IPv6 network until they reach the AFTR router. It also has another vital role when it processes the returning packets from the AFTR toward the B4, where it decapsulates the IPv6 packet and extracts the original IPv4 packet from the payload [23]. The tunnel that is created by

the CPE, which is the B4, is called the softwire tunnel that connects the B4 with the AFTR. Some facts about the B4 router that were mentioned by RFC-6333 [23]:

- B4 announces itself as the default IPv4 router, and this route is applicable for all IPv4 clients sitting behind this B4 router;
- The B4 router should also announce itself as a DNS server in the DHCP (Dynamic Host Configuration Protocol) option 6 [23];
- As for the operation side, it acts as a DNS proxy for all IPv4 clients who are willing to connect with it and beyond, it forwards those requests to the DNS server of the ISP [23];
- B4 also forwards native IPv6 packets to the AFTR without any intervention;
- The default structure of DS-Lite is as explained above (client ⇒ CPE ⇒ ISP). However, some devices (IPv4 clients) are connected directly to the ISP without the home gateway. The reason behind this is that those devices have the ability to act as CPEs themselves [23].

2.2.2 AFTR (Address Family Transition Router)

The AFTR decapsulates the 4in6 traffic that comes from the B4 router; then, it translates the IPv4 packet into public IPv4, which is a function similar to the well-known stateful NAT44 [23]. However, it is a more complex function than stateful NAT44, because here the translation table contains also the Softwire-ID, which is the IPv6 address of the CE device, as shown in Table 1. Of course, it also performs the reverse functions for the returning packets. The technology in general has two types of topologies (Gateway-based and Host-based architecture), and each one has its own applications:

- Gateway-based: Mostly based in residential broadband infrastructure, where the client and the CPE are based in different but directly connected machines [23];
- Host-based: Designed for large-scale deployments and especially when the client is directly connected to the service provider network, which means the IPv4 client and the CPE are both mounted on the same device [23].

In our research, the focus is on the Gateway-based architecture topology because this topology is easier to build for testing purposes and a virtual environment. It is the most represented topology in residential households; for example, IPv4 client ⇒ router ⇒ ISP.

The AFTR has two main interfaces (softwire interface and network interface), and each one of them faces one end of the device:

- Softwire interface: Its main function is connecting B4 with AFTR and translating the datagram with (softwire identifier + source IPv4 address + source port number) to another with (source IPv4 address and source port number) then add this entry to Table 1 if it has not been added before. So, it decapsulates the IPv4 datagram from the IPv6 datagram (for the packets coming from the B4 side), and performs the reverse by encapsulating the IPv4 datagram

into the IPv6 datagram (for the packets heading towards the B4 side);

- Network interface: Resides on the other side of the AFTR—the WAN (Wide-Area Network) side—and translates the decapsulated source IPv4 and source port into the interface IP (192.0.2.1) and source port 5000 (for example purposes only). The interface is also in charge of the translation in the reverse direction (IPv4 server ⇒ AFTR ⇒ B4), where it translates the original packet's destination IP and destination port (according to AFTR's Carrier-Grade NAT translation table) to the destination IP and destination port of the softwire.

However, it is worth mentioning that the functions of the two interfaces work hand-in-hand, in a way that the AFTR router performs more than the role of a conventional NAT44 translator. In each packet exchange, the AFTR router examines the softwire ID (CE's IPv6 address) of the transmitted packet. It compares this ID with the entries listed in Table 1. The encapsulation/decapsulation + NAT44 process is initiated only when a matching softwire ID is identified.

Fig. 4 shows the outbound communication between B4 and AFTR. The NAT table in this example is configured in a way that translates any incoming packet with the source IP address 10.0.0.1 and source port number 10000 to the IP/port pair of 192.0.2.1:5000.

- Once Datagram 1 reaches the B4 router, it will be encapsulated into Datagram 2, which is an IPv4 in IPv6 datagram, and then forwarded to the AFTR through the softwire tunnel;
- When the AFTR receives Datagram 2, it decapsulates it, extracts the IPv4 Datagram from it, and forwards it to the stateful NAT function, which performs the following (according to its NAT table);
- The received datagram with source IP and port pair of 10.0.0.1:10000 should be translated to Datagram 3 with the following specifications: source IP and port pair: 192.0.2.1:5000.

How does the AFTR translation table function?
The IPv6 address for the B4 router is called the softwire-ID. This ID is being shared by every client that wants to connect with the IPv4 server and triggers the DS-Lite communication process. Every single client of those is equipped (by the B4 router) with a source IPv4 address [RFC 1918] such as 10.0.0.1, and it is unique within its network.

When an embedded packet (source IP address: 10.0.0.1, source port number: 10000) is forwarded from the B4 to the AFTR through the softwire tunnel, the AFTR combines the softwire-ID with this packet's details as one entry: softwire-id/IPv4/TCP/10000. In fact, this is only half of the entry. The other half will be the IP address of the AFTR network interface, the protocol identifier (e.g., TCP or UDP) and the source port number (192.0.2.1/TCP/5000), as shown in Table 1. The reverse direction of the packet (IPv4 server ⇒ AFTR ⇒ B4 ⇒ IPv4 client) functions in a similar manner:

- AFTR receives Datagram 4 through the network interface with a public IPv4 address, then processes it by checking the internal NAT table and looking for matching entries. In this

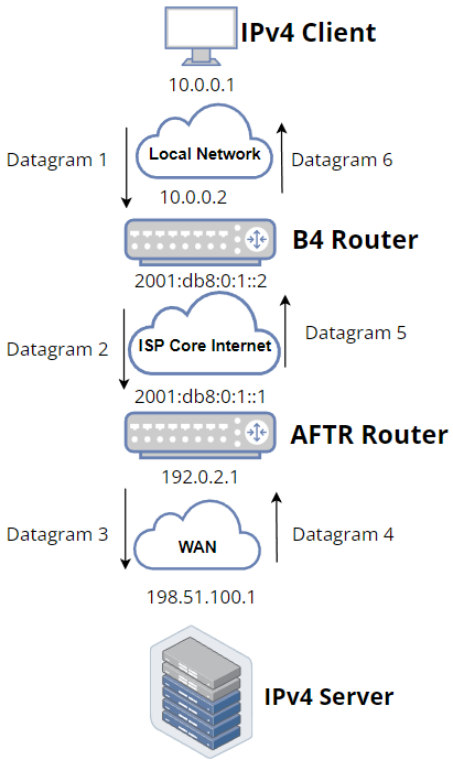

Figure 4: Overview of the DS-Lite datagram path

case, Datagram 4 has the following details:

  - IPv4 destination address: 192.0.2.1;
  - Destination port: 5000;
  - Protocol is TCP;

- The corresponding entry does exist in the NAT table (see Table 1); AFTR, therefore, translates the IPv4 packet of Datagram 4 to IPv4 destination address 10.0.0.1 and TCP destination port 10000;
- AFTR then encapsulates the translated IPv4 packet into an IPv6 packet and sends Datagram 5 over to the B4 router at 2001:db8:0:1::2 IPv6 address (which the AFTR knows from the table entry itself as softwire-ID);
- B4 receives Datagram 5, decapsulates the embedded packet, extracts the original IPv4 packet from the 4in6 Packet and forwards it accordingly to the IPv4 client (10.0.0.1:10000) as Datagram 6.

Table 1: Dual-Stack Lite Carrier-Grade NAT Translation Table

Softwire-ID/IPv4/Protocol/Port	IPv4/Protocol/Port
2001:db8:0:1::2/10.0.0.1/TCP/10000	192.0.2.1/TCP/5000

Table 2 lists all IP addresses and port numbers for the DS-Lite packet route for Datagrams 1–6.

Table 2: DS-Lite Datagrams

Datagram	Header Details
IPv4 Datagram 1	IPv4 Src: 10.0.0.1 IPv4 Dst: 198.51.100.1 TCP Src: 10000 TCP Dst: 80
IPv6 Datagram 2	IPv6 Src: 2001:db8:0:1::2 IPv6 Dst: 2001:db8:0:1::1 IPv4 Src: 10.0.0.1 IPv4 Dst: 198.51.100.1 TCP Src: 10000 TCP Dst: 80
IPv4 Datagram 3	IPv4 Src: 192.0.2.1 IPv4 Dst: 198.51.100.1 TCP Src: 5000 TCP Dst: 80
IPv4 Datagram 4	IPv4 Src: 198.51.100.1 IPv4 Dst: 192.0.2.1 TCP Src: 80 TCP Dst: 5000
IPv6 Datagram 5	IPv6 Src: 2001:db8:0:1::1 IPv6 Dst: 2001:db8:0:1::2 IPv4 Src: 198.51.100.1 IPv4 Dst: 10.0.0.1 TCP Src: 80 TCP Dst: 10000
IPv4 Datagram 6	IPv4 Src: 198.51.100.1 IPv4 Dst: 10.0.0.1 TCP Src: 80 TCP Dst: 10000

For a better understanding of the functionality of AFTR translation even more, please refer to RFC 6333 [23], where further details about the translation steps were explained.

2.3 Lightweight 4over6

DS-Lite has an issue with scalability as all of the NAPT operations were executed by the AFTR router, which makes it very hard to scale its operation. Therefore, lw4o6 technology was invented to function as an improved version of DS-Lite [21]. To understand the essence of this extended technology, we will analyze the added values that lw4o6 presents.

Fig. 5 shows the topology of lw4o6 with its main components. Lw4o6 works in a similar man-

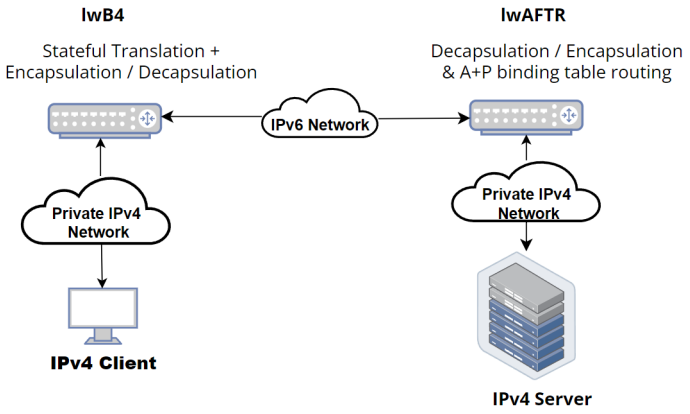

Figure 5: Lw4o6 Topology

ner as DS-Lite, however, it has one main improvement, which made it more scalable than the conventional DS-Lite. The stateful translation has been moved from the centralized AFTR into the B4. This change made a big difference for the ISPs, and can be summarized by the below points:

- Lw4o6 optimizes the work for ISPs by avoiding the complicated process of stateful translation.
- The ”per-flow state” method is replaced by ”per subscriber state”, which will save a lot in terms of CPU and memory consumption [21]. This is considered to be the most important feature of lw4o6, where every subscriber or CPE has an allocated and dedicated port set that can be used in his communication. This opens up the possibility for multiple lwB4 routers to share the same public IPv4 address in their softwire (tunnel) under the condition that they use two different port sets [21].
- The fact that no stateful translation is required on the lwAFTR side means that considerably less logging in the ISP side is actually required [21].

In other words, lw4o6 is an optimization of DS-Lite [21], as it reduces the overhead at the lwAFTR side by relocating the stateful translation to the lwB4 side.

Lw4o6 infrastructure consists of two main types of routers, lwB4 and lwAFTR:

- lwB4: works as stateful NAT44 translator and encapsulator / decapsulator.
- lwAFTR: works only as encapsulator / decapsulator.

Softwire is a mechanism that allows the encapsulation and transport of IPv4 traffic over an IPv6 network (or vice versa). Moreover, it is a tunnel (“virtual wire”) that carries IPv4 traffic over an IPv6 infrastructure. In the lw4o6 case, softwire encapsulates IPv4 packets within IPv6 packets and facilitates communication between two IPv4 networks using an IPv6 network [21].

2.3.1 Port Set Allocation

As described previously, lwAFTR allocates a specific port set for every lwB4 router using a method illustrated in RFC-7597 [80] Section 5.1, where PSID (Port Set Identifier) can be calculated as follows.

For example, to configure a softwire at the lwAFTR side, every softwire has to have the below three parameters (the numbers we used here are just examples):

- Port set size = 10 $\Rightarrow 2^{10} = 1024$ ports per set.
- PSID length = 6 $\Rightarrow$ number of port sets: $2^6 = 64$, it is also called "the sharing ratio" [80].
- PSID = 1 $\Rightarrow$ allocated ports range = [1024-2047].

To explain it in a simpler way, let us calculate the number of concurrent lwB4s (subscribers) who can share the same public IPv4 address, while having different port sets. The total number of source ports numbers is $2^{16} = 65536$, divided by 64 (number of port sets) is 1024. This number represents the size of one ports set, which can be also concluded from PSID size value (2^{10}). So, we have the number of sets, and the size of the sets themselves, all that is left is to select the PSID value, which will decide the exact port set to be allocated for the specific CPE (subscriber).

- PSID = 0 $\Rightarrow$ allocated ports = [0 - 1023].
- PSID = 1 $\Rightarrow$ allocated ports = [1024 - 2047].
- PSID = 8 $\Rightarrow$ allocated ports = [8192 - 9215].

In conclusion, with PSID length = 6, we can support 64 different CPEs (subscribers)) per public IPv4 address with 1024 ports for each of them. Furthermore, by selecting PSID = 1, the range of allocated ports will amount to [1024 - 2047]. However, if we exclude the first set, which contains the well-known ports [0 - 1023], we end up with 63 subscribers. That means 63 subscribers have the possibility of sharing the same public IPv4 address.

2.3.2 Packet Path Through the Lw4o6 Infrastructure

For a better understanding of the packet translation, encapsulation and decapsulation process of lw4o6, we follow the packet flow step by step. Fig. 6 shows the topology of lw4o6 with its elements, where its operation can be summarized as below:

- The IPv4 client sends a packet with the following details:
  - Source IP address: Client IPv4 address (10.0.0.2)
  - Source port number: 5000
  - Destination IP address: IPv4 server address (192.0.2.2).
- The lwB4 receives the packet and performs the following steps:

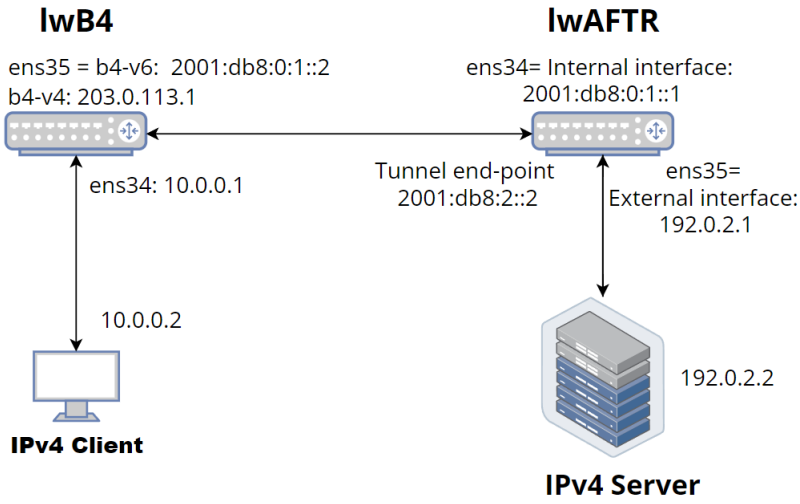

Figure 6: Lw4o6 Topology in details

  - NAPT44 function: the private source IPv4 address is replaced with a public IPv4 address (203.0.113.1). The source port number is replaced with an unused one from the range assigned to the subscriber. Assuming that the assigned range is 1024-2047, let the new source port number be 1050. At lwB4, the entry of the NAPT binding table is shown in Table 3.
  - Encapsulation: lwB4 encapsulates the IPv4 packet into an IPv6 packet by prepending an IPv6 header to it and forwards the 4in6 packet to the lwAFTR through the softwire tunnel with the following details:
    * Source IP address: lwB4's IPv6 address: 2001:db8:0:1::2
    * Destination IP address: lwAFTR tunnel end-point IPv6 address: 2001:db8:2::2
    * Encapsulated IPv4 packet content:
      · Source IP address + port number: 203.0.113.1:1050
      · Destination IP address: 192.0.2.2

- The lwAFTR router receives the 4in6 packet with the same content as above. The lwAFTR decapsulates the IPv4 packet and scans its binding table (Table 7), then acts accordingly:
  - If a matching entry is found, then the IPv4 packet is forwarded to the IPv4 Internet via the external interface.
  - Otherwise the packet is dropped.

Table 3: LwB4 Router NAPT44 Binding Table

Private IPv4	Source port	External IPv4	Temporary Port	Transport Protocol
10.0.0.2	5000	203.0.113.1	1050	TCP

- Finally, the IPv4 packet arrives to the IPv4 server.

Packets in the reverse direction: IPv4 server ⇒ lwAFTR ⇒ lwB4 ⇒ IPv4 client, are processed as below:

- IPv4 server replies and sends the packet to lwAFTR with the following details:
  - Source IP address: 192.0.2.2
  - Destination IP address + port number: 203.0.113.1:1050
- LwAFTR receives the above packet on the lwAFTR's external interface, then the lwAFTR scans Table 7, looking for a matching entry. Port number 1050 is a part of the port range [1024-2047], where PSID 1 refers to it explicitly. Therefore, we have a matching entry. The lwAFTR encapsulates the IPv4 packet into an IPv6 packet using "b4-ipv6" as the IPv6 destination address and forwards the resulting 4in6 packet to the lwB4 router with the following details:
  - Source IPv6 address: 2001:db8:2::2 (tunnel end-point).
  - Destination IPv6 address: 2001:db8:0:1::2
  - Encapsulated IPv4 packet content:
    - Source IPv4 address: 192.0.2.2
    - Destination IPv4 address + port number: 203.0.113.1:1050
- LwB4 receives the above 4in6 reply packet and performs the following:
  - In case of incorrect parameters, such as port number, then the packet will be dropped immediately.
  - Decapsulates the IPv4 packet and scans its NAPT binding table (Table 3) for a match. The lwB4 router then rewrites the destination IPv4 address and port number of the packet based on the found entry to 10.0.0.2:5000, forwarding it to 10.0.0.2.
- IPv4 client receives the packet with the below details, which concludes the packet's journey:
  - Source IPv4 address: 192.0.2.2
  - Destination IPv4 address + port number: 10.0.0.2:5000

The lwAFTR router can have multiple softwires configured in its binding table and provisioned to communicate with multiple lwB4 routers [21].

2.4 MAP-E / MAP-T

MAP-E, and MAP-T technologies employ IPv4-embedded IPv6 addresses to enable the coexistence of IPv4 and IPv6 networks by facilitating communication and translation between the two protocols. To achieve this, both MAP-E and MAP-T utilize a stateless algorithm, which involves

incorporating specific segments of the customer's allocated IPv4 address (or a portion of an address in cases involving A+P routing) into the delegated IPv6 prefix given to the client. This ingenious approach facilitates the provisioning of a large number of clients using just a single MAP rule, referred to as a "MAP domain". Moreover, this algorithm enables direct IPv4 peer-to-peer communication between hosts provisioned under common MAP rules.

In general, the MAP-E / MAP-T topology consists of two main routers: CE (Customer Edge) and BR (Border Relay). In practice, the CE router takes on the task of performing stateful NAPT44 (Network Address and Port translation between private and public IPv4 addresses), effectively translating private IPv4 source addresses and source ports into an address and port range as dictated by the application of the MAP rule to the delegated IPv6 prefix. The specific size of the client's address/port allocation is governed by a configurable parameter, offering a level of flexibility in the deployment process.

In Fig. 7, we illustrate the main differences between MAP-E and MAP-T topologies and the core of their functionalities, which are summarized below:

- Both technologies consist of two main components, the CE router and the BR router and each one of them has its own purpose and functionality.
- Both technologies have a shared feature that they critically depend on, which is called "Port Mapping". This feature allocates a certain range of UDP (User Datagram Protocol) / TCP (Transmission Control Protocol) ports to certain CE routers so that applications of the user (IPv4 client) can communicate with the IPv4 server. The port sets are previously provisioned in the CE and BR routers.

  The benefit of port mapping is the ability for multiple CE routers to share the same public IPv4 address, given that they use unique port sets. The port set is selected through a parameter called PSID (Port Set Identifier) [80].
- Both technologies adopt the MAP-domain deployment, which represents one or multiple CE and BR devices connected through a virtual link [80]. The ISPs can deploy more than one MAP domain as needed.
- Both MAP-T and MAP-E configure one or more MAP-rule(s), which are sets of parameters that regulate the communication between the MAP nodes (CE and BR) or among the CEs themselves, especially when it comes to IPv4 /IPv6 prefixes and port sets. Those rules are unique among the MAP domains such as BMR (Basic mapping rule), FMR (Forwarding Mapping Rule) and DMR (Default Mapping Rule).

2.4.1 MAP-E Operation

Fig. 7 shows a simple topology of MAP-E, which consists of four elements: IPv4 client, CE, BR and IPv4 server. MAP-E was originally presented in RFC-7597 [80]. We explain below the packet journey throughout the MAP-E infrastructure and what are the CE and BR roles here:

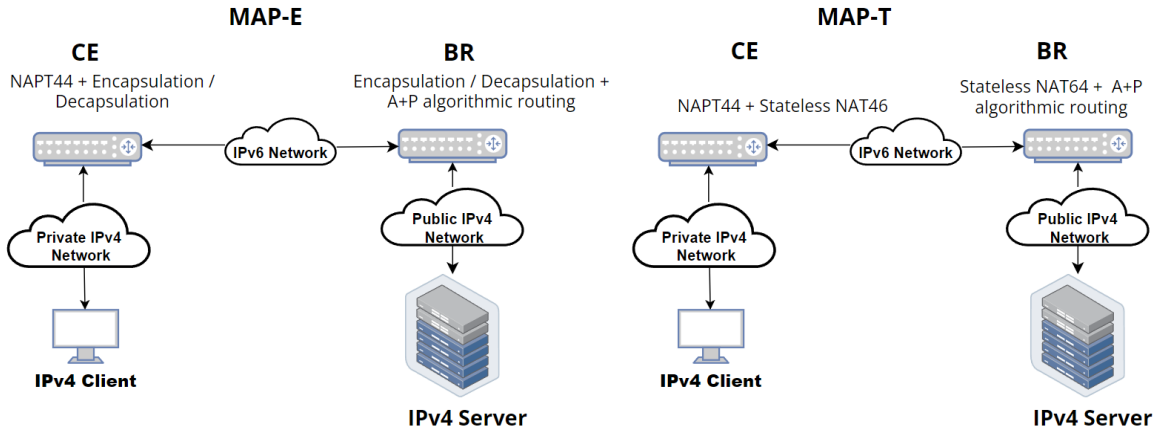

Figure 7: MAP-E and MAP-T Topologies

Table 4: CE router NAPT table

Src. IP Address	Src. Port Number	External IP Address	Temporary Port Number	Dest. IP Address	Dest. Port Number	Transport protocol
10.0.0.2	5000	203.0.113.1	1050	192.0.2.2	80	TCP

- The IPv4 client sends an UDP / TCP packet to the IPv4 server.
- The CE router receives the packet and performs the NAPT44 process. The NAPT44 involves the translation of the private source IPv4 address into a public one along with the original TCP / UDP source port number being mapped to a port number from the assigned ports set. The CE router is responsible for the translation process and is preconfigured with a specific set of port numbers allocated for this purpose. The CE router then adds an entry to its NAPT44 binding table (Table 4), because the CE router here acts as a stateful NAPT44 translator. Finally, CE encapsulates the resulting IPv4 packet into an IPv6 packet and sends the 4in6 packet to the BR router.
- The BR router receives the 4in6 packet, decapsulates the IPv6 packet, selects a best-matching MAP domain rule (based on the prefixes and port set), then forwards the packet to the IPv4 server.
- The IPv4 server receives the packet, and replies with CE's public IPv4 address as a destination address.
- The BR router receives the packet, encapsulates it in an IPv6 packet and forwards it to the CE through the IPv6 interface.
- The CE router receives the packet back, decapsulates the public IPv4 address and looks for a matching entry from Table 4. When a matching entry is found, the source IPv4 address and the source port number will be translated back according to the found entry, and the packet will be forwarded to the IPv4 client. If no matching entry is found, then the CE router will simply drop the packet.
- Finally, the IPv4 client receives the answer of the IPv4 server.

2.4.2 MAP-T Operation

MAP-T was presented in RFC-7599 [56], its infrastructure does not differ much from MAP-E. As shown in Fig. 7, the main difference is that MAP-T uses translation instead of encapsulation on its MAP nodes (CE and BR routers). For a better understanding of the MAP-T translation process, we go through the packet journey throughout the MAP-T infrastructure:

- The IPv4 client sends an UDP / TCP packet to the IPv4 server.
- The CE router receives the packet, performs NAPT44 translation (just like in MAP-E), then adds the translation entry to its NAPT binding table (Table 4). The CE router then performs a stateless NAT46 translation process by translating the IPv4 packet into an IPv6 packet using the pre-configured prefix (end-user IPv6 prefix) and forwards the translated packet to the BR.
- The BR router undertakes the following actions upon receiving the IPv6 packet:
  - Selects a matching MAP domain based on the longest address match by comparing the IPv6 source address with the configured MAP rules within the BR.
  - Based on the selected MAP rule, BR double-checks if the source port is part of the allocated port sets for this MAP rule and drops the packet if the port is outside the allowed range.
  - Derives the source and destination port and translates the IPv6 header to IPv4 one.
  - Forwards the resulting IPv4 packet to its final destination (the IPv4 server in our example).

  The BR router is already provisioned with the public IPv4 address and the allocated port range for each CE within the MAP domain.
- The IPv4 server receives the IPv4 packet and replies with CE's public IPv4 address as the destination address.
- The BR router undertakes the following actions upon receiving the IPv4 packet:
  - Selects the MAP-T domain based on the longest match consisting of the IPv4 address + transport layer port. As a result, the designated FMR and the DMR rules will be applied.
  - Computes the IPv6 source address from the IPv4 source address and BR's IPv6 prefix.
  - Computes the IPv6 destination using the FMR rule based on the IPv4 destination address and destination port number.
  - Forms an IPv6 packet and forwards it to the CE router.
- The CE router receives the IPv6 Packet and performs several processes on the packet, such as pre-routing, filtering and NAPT44 translation as explained below:

  - Checks the source IPv6 address if it matches the IPv6 MAP rule prefixes of the DMR or the FMR (based on the longest match).
  - Checks if the destination port in the transport layer belongs to the allocated port range for the CE router.
  - If any of the previous checks fails such as wrong source IPv6 address or wrong port number, the CE will drop the packet.
  - The CE router performs a stateless NAT64 translation on the IPv6 packet to an IPv4 packet based on the configured BMR.
  - The resulting IPv4 packet will be forwarded to the NAPT44 function, where the CE router translates the public IPv4 address into a private IPv4 address by comparing its details (IP addresses and port numbers) with its NAPT binding table (Table 4). When a matching entry is found, the destination IPv4 address and destination port will be translated according to the found entry and the packet will be forwarded to the IPv4 client. If no matching entry is found, then the CE router will simply drop the packet.
  - Finally, the IPv4 client receives the answer from the IPv4 server.

2.4.3 IPv4 Destination Address and MAP-Domain

When considering the packet traversal in the direction of IPv4 client $\Rightarrow$ CE $\Rightarrow$ BR $\Rightarrow$ IPv4 server, the value of the IPv4 destination address plays a significant role in the translation process within the CE router, which has two scenarios:

- If the IPv4 destination address is within the MAP domain:
  - The source IPv6 address is derived via the BMR.
  - The destination IPv6 address will be synthesized using the FMR.
- If the IPv4 destination address is outside the MAP domain:
  - The source IPv6 address will be the MAP CE's IPv6 address.
  - The destination IPv6 address will be synthesized using the Default Mapping Rule. For instance, if 192.0.2.2 (0xc0000202 in hexadecimal) is the destination IPv4 address and the DMR IPv6 prefix is 64:ff9b::/96, then the IPv4 embedded IPv6 destination address will be translated to 64:ff9b::c000:202.

2.4.4 Port Mapping

As we described previously, MAP-E and MAP-T assign specific port set for every CE router by applying a criterion illustrated in RFC-7597 [80] Section 5.1. Moreover, we already described the process of port-mapping and PSID calculation in Sub-Section 2.3.1, where lw4o6 technology applies the same method.

Chapter 3

Threat Modelling Techniques

There are multiple methods to conduct security analysis and threat modeling, such as CAPEC (Common Attack Patterns Enumeration and Classification) and STRIDE. In a nutshell, CAPEC is a tool that helps developers to think like a hacker, while it provides the basic knowledge of attack patterns [69]. CAPEC can list attack patterns and allow researchers and developers to identify weaknesses in a system, which can be exploited through an attack.

However, we have selected STRIDE to be our threat modeling tool due to its wide coverage of several topologies and most importantly its reputation, as it has been used by multiple research works to analyse potential security threats for multiple network infrastructures.

3.1 STRIDE Methodology

STRIDE was explained in detail in [78]. We briefly presented it in Section 1.3, which is a widely recognized and structured approach used in cybersecurity and software development to systematically identify and address potential security threats and vulnerabilities within a system or application. Security professionals employ the STRIDE method to analyze and assess the potential risks associated with each of STRIDE categories, enabling them to better understand potential attack vectors and formulate effective countermeasures. By incorporating STRIDE, organizations can proactively enhance the security posture of their systems, making informed decisions to safeguard against various forms of cyber threats throughout the development life-cycle and beyond.

According to [78], the best method to test the system's vulnerabilities is to build a DFD (data flow diagram) for the system and apply the STRIDE method to it. The types of potential vulnerabilities depend on what is being done with the data (processing, storing, etc.). Table 5 shows those vulnerabilities accordingly.

In general, the best models are diagrams that help participants understand the software and find threats against it. Each element of the DFD has its own security threats as explained in Table 5. It means that each element is susceptible to some threats while not susceptible to others [78].

Table 5: Potential Vulnerabilities of Different DFD Elements to Different Threats [50]

	Spoofing	Tampering	Repudiation	Information Disclosure	Denial of Service	Elevation of Privilege
Data Flow		✓		✓	✓	
Data Stores		✓		✓	✓	
Processes	✓	✓	✓	✓	✓	✓
Interactors	✓		✓			

STRIDE attacking elements are explained briefly below: -

- Spoofing: the act of an adversary to claim that he is someone else to perform any sort of malicious activities [78].
- Tampering: the possibility of altering the actual content of the exchanged data [78].
- Repudiation: the act of denying the responsibility of a certain act (most probably malicious one), like sending a specific packet [78].
- Information Disclosure: accessing highly confidential information [78].
- Denial of Service: one of the most implemented attacks word-wide and it is based on overwhelming the target with a huge number of queries that are not useful to the targeted server while blocking the legitimate queries from being processed at all [78].
- Elevation of Privileges: the access of a certain user to a higher level sensitive data, that he was not supposed to reach. This could be by granting himself root user privilege and accessing confidential documents [78].

Furthermore, STRIDE has different approaches regarding threat models:

- Assets-centered threat model: anything the attacker wants to access, control or damage. According to [68], the assets-centered threat model is being conducted using 4 approaches: DREAD, Trike, OCTAVE and PASTA. For instance, OCTAVE, which stands for Operationally Threat Asset and Vulnerability Evaluation, is a robust approach but its rather complicated, it takes considerable time to learn and get familiar with its process. Furthermore, its documentation is voluminous [68].
- Attacker-centered threat model: it is based on knowing the attacker, his motivations and skills. It is useful but hard to implement [78].
- Software-centric threat model: focuses on the software being built and the deployed systems, it's the best approach for threat modeling [78], because it supposes that software developers are the best people to understand the software they are developing, which makes the software an ideal starting point to trigger the threat modeling process.

3.2 Technology Classification

As previously presented in Sub-section 1.3, this method starts by categorizing the IPv6 transition technologies based on the way they perform the transition process itself such as tunneling, translation, or dual-stack. Another layer is concerned with the implementation method, that each technology applies such as open-source software or proprietary software. Fig. 8 illustrates the cost and benefit of the application of such a method to threat analysis.

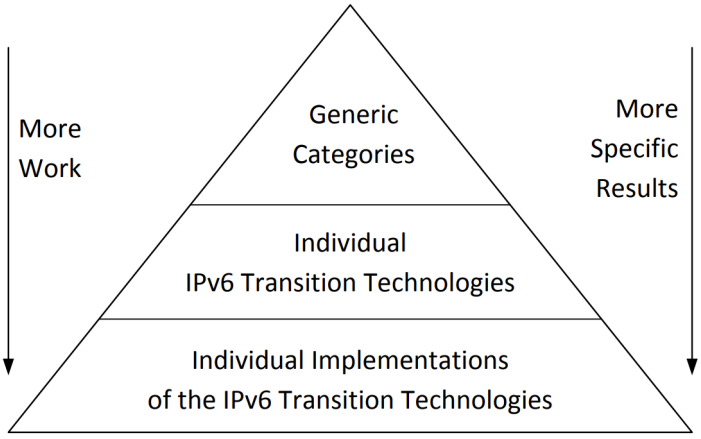

Figure 8: Method hierarchy: Costs and benefits of the different threat analysis methods [50]

We first analyzed the DFD of each technology and then tested some of the identified vulnerabilities using an open-source software-based implementation. By concentrating on the elements within the constructed DFDs representing the main general categories mentioned above, this approach facilitated a comprehensive threat analysis of all five studied IPv4aaS IPv6 transition technologies, addressing the associated threats outlined in the STRIDE model. Consequently, we identified vulnerabilities on two levels: within the IPv6 technology itself and within the implementation method (software) used to build the topology.

3.3 New Proposed Methodology

Georgescu's [34] and Lencse's [50] approaches, which categorize IPv6 transitions based on core network traversal (dual stack, single translation, double translation, encapsulation), have shown limitations in effectively capturing the diverse security vulnerabilities inherent in specific transition technologies. For instance, despite belonging to the same category of double translation, technologies such as 464XLAT and MAP-T exhibit markedly different vulnerabilities. Similarly, within the category of encapsulation, DS-Lite and Lw4o6 demonstrate distinct security risks. This discrepancy underscores the inadequacy of broad categorization schemes in accurately assessing the nuanced security challenges posed by IPv6 transition technologies.

As a result we developed a new novel method that surpasses conventional methods. By leveraging the STRIDE threat modeling technique, we conducted a comprehensive security analysis of prominent IPv6 transition technologies. Our methodology not only evaluates the categorization of

transition technologies but also considers the location and the statefulness of the attacked router, whether it's a customer edge (CE) router or a provider edge (PE) device. Additionally, we introduce an abstraction method to derive potential vulnerabilities at a more general level from those discovered at a more specific level.

For more details, please refer to Chapter 10.

Chapter 4

Literature Review

4.1 464XLAT

Research by Hyunwook Hong [40] has focused on the IPv6 security issues as far as cellular networks are concerned and it came up with different categories of possible attacks. They demonstrated three different DoS attacks on NAT64 block targeting features that only exist in IPv6 cellular networks:

- NAT overflow attack: According to [40], in IPv4 mobile networks, since the majority of major service providers discard packets with spoofed source IP addresses and employ comprehensive routing based on full IPv4 addresses, a device can transmit and receive data packets using just one private IPv4 address assigned via NAT.

  As a result, the maximum of external mapping for a single targeted service is 65,535. Meanwhile, in IPv6 cellular networks, a device can utilize 2^{64} IPv6 addresses. So, if a device creates mapping on NAT64 using all the 2^{64} IPv6 addresses, the result will be 65,535 * 2^{64} mappings, which can lead to overload for NAT64 [40].

- NAT wiping attack: The targeted victim, in this case, is the mapping entry itself. NAT64 uses the N:1 mapping criterion. If an adversary targets the external IPv4 of the NAT64 gateway, N hosts sharing the same external IPv4 address will be liable to a DoS attack. The adversary will send malicious TCP (Transmission Control Protocol) packets with RST flag to wipe out the target mappings within the NAT64. As a result, the mapped users to the very same external IPv4 address will be denied access to their service.

  To do so, the attacker needs to know the TCP 4-tuple of the targeted service (Destination IP address, port number, External IP address of NAT64 and External port number of NAT64).

- NAT Bricking attack: It's a type of DoS attack that also exploits the N:1 mapping algorithm adopted by NAT64. Basically, the adversary can send a huge number of requests using the external IPv4 address(es) of the NAT64 gateway [40]. However, big vendors (Google, YouTube, etc.) have an IP blocking approach if it exceeds a specific number of requests per

minute. Nevertheless, [40] has done an experiment to target the Google Scholar website, which is an IPv4-based site. So, the IPv6 cellular host sends 150 requests per minute to trigger CAPTCHA requests. Every time a CAPTCHA request emerges, the adversary source IP address is changed by turning the airplane mode on and off, this process was repeated 1000 times. Finally, the NAT bricking attack was able to trigger a CAPTCHA request for a total of 631 external IPv4 from Google Scholar, including one of the victim's external IP addresses [40].

Moreover, [42] has explained that the majority of the transition technologies use some form of NAT, NAT44, NAT64, NAT46, etc. and how it is a myth that NAT is putting the user inside this secured box of protective shield from the outside attackers, the sequence of communications below explains how vulnerable the NAT client could be:

- Attacker attracts the victim towards a specific website.
- Victim clicks on the malicious URL and enters the page.
- The page has a hidden form connecting to http://attacker.com:6667 (IRC port).
- The victim submits the form without his consent.
- An HTTP connection is created to the (fake) IRC server.
- The form as well has a hidden value which sends: "OPEN DCC CHAT PORT".
- Router sees an "IRC connection" and then opens a port back through the NAT.
- The attacker now has an open path to the network.

The very same process could have been applied using FTP NAT helper if not IRC. According to [42], today's preferred transition technologies are 6rd, DS-Lite and 464XLAT, while risk Mitigation Strategies can be summarized as follow:

- Minimizing the need for SP-NAT (Service-Provider NAT).
- The more IPv6 established sessions, the less you rely on SP-NAT and all the security issues associated with that.

As for the test-bed, several attempts were conducted by researchers to build an efficient test-bed in order to test the transition technologies, their weak spots and vulnerabilities. A successful test-bed was built by Marius Georgescu [33], in which he measured the latency, throughput and packet loss by adopting 464XLAT transition technology and some other methods as well.

4.2 Dual-Stack Lite

A limited amount of experiments have been published regarding DS-lite and its security analysis. A survey of the most prominent IPv6 transition technologies and their security analysis was carried

out in [51], where DS-Lite was mentioned and its security analysis has been classified as important but replaceable due to several issues mentioned by [48], such as the following:

- The need for two separate physical interfaces at the AFTR;
- The need for high scalability at the AFTR side due to the fact that many B4 routers may be connected to the same AFTR [48];
- The location of deploying AFTR router within the ISP network and the trade-off it creates between the high operation cost and installing an extremely powerful AFTR [48]. The trade-off can be explained by dividing the issue at hand into two options:
  - Deploying AFTR at the edge of the network to cover a small area serves few B4s and requires less-powerful AFTR;
  - Deploying AFTR at the core of the network to cover a big area covers more B4s and requires extremely powerful AFTR (or even more than one AFTR);
- The complexity of deploying a proxy DNS resolver, which will proxy every DNS query stemming from all IPv4 clients heading towards a DNS server that resides in an IPv6 network [48].

Another work [29] conducted a security analysis for DS-Lite in terms of its MIB (Management Information Base), and it consists of several objects. MIB is a module that can be used to monitor the AFTR router within the DS-Lite infrastructure by leveraging SNMP (Simple Network Management Protocol).

According to [29], the most vulnerable objects that are susceptible to attacks are:

- Notification threshold objects: an attacker manipulating a threshold's value to a very low level, which will lead to a flood of useless alarms and thus disrupt the AFTR and its monitoring mechanism, or the attacker sets it to a very high level that makes the idea of setting the alarm literally useless:
  - DsliteAFTRAlarmConnectNumber: The alarm is sent when the number of current DS-Lite tunnels reaches the threshold, which means for every B4 router, the AFTR has to have a separate tunnel for it;
  - DsliteAFTRAlarmSessionNumber: An alarm will be sent when the threshold of sessions per IPv4 user is reached. This metric goes hand in hand with RFC-6333 [23], where AFTR has to be able to log softwire-ID, IP, ports and protocol (see Table 1) in order to keep track of user sessions;
  - DsliteAFTRAlarmPortNumber: An alarm is to be sent when the threshold for the number of ports used by a user is reached or even crossed;
- Table entry objects: An attacker can alter the content of such entries causing the drop of legitimate entries or adding harmful and faulty ones:

- DsliteTunnelTable: Consists of mapping entries of B4 address to AFTR address;
- DsliteNATBindTable: Contains entries about the current active bindings within the NAT table of the AFTR (see Table 1).

What makes entries of Table 1 a potential security threat is the possibility of an attacker assessing the number of hosts being served by a single AFTR router, which reveals sensitive information about the whole topology of DS-Lite [29]. A chance of an inside job is also possible, where an internal employee can access the list of hosts that are in active sessions, which will be a violation of the subscriber's privacy [29]. Moreover, RFC-6334 [38] referred to DS-Lite security briefly and recommended that an IP firewall be implemented in order to avoid MITM (man-in-the-middle) attacks along the softwire connection of DS-Lite.

RFC-8513 [16] proposed another method to implement DS-Lite using the YANG module, which is a schema that facilitates data assessment mechanisms through network management protocols such as NETCONF and RESTCONF. This module allows the administrator to add some features and add-ons to the B4 and AFTR interfaces such as "b4-address-change-limit", "Tunnel-MTU" (Maximum Transmission Unit), etc.

Moreover, RFC-8513 [16] suggested a solution for the DoS attack by raising the "b4-address-change-limit". This value specifies the minimum time between two consecutive changes in the IPv6 address of the B4 device. Setting it to a low value would enable the attacker to send a higher number of attacking packets with different source addresses. The recommended mitigation is to set its value to 30 min. In RFC-8513 [16], the authors presented the security analysis of their DS-Lite architecture, which emphasized that the main vulnerability is an attacker having access to either B4 or AFTR router and undertaking several kinds of malicious activities:

- Manipulating the AFTR IPv6 address on the B4 tunnel endpoint, which will deceive the B4 router and force it to forward the 4in6 traffic to the wrong recipient;
- Altering the value of the "b4-address-change-limit", gives the B4 more flexibility in configuring the softwire. An attacker lowering this value will boost the possibility of a DoS attack against the B4 router.

4.2.1 Tunneling Issues and Solutions

Since DS-Lite is made possible by constructing a tunnel between B4 and AFTR routers, a different set of challenges has emerged regarding the tunneling procedure. The most serious issue with tunneling is that tunneled IP traffic does not go through the same inspection process as normal traffic (non-encapsulated packets). This occurs unless extra dedicated devices are installed on the premises to perform deep packet inspection [46]. For instance, in the Teredo tunnel, the router will check the IP and UDP (User Datagram Protocol) layer normally. However, it cannot find out that there is actually another IP layer encapsulated within the UDP payload. Another issue with tunneling is when the already encapsulated packet is targeting a host that lies further beyond the tunnel endpoint. In this case, the machine will normally forward the packet simply to the built-in

next hop [46]. The tunneled data endpoints are aware of the tunneled data and the encapsulated packets. The network devices in the middle are not, and that could be an issue [46]. A solution is proposed in [46], wherever IPv6 transition is required, native IPv6 is the way forward in terms of connecting two sides with tunneling solutions, such as ISATAP, 6over4, etc. [17], in order to encapsulate traffic between a device and the router on the other side that resides in the same network. The difficulties that come with inspecting the inner content of the encapsulated packet make it a quite complicated process. NAT process also presents another challenge when it comes to tunneling, opening more doors for attacking possibilities on the incoming NAT interface. Therefore, it is recommended that the NAT interface should not be configured by default and only used when it is the last resort [46]. Another recommendation is to deactivate the interface itself after its usage [46]. Furthermore, IP address guessing emerged as a potential risk to the tunnel end-point, where some protocols use a regular or a well-known IP address or range of IP addresses. For instance, Teredo uses a specific IP address range in its infrastructure, which makes the tunnel liable to IP address guessing attacks. Furthermore, sometimes guessing the IP address gives an indication about which kind of OS (Operating System) is being used. For instance, Teredo implies that the machine is most probably running Microsoft Windows. The solution is to avoid those well-known IP address ranges and use random ones [46]. On the other hand, an adversary can have the ability to alter the tunnel's server settings on the client side while the client itself has no idea about it. One way to avoid such a breach is to use authentication for tunnel endpoints such as HTTPS (Hypertext Transfer Protocol Secure) [46]. A second mitigation method can be the use of secure ND (neighbor discovery) whenever a client receives a router advertisement packet [17]. Tunnels, in general, are less secure than normal conventional links due to the fact that an attacker can send an already encapsulated packet to the tunnel end-node where it is supposed to be decapsulated [46]. This action might cause an injection of the faulty packet into the decapsulation side. This threat might be avoided by turning on a decapsulation check, which will drop such malicious packets [67]; it is also highly discouraged to set the tunnel interface to reply by acknowledging the existence of a tunnel, such as an "ICMP (Internet Control Message Protocol) error message", representing another valuable recommendation.

4.3 Lightweight 4over6

Very little research has been published regarding lw4o6. Ahmed Al-hamadani proposed a test environment for benchmarking lw4o6 and especially its two main components (lwB4 and lwAFTR) [10]. The author carried out an analysis of the operational requirement to build such a tester, which aimed to be the world's first RFC-8219 [35] compliant lw4o6 tester. Omar D'yab built a test-bed for lw4o6 [24], where the author demonstrated the operation of lw4o6 with its encapsulation/decapsulation mechanism. However, the author didn't examine the security analysis of lw4o6 as it was not the main focus of his research.

As for the lwB4 router's implementation, Marcel Wiget [32] has built a complete and functioning lwB4 machine, where he used several Linux commands to build the NAT44 and IPv4-in-IPv6

tunnel. In addition, this proposed lwB4 network function is isolated into its own dedicated network namespace, which gives users the flexibility and the benefit of avoiding the use of a separate VM (virtual machine) [32]. Nevertheless, the later implementation didn't analyze the potential security vulnerabilities of the lwB4 or lw4o6 in general. Previous trials had been carried out to build lwB4 router using OpenWrt software [72]. However, they have proven to be complicated and not reliable [32].

4.4 MAP-E / MAP-T

In [52], the performance and scalability of the 464XLAT and MAP-T IPv6 transition technologies were tested and compared. As for MAPT-T, the authors built a test-bed on Debian-based machines, where they used Jool open-source software [41] to implement CE and BR routers. The authors tested the scalability of the performance of CE and BR routers while adding more CPU (Central Processing Unit) cores and monitoring the performance. They found out that the BR router scales better than the CE router, where a bottleneck is obvious at the CE side [52]. However, the focus of the authors was merely on performance and not on security, which is our current focus area. Another test environment for MAP-T infrastructure was built by several researchers in Brazil [20]. The research aimed to test the connectivity of some applications using the MAP-T translation mechanism. As for MAPT-T implementation, they used software developed by Cernet Center [18]. They used Fedora Linux-based virtual machines for the CE and BR routers. In addition, for the host machine (IPv4 client), they used three different machines (Linux Kubuntu, Windows 7 and Windows XP) to gather as many results as possible. Unfortunately, this test-bed also did not inspect nor analyse the security threats that MAP-T might face.

A MAP-T tester was designed by Al-Hamadani [11], where the author presented his progress towards building the world's first RFC-8219 compliant tester for MAP-T to record some measurements such as throughput and latency. For his MAP-T test-bed, the author used Jool software [41] to implement the CE and BR routers.

Furthermore, another experiment was performed by Georgescu [34], where the authors built a penetration test-bed for MAP-T and conducted multiple attacking scenarios such as ARP Cache Poisoning, Neighbour Advertisement Flooding and Traffic Analysis.

Chapter 5

Implementations

It is worth mentioning that the IP scheme that we used was based on documentation IP addresses because we built our topologies in a test environment that had internet access, while we did not want to cause any sort of routing conflict.

5.1 464XLAT Implementation

5.1.1 464XLAT Test-bed

The test-bed was built through remote access to a Windows-based virtual machine with the following specifications:

- Intel(R) Xeon(R) Silver 4215 CPU @ 2.50GHz, (16 VCPUs)
- 16.0 GB RAM
- 64-bit Windows 10 operating system.

Further virtualization was created by installing VMware Workstation 12 Player. Several virtual machines were created using Debian image, which was created by using the Debian-VM tool of Daniel Bakai. Every machine had Debian 8.9 operating system.

5.1.2 Test Topology

The topology of the 464XLAT test-bed is shown in Fig. 9, it can be divided into two sides:

- On the left side, there are eight clients (10.0.0.1 – 10.0.0.8) and the CLAT.
- On the right side, there are the PLAT and the IPv4 server.

5.1.3 Test-bed Implementation

In our test-bed, each virtual machine has the following specifications:

Table 6: Linux and VMware Network Setting For Virtual Machines

VM Name	Clients 1-8	CLAT	PLAT	IPv4 Server
eth0 Linux setting	DHCP	DHCP	DHCP	DHCP
eth1 Linux setting	Static IPv4: 10.0.0.1-8	Static IPv4: 10.0.0.11	Static IPv6: 2001:db8:2::2/64	Static IPv4: 198.51.100.2
eth2 Linux setting	N/A	Static IPv6: 2001:db8:2::1/64	Static IPv4: 198.51.100.1	N/A
eth0 VMware setting	NAT	NAT	NAT	NAT
eth1 VMware setting	VMnet11	VMnet11	VMnet12	VMnet13
eth2 VMware setting	N/A	VMnet12	VMnet13	N/A;

- Clients 1-8: 1GB RAM, 1 CPU core and 20 GB hard disk. CLAT: 1GB RAM, 3 CPU cores and 20 GB hard disk.
- PLAT: 1GB RAM, 3 CPU cores and 20 GB hard disk.
- IPv4 Server: 1GB of RAM, 1 CPU core and 20 GB hard disk.

Furthermore, the topology was built using three separated virtual networks: VMnet11, VMnet12 and VMnet13.

- VMnet11: the network between the eight clients and CLAT eth1 interface. The network is IPv4 only.
- VMnet12: the network between CLAT eth2 and PLAT eth1 interfaces. The network is IPv6 only.
- VMnet13: the network between PLAT eth2 and IPv4 server eth1 interfaces. The network is IPv4 only.

Table 6 shows the Linux and VMware settings used for each virtual machine.

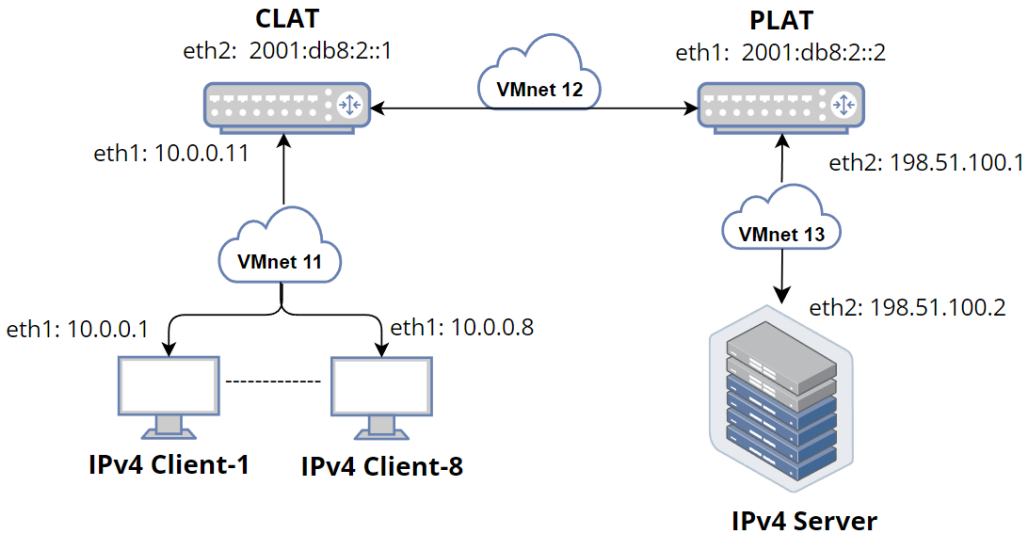

Figure 9: 464XLAT Test-bed

5.1.4 Test configuration

An open-source software called TAYGA was used to build NAT46 and NAT64 gateways [58].TAYGA, released under the GPLv2 license, is a free software that aims to deliver a high-quality NAT64 service. Its developers designed TAYGA as a stateless NAT64 solution specifically for Linux. In essence, TAYGA is capable of establishing direct correspondence between IPv6 and IPv4 addresses on a one-to-one basis. However, to achieve stateful NAT64 functionality, TAYGA is typically utilized in conjunction with a stateful NAT44 packet filter (specifically, `iptables` in the Linux environment). [54]

The main configuration blocks are CLAT (stateless translator) and PLAT (stateful translator). The details of their settings are presented below.

Configuring CLAT

It was set in the `/etc/default/tayga.conf` file that TAYGA should be started at operating system boot time:

```
RUN="yes"
CONFIGURE_NAT44="no"
```

Here, we did not intend to use TAYGA as a stateful NAT64 because CLAT is a stateless NAT46 translator. The operating parameters of TAYGA were set in the `/etc/tayga.conf` file as follows:

```
tun-device nat64
ipv4-addr 10.0.0.9
ipv6-addr 2001:db8:2::9
prefix 2001:db8:a::/96
map 10.0.0.1 2001:db8:c::10.0.0.1
map 10.0.0.2 2001:db8:c::10.0.0.2
map 10.0.0.3 2001:db8:c::10.0.0.3
map 10.0.0.4 2001:db8:c::10.0.0.4
map 10.0.0.5 2001:db8:c::10.0.0.5
map 10.0.0.6 2001:db8:c::10.0.0.6
map 10.0.0.7 2001:db8:c::10.0.0.7
map 10.0.0.8 2001:db8:c::10.0.0.8
```

As for further settings, the following bash shell script was responsible for setting up routes and enabling Linux kernel routing:

```
#!/bin/bash
ip route add 198.51.100.0/24 dev nat64
ip route add 2001:db8:c::/96 dev nat64
ip route add 2001:db8:a::/96 via 2001:db8:2::2
echo 1 > /proc/sys/net/ipv4/ip_forward
echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
ip route del 2001:db8:a::/96 dev nat64
```

We note that the last command was to delete a routing rule that was automatically set by TAYGA.

Configuring PLAT

It was set in the `/etc/default/tayga.conf` file that TAYGA should be started at operating system boot time:

```
RUN="yes"
CONFIGURE_NAT44="no"
```

Here, we used TAYGA as a stateful NAT64 translator, but we set the `iptables` rule manually (see below). The operating parameters of TAYGA were set in the `/etc/tayga.conf` file as follows:

```
tun-device nat64
ipv4-addr 198.51.100.9
ipv6-addr 2001:db8:2::9
prefix 2001:db8:a::/96
map 10.0.0.1 2001:db8:c::10.0.0.1
map 10.0.0.2 2001:db8:c::10.0.0.2
map 10.0.0.3 2001:db8:c::10.0.0.3
map 10.0.0.4 2001:db8:c::10.0.0.4
map 10.0.0.5 2001:db8:c::10.0.0.5
map 10.0.0.6 2001:db8:c::10.0.0.6
map 10.0.0.7 2001:db8:c::10.0.0.7
map 10.0.0.8 2001:db8:c::10.0.0.8
```

As for further settings, the following bash shell script was responsible for setting up routes and enabling Linux kernel routing:

```
#!/bin/bash
ip route add 10.0.0.0/24 dev nat64
ip route add 2001:db8:a::/96 via 2001:db8:2::1
echo 1 > /proc/sys/net/ipv4/ip_forward
echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
```

Furthermore, the below `iptables` command was applied:

```
iptables -t nat -A POSTROUTING -o eth2 -j MASQUERADE
```

The aim of this command is to perform a stateful NAT44 translation using the well-known Netfilter framework [26]. It was necessary, because TAYGA is only a stateless NAT64 translator by itself, and thus it requires an additional stateful NAT44 translator to implement stateful NAT64.

5.2 DS-Lite Implementation

5.2.1 Testbed Topology and Specifications

As shown in Fig. 10, the testbed consisted of five machines (IPv4 client, B4, AFTR, IPv4 server and the attacker). They were all based on VMware workstation VMs and built upon CentOS-7 images. Every machine had the following specifications:

- RAM: 3 GB;

- Hard disk: 20 GB;
- CPU: 1 core.

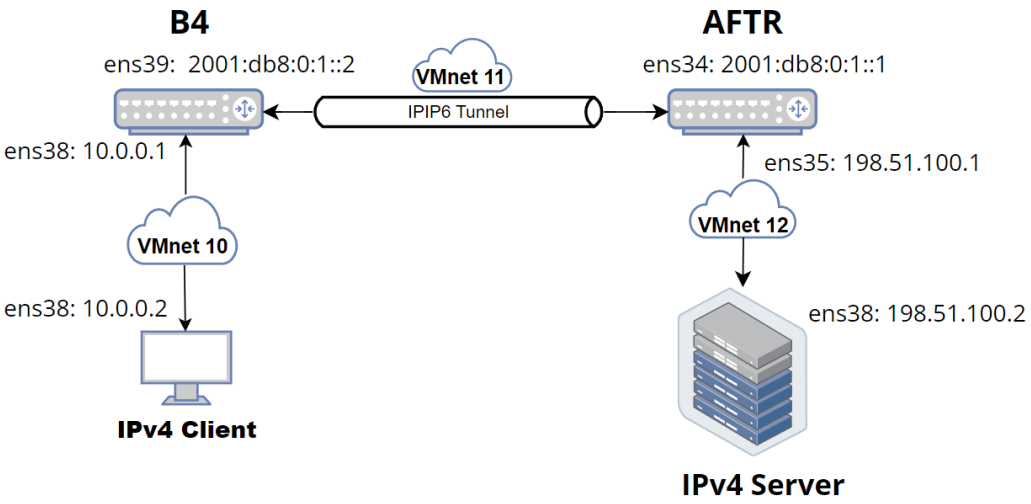

Figure 10: DS-Lite test-bed

The host PC that hosted the whole test-bed had the following specifications:

- OS: Windows 10 Pro;
- RAM: 16 GB;
- Hard disk: 1 TB;
- CPU: Intel-Core i7, 4 physical cores.

The IPIP6 tunnel was built between B4 and AFTR, which took care of encapsulating the IPv4 packet inside the IPv6 packet and then decapsulating it. For the NAT process on the AFTR side, the `iptables` rule was configured to masquerade the source IP address to the AFTR's network interface (ens35). As for the tunnel between B4 and AFTR, a shell script was executed on the B4 machine, which consists of the following commands:

```
ip link add name ipip6 type ip6tnl local 2001:db8:0:1::2 remote \
    2001:db8:0:1::1 mode any dev ens39
ip link set dev ipip6 up
ip route add 198.51.100.0/24 dev ipip6
```

It is noted that an important simplification is contained in the above setup compared to Table 1: the Softwire-ID is missing from the connection tracking table. It could be done without influencing the experiments because only a single client was used. This solution simplified the creation of the testbed significantly: `iptables` could be used instead of a real AFTR implementation.

5.3 Lw4o6 Implementation

5.3.1 LwB4 Implementation

The full bash script to configure the lwB4 router is available through the "`lwB4.sh`" script in our public GitHub repository [2]. The main commands that we used to create a tunnel for encapsulation/decapsulation plus NAPT44, were as follows:

```
ip -6 tunnel add lw4o6-tun remote 2001:db8:2::2 local 2001:db8:0:1::2 mode ipip6
ip route add 192.0.2.0/24 dev lw4o6-tun proto static
iptables -t nat -A POSTROUTING -p tcp  -o lw4o6-tun -j SNAT --to-source \
   203.0.113.1:1024-2047
iptables -t nat -A POSTROUTING -p udp  -o lw4o6-tun -j SNAT --to-source \
   203.0.113.1:1024-2047
iptables -t nat -A POSTROUTING -p icmp  -o lw4o6-tun -j SNAT --to-source \
   203.0.113.1:1024-2047
```

Below is an explanation for the IP addresses that we used in the commands above and illustrated in Fig. 11:

- 2001:db8:2::2 is the tunnel endpoint at lwAFTR side.
- 2001:db8:0:1::2 is the IPv6 address of lwB4.
- 192.0.2.0/24 is the network address of the IPv4 server.
- 203.0.113.1 is the public IPv4 address that will be used as source IP address by lwB4 when it forwards packets to lwAFTR through the softwire tunnel.

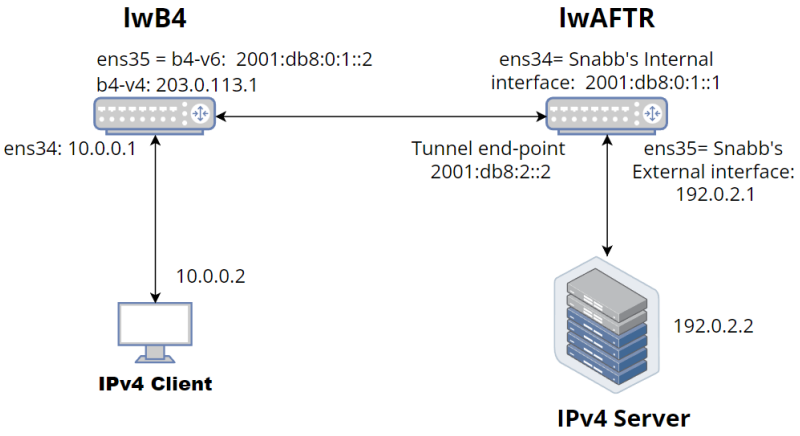

Figure 11: Lw4o6 Testbed Implementation using Snabb

5.3.2 LwAFTR Implementation

Several software solutions were presented to build lwAFTR routers. It was featured in VPP [27] since version v16.09. However, VPP has demonstrated complexity in its configuration, and certain modules such as lw4o6 have become outdated and lack proper maintenance from developers. In contrast, Snabb software has received better maintenance and documentation [31]. Therefore,

we have decided to deploy Snabb to build our lwAFTR router. Snabb in general is a toolkit that can be used for developing network functions in user-space, which means it bypasses the kernel to process network packets [31]. Therefore, Snabb leverages technologies like Intel's Data Plane Development Kit (DPDK) or the Solarflare OpenOnload library to directly access the NIC (Network Interface Card) and perform packet processing in user space. By bypassing the kernel, Snabb aims to achieve lower latency and higher throughput [31].

Snabb divides the machine into two separate spheres:

- Internal interface, where IPv6 packets are received and processed.
- External interface, where public IPv4 packets are forwarded to the outside world.

Snabb resides in between those interfaces and creates a binding table. The core of Snabb's configuration is a file called "lwaftr.conf", which can be used while running the "`snabb lwaftr run lwaftr.conf`" command. The full configuration script of the lwAFTR router is available through the "`lwaftr.conf`" file in our GitHub repository [2].

Table 7: LwAFTR Binding Table

Public IPv4	PSID	PSID length	b4-IPv6
203.0.113.1	1	6	2001:db8:0:1::2

On the lwAFTR router, Snabb has a pre-configured binding table (see Table 7), where it stores the relevant information for every lwB4 router (every softwire), such as lwB4's public IPv4 address, IPv6 address and the allocated port set for it.

The binding table is directly generated from the set of the configured softwires. It is never changed in response to data-plane traffic.

One of Snabb's (or lw4o6 in general) challenges is the liability of lwAFTR to have millions of softwires (tunnels) configured [30].

5.4 MAP-T Implementation

5.4.1 Jool Implementation

In the current research, we focus on the MAP-T implementation and leave MAP-E for future research opportunities. For example, Jool [41], which is an open-source IPv4/IPv6 translator, supports MAP-T and other solutions, but not MAP-E. On the other hand, VPP [27] does support (in theory) both MAP-E and MAP-T. However, it proved to be complicated to configure. As a result, we have chosen Jool as a candidate for our MAP-T implementation.

As shown in Fig. 12, MAP-T consists of two main routers (CE and BR). CE router consists of two separate name spaces (napt and global). The "napt" namespace represents the communication between the IPv4 client and the CE router, while the "global" namespace oversees the communication between the CE router and the outside world (the BR router). On the CE side, Jool

applies the NAT64 well-known prefix (64:ff9b::/96) to translate the IPv4 address into the IPv6 address by appending the IPv4 address to the end of it. For example, if the IPv4 destination address (for packets heading from the IPv4 client toward the IPv4 server) is 192.0.2.2, it will be translated into 64:ff9b::c000:202. The two namespaces are connected through static route Linux commands.

Within the "napt" namespace, a default route was set to direct all traffic via "192.168.0.1", which means that Jool forwards the packet to the other namespace of "global":

```
ip netns exec napt ip route add default via 192.168.0.1
```

Furthermore, any packet heads towards the well-known prefix "64:ff9b::/96" will be directed through the interface that has an IPv6 address of "2001:db8:6::1", which is the BR router interface IPv6 address:

```
ip route add 64:ff9b::/96 via 2001:db8:6::1
```

Finally, in our testbed, we used the documentation IPv4 "203.0.113.8" as the public IPv4 for the NAT44 function. Therefore, any packet with a destination IPv4 address of NAT44 public IP will be forwarded to "napt" namespace network:

```
ip route add 203.0.113.8/32 via 192.168.0.2
```

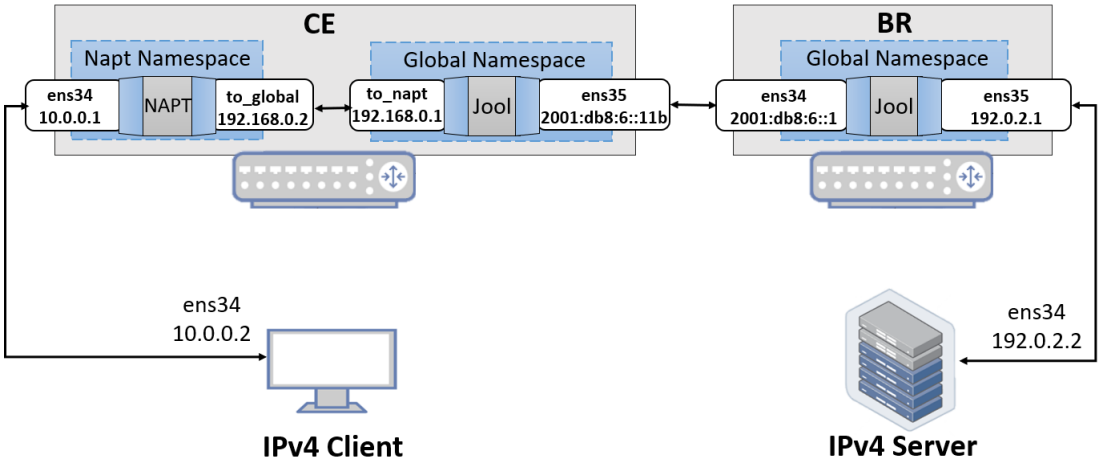

Figure 12: MAP-T Implementation using Jool Software

The IP address of "203.0.113.8" is the public IPv4 address of the NAT44 process within the CE router, it will be used as the source IPv4 address for all packets headed from CE towards the BR, before the IPv6 translation.

The full bash script to configure the CE router is available through the "`CE.sh`" script in our GitHub repository [3].

As for the BR, Fig. 12 shows that it has only one name-space and IPv6 address-equipped interface (ens34), where it receives the translated packet from the CE router and translates it back to IPv4 and forwards it to its final destination (IPv4 server in our example).

For BR to perform such actions, Jool applies another set of Linux commands. Most importantly, it sets an IP route for packets heading from the BR to the CE router using the End-User-IPv6 Prefix: "2001:db8:ce:11b::/64"

```
ip route add 2001:db8:ce:11b::/64 via 2001:db8:6::11b
```

Furthermore, the below commands were used to run Jool and to set the default mapping rule (DMR):

```
/sbin/modprobe jool_mapt
jool_mapt instance add "BR" --netfilter --dmr 64:ff9b::/96
jool_mapt -i "CE 11b" global update end-user-ipv6-prefix 2001:db8:ce:11b::/64
```

The full bash script to configure the BR router is also available through the "`BR.sh`" script in our GitHub repository [3]. Moreover, the installation of Jool software itself was made possible through an automated script "`jool.yml`" in the same repository.

5.4.2 MAP-T Test-Bed

To build our test-bed, we used a "P" series node of NICT StarBED, Japan [60], which is a Dell PowerEdge 430 server that has the following details: two 2.1GHz Intel Xeon E5-2683v4 CPUs with 16 cores each, 348GB 2400MHz DDR4 SDRAM. On this server, we have installed a Windows 10 Pro operating system.

As shown in Fig. 12, our test-bed consists mainly of 4 machines created with Linux-based Debian-10 Virtual machines built on top of VMware Workstation player virtualization software. Every machine has 8 GB RAM, 6 CPU cores and 20GB HDD. The full and detailed configurations were written and explained in an automated manner in our GitHub repository [3]. We have used an open-source IT automation tool called Ansible [39] to automate the installation process. In the repository, separate automation scripts were written and customized to configure the IPv4 client, IPv4 server, CE and BR router separately [3].

For instance, to configure the CE router, perform the following: -

- Clone the repository [3]
- Run this command: `ansible-playbook jool.yml`
- The user will be prompted to choose what to configure, CE or BR, for CE, type "e", while for BR, type "b".

The rest will be done by Ansible script, which will inspect the host machine, determine what Linux distribution it has, install all of the dependencies, deploy Jool software and finally run the Linux networking commands that we listed in Sub-section 5.4.1.

The same process applies to the IPv4 client and the IPv4 Server, which can be triggered by the following commands: `ansible-playbook clinet.yml` and `ansible-playbook server.yml`. Therefore, the IP configuration and routing commands will be configured by Ansible script.

5.4.3 Operation of the Testbed

While sending a crafted UDP packet from the IPv4 client (with a source port of 5000) to the IPv4 server (see Fig. 12), we monitored the traffic with the `tshark` command at different locations. As shown in Table 8, the NAPT44 + NAT46 translation process on the CE router's side and the stateless NAT64 on the BR router's side are quite obvious. UDP Packets were successfully translated by the CE and the BR routers as configured and no packet loss was reported.

As shown in Table 8, packets stemming from ens35 of the CE router (CE ⇒ BR) have the following source IPv6 address: `2001:db8:ce:11b:0:cb00:7108:1b`.

Table 8: MAP-T Packet Translation Process on CE and BR

Packet capture at IPv4 Client `ens34`
`1 0.000000000 10.0.0.2 > 192.0.2.2 UDP 42 Source port: 5000 Destination port: 80`
Packet capture at CE `to_napt`
`1 0.000000000 203.0.113.8 > 192.0.2.2 UDP 42 55398 → 80 Len=0`
Packet capture at CE `ens35`
`1 0.000000000 2001:db8:ce:11b:0:cb00:7108:1b > 64:ff9b::c000:202 UDP 62 55398 → 80 Len=0`
Packet capture at BR `ens35`
`1 0.000000000 203.0.113.8 > 192.0.2.2 UDP 42 55398 → 80 Len=0`

The above IP address consists of two separate segments:

- end-User-IPv6 Prefix (2001:db8:ce:11b), as configured in CE and BR,
- IPv6 Interface Identifier (0:cb00:7108:1b).

As illustrated in Fig. 13, the IPv6 Interface identifier consists of three main parts (padding, IPv4 address and PSID). As a result, applying the same structure on our IPv6 Interface Identifier (0:cb00:7108:1b) shows us the following:

- 0: padding on the left side.
- cb00:7108: hexadecimal version of the public IPv4 address (203.0.113.8), which is the allocated public IPv4 address for the CE router.
- 1b: the PSID value, which is "27" in decimal and corresponds to ports [57343 - 55296] that we assigned to our CE router.

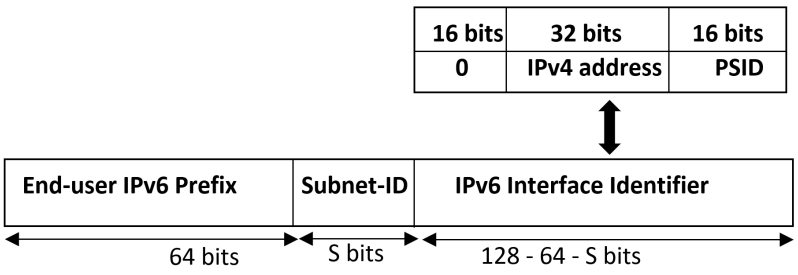

Figure 13: Derivation of MAP-T IPv6 address

The PSID value is derived and calculated based on the PSID length and the given EA (Embedded Address) bits, which should be pre-configured. In our Test-bed implementation, we passed PSID

length of 2048 and “13” to EA bits, which led to PSID = 27.

The derivation process of PSID value out of EA bits and PSID length is explained in RFC-7597, Section 5.2 [80]

Chapter 6

Security Analysis of 464XLAT

464XLAT is very important for network operators, especially ISPs. It is among the five most important IPv4aaS technologies that have to be supported by all customer edge routers [73]. It has several advantages like its port number efficiency and its widespread support in cellular networks [53]. Therefore, its security properties may be extremely important decision factors for network operators.

6.1 Applying STRIDE on 464XLAT

In order to apply the STRIDE method on any system, a DFD (Data Flow Diagram) must be drawn to highlight the locations of potential security threats. Therefore we present the DFD of 464XLAT in Fig. 14. Some security threats are visible at points (1-11), which represent the threat possibilities within the DFD.

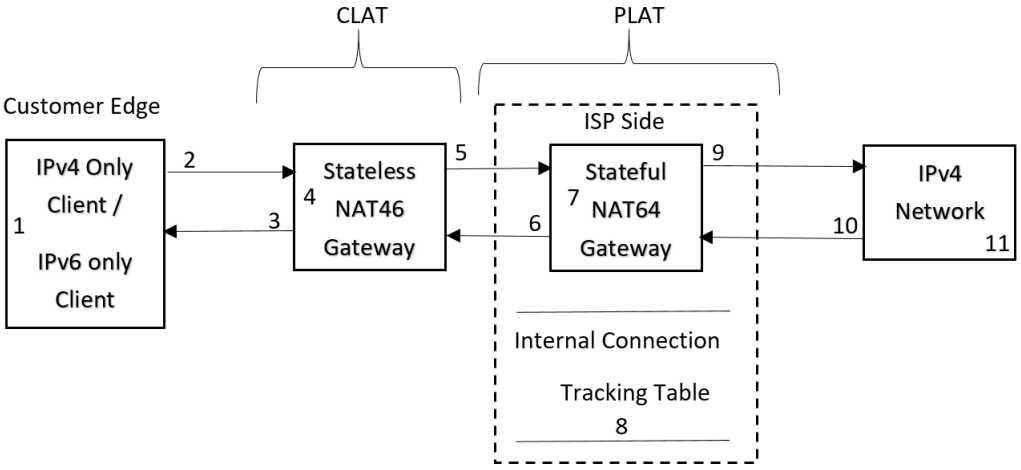

Figure 14: DFD for the Threat Analysis of 464XLAT

6.2 Attacking Possibilities

In this section, we carefully examine all the elements of the DFD for all possible threats & attacks in detail.

6.2.1 IPv4 / IPv6 Client

(i) Spoofing: an adversary spoofs the IP address of the IPv4 client and starts sending harmful packets to the CLAT router.

(ii) Repudiation: an attacker denies his responsibility for sending specific packet from the client machine. Therefore, the attacker can send packets with harmful content packet to the CLAT gateway and then disown the action.[78].

6.2.2 Data flow from IPv4 only client to the CLAT

(i) Tampering: an attacker monitors the communication channel between the client and the CLAT machine and tampers with the exchanged packets such as the payload content or destination IP address, which might be used to direct the packet toward a fraudulent server. This kind of attack is also called FoS (Failure of Service) because it prevents the real client from receiving an answer to its real query [50].

(ii) Information Disclosure: an attacker might be interested in knowing the browsing habits of the requester, and the packet itself might contain some sensitive information sent by the client himself [50].

(iii) DoS: and attacker floods the CLAT gateway with unwanted useless queries to prevent the real queries from getting an answer.

6.2.3 Data flow from the CLAT gateway to the client

(i) Tampering: an attacker sniffs the communication channel between the CLAT and the client then injects a malicious packet such as TCP packet with RST flag to terminate the TCP connection [78].

(ii) Information Disclosure: an attacker getting access to sensitive data.

(iii) DoS: an attacker sends a high number of forged replies to the client to prevent it from processing the genuine ones.

6.2.4 NAT46 Gateway (CLAT)

(i) Spoofing: in this case, spoofing means an attacker impersonates the CLAT gateway and starts to communicate with the client and the PLAT by sending different sort of malicious packets.

(ii) Tampering: an attacker tampers with the data within the gateway itself which might result in e.g. returning the wrong IPv6 address [50].

(iii) Repudiation: after spoofing the CLAT, an attacker might deny sending a packet that was actually sent by him. Logging is the key here, if the database administrator is not fully trusted, then a system in another privileged domain has to be installed.

(iv) Information Disclosure: an attacker might make use of the browsing data and queries made by the requester in order to hack the main requester later on.

(v) DoS: an attacker floods the CLAT with a huge number of requests, please refer to Sub-Section 6.2.2.(iii).

(vi) Elevation of Privilege: an attacker gains access to a service he should not be getting in the first place, such as the right of admin or root access to a server. However, it mainly happens due to an inside job [50].

6.2.5 Data flow from the CLAT to the PLAT

(i) Tampering: an attacker monitors the traffic and alters the packet destination IP which might direct the packet to a malicious server.

(ii) Information Disclosure: please refer to Sub-Section 6.2.2.(ii).

(iii) DoS: an attacker might send numerous useless packets to the PLAT in order to exhaust its processing capabilities.

6.2.6 Data flow from the PLAT to the CLAT

(i) Tampering: please refer to Sub-Section 6.2.2.(i).

(ii) Information Disclosure: an attacker accesses the packet details on its way back to the NAT46 gateway and extracts sensitive information from it.

(iii) DoS: an attacker floods the CLAT with unwanted packets to prevent it from translating the genuine traffic or to exhaust the capacity of its connection tracking table or to exhaust its available source port number pool (public IPv4 address pool)..

6.2.7 NAT64 Gateway (PLAT)

(i) Spoofing: an attacker impersonates the PLAT gateway and starts to communicate with the CLAT or with the IPv4 server, which will endanger every network element with the risk of exchanging data with a malicious entity, please refer to Sub-Section 6.2.4.(i).

(ii) Tampering: an attacker alters the content of a packet while being processed within the PLAT, please refer to Sub-Section 6.2.4.(ii).

(iii) Repudiation: please refer to Sub-Section 6.2.4.(iii)..

(iv) Information Disclosure: please refer to Sub-Section 6.2.4.(iv).

(v) DoS: an attacker (e.g., by tampering with the data flow from the CLAT to the PLAT) sends a huge amount of useless queries to the PLAT in order to exhaust its pool of source ports or public IPv4 addresses , which is an issue since the gateway uses 63K number of source ports per public IPv4 address. An enhanced algorithm presented by [28] helps in tackling this issue in detail.

(vi) Elevation of Privilege: one of the elevation problems is called buffer overflow attack [45], which could happen if a device like the PLAT gets inputs from both sides, which might affect its memory storage units.

6.2.8 Internal connection tracking table of the PLAT

(i) Potential attackers have no direct access to it, they can influence its content in indirect ways only.

(ii) DoS: the attacker may initiate fake connections (either using his real IPv6 address or fake ones) and thus achieve the insertion of fake entries into the connection tracking table. The high number of fake entries may slow down the operation of the PLAT or even prevent legitimate users from establishing further connections when the table is full. If the PLAT applies a connection limit per source IPv6 address, then the attacker may exhaust the available number of connections for legitimate users by spoofing their IPv6 addresses and initiating fake connections.

6.2.9 Data flow from PLAT to IPv4 Server

(i) Tampering: an attacker might intercept the exchanged data and alter the source IP address or the payload of the packet to deceive the IPv4 server.

(ii) Information Disclosure: please refer to Sub-Section 6.2.2.(ii).

(iii) DoS: an attacker floods the IPv4 server with plenty of undesired requests to exhaust its processing power.

6.2.10 Data flow from the IPv4 server / IPv4 network to the PLAT

(i) Tampering: an attacker might send a TCP packet with RST flag-activated, which will erase the mapped entries within the PLAT.

(ii) Information Disclosure: it is possible that an attacker might access the packet details on its way back to the PLAT and extract sensitive information out of it, like TTL value or browsing habits [50].

(iii) DoS: an attacker can flood the PLAT with an extremely high rate of unwanted requests to prevent it from translating the real traffic.

6.2.11 IPV4 server / IPv4 network

(i) Spoofing: please refer to Sub-Section 6.2.1.(i).

(ii) Repudiation: an attacker spoofs the IP address of the IPv4 server, sends a packet with suspicious payload then denies his action, please refer to Sub-Section.6.2.1.(ii).

6.3 Results Summary

In Table 9, we summarized the attacks or the vulnerabilities within the 464XLAT structure.

Table 9: Summary of 464XLAT Threats

DFD Element	Threat	Possible Attacks
1	Spoofing & Repudiation	DoS attack against the CLAT
2, 3	Tampering, Information Disclosure and Dos	FoS, collecting unauthorized information, DoS
4	All STRIDE Elements	FoS, DoS and unauthorized access
5, 6	Tampering, Information Disclosure and Dos	FoS, collecting unauthorized information and Dos
7	All STRIDE Elements	FoS, DoS and unauthorized access
8	Only indirect attacks	Tampering with Connection Tracking Table and DoS
9, 10	Tampering, Information Disclosure and DoS	FoS, collecting unauthorized information, DoS
11	Spoofing & Repudiation	DoS attack against the PLAT

6.4 DoS Attack Scenario

The aim of the DoS attack is to exhaust the resources of the PLAT device. It is carried out by sending high-frequency ping requests from the eight clients to the IPv4 server using the `hping3` command, please refer to Fig. 9 in Sub-Section 5.1.2.

To carry out the attack, SSH (Secure Shell) authentication was established between client one (10.0.0.1) and the rest of the network elements in order to be able to login to all machines and carry out the experiment automatically. The script was responsible for starting traffic capture by using `tshark` package and for starting the attacking program on all clients at (mostly) the same time by a shell script.

The attack was through a `hping3` command from all 8 clients with specific arguments:

```
hping3 -S -p80 -s5000 -k 198.51.100.2 -iu1500
```

- S: TCP SYNC attack,
- p: destination port number,
- s: source port number,
- k: to maintain the same port number and avoid its increment,
- iu: to control the number of sent packets per second.

We note that the last parameter does not directly set the packet rate, but it specifies a targeted delay in microseconds. Different packet rates can be applied by manipulating the "`-iu`" argument of the `hping3` command (e.g. 100 packets/s, 1000 packets/s, etc.). The experiments were carried out using various packet rates. For the demonstration of the DoS attack, we selected a rate (about 460 packets/s) at which almost all packets were correctly translated by the PLAT, when only a single client was used. However, a significant amount of the frames were not correctly translated, when 8 attacking clients were used. Please see a fragment of the `tshark` output in Fig. 15. The incorrectly translated frame is highlighted by a red oval.

```
158   0.081455 198.51.100.1 -> 198.51.100.2 TCP 54 [TCP Port numbers reused] 5000 ^ ^ 80 [SYN] Seq=0 Win=512 Len=0
159   0.081510    10.0.0.2 -> 198.51.100.2 TCP 54 [TCP Port numbers reused] 5000 ^ ^ 80 [SYN] Seq=0 Win=512 Len=0
160   0.081953 198.51.100.1 -> 198.51.100.2 TCP 54 [TCP Port numbers reused] 5000 ^ ^ 80 [SYN] Seq=0 Win=512 Len=0
161   0.082078 198.51.100.2 -> 198.51.100.1 TCP 60 [TCP ACKed unseen segment] 80 ^ ^ 5000 [RST, ACK] Seq=1 Ack=1329834715 Win=0 Len=0
162   0.082130 198.51.100.2 -> 198.51.100.1 TCP 60 [TCP ACKed unseen segment] 80 ^ ^ 5000 [RST, ACK] Seq=1 Ack=579732553 Win=0 Len=0
163   0.082666 198.51.100.2 -> 198.51.100.1 TCP 60 80 ^ ^ 5000 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
```

Figure 15: PLAT eth2 tshark capture

The attack process was as below:

- Start the measurements by monitoring the traffic in the PLAT side at eth2 interface, then wait for 10 seconds to monitor the performance before the attack.
- Start the attack with client 1, then wait for 100 seconds to monitor the effect of one client.
- Start the attack with client 2, then wait for 100 seconds.
- Start the attack with client 3 & client 4, then wait for 100 seconds.
- Start attack with clients 5, 6, 7, 8, then wait for 100 seconds.
- End the measurements, then stop the eight attacks.

6.5 Results and Analysis

Fig. 16 shows the number of correctly translated, not translated and all frames per second as a function of time in the case of 460 packets/s rate. It is fairly obvious that the number of sent packets is increasing by doubling the number of clients (1, 2, 4, 8).

Every spike in the graph represents newly added client(s).

From 0-100 seconds (only client1).
From 100-200 seconds (client 1& 2).
From 200-300 seconds (client 1,2,3 and 4).
From 300-400 second (client 1,2,3,4,5,6,7 and 8).

The number of packets arriving at the PLAT is slightly fewer than the ones sent from the CLAT. The average value of sent packets when 8 clients were applied was around 3500 packets /s. Furthermore, the orange line represents the number of untranslated packets by `iptables`.

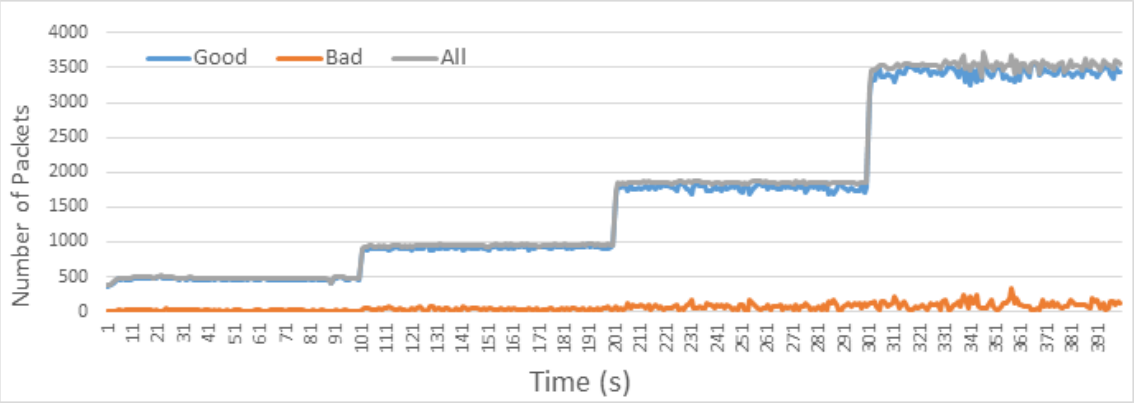

Figure 16: Measurements with 460 packets/s per client load: the number of good/bad/all packets as a function of time (number of attacking clients: 1, 2, 4 and 8)

We divided the packets into three types (good, bad, all). "Good" label represents the properly translated packets, "Bad" label represents packets that failed in the masquerading process and kept their original source IP address. As for "All" label, it is fairly obvious that it represents the total number of good and bad packets together.

The MASQUERADING feature is supposed to change the source IP address of packets leaving the PLAT and heading toward the IPv4 server. That means every packet heading towards the IPv4 server should have the source IP address of the PLAT eth2 interface (198.51.100.1). However, some packets are passing through the filter without getting translated by keeping their original IP address (10.0.0.1-8) instead of the PLAT eth2 interface IP address (198.51.100.1) as shown in Fig. 15. We don't know the exact root cause of this behavior. What we know is the frequency of these untranslated packets increases by increasing the number of applied clients as illustrated by Fig. 16.

6.6 DoS Attack Mitigation

We successfully countered the attack by implementing a defense strategy on the CLAT machine using `iptables` rules. This strategy involved the imposition of a strict rate-limiting mechanism, which was achieved through the creation of a filter containing two key rules. The first rule was configured to permit the forwarding of packets within specific constraints, such as a maximum of 10 packets per second. Simultaneously, the second rule was configured to unequivocally reject any packet that did not conform to the defined limits. These measures proved highly effective in safeguarding our system from malicious intrusion.

```
iptables -A FORWARD -p tcp --syn -m limit --limit 10/s -j ACCEPT
iptables -A FORWARD -p tcp --syn -j DROP
```

6.7 Conclusion

Threat analysis of the 464XLAT IPv6 transition technology was performed by applying the STRIDE method in order to point out the potential vulnerabilities of the technology. This method of using the data flow diagram of the 464XLAT system to analyze its potential security vulnerabilities has proven its efficiency.

The double translation mechanism of 464XLAT proved its effectiveness in terms of IPv4 literals communications over IPv6 infrastructure. However, it has some security issues and vulnerabilities such as DoS attack possibility. Some faulty translated packets were monitored and their percentages increased by adding more load to the topology and therefore affecting the PLAT performance. Some readings have visible fluctuation and this was mainly caused by the instability of the `hping3` command and its packet rate controlling ratio. In general, the experiment proved that 464XLAT is an effective transition technology to establish a stable connection over different IP versions' infrastructure.

Chapter 7

Security Analysis of DS-Lite

7.1 Applying STRIDE on DS-Lite

Before applying the STRIDE method, we built the DFD of the examined system to specify the potential attacking points at each element. Fig. 17 shows the spots (1–11). The security analysis is divided into several groups such as the traffic between the client and B4, then between B4 and the AFTR, etc.

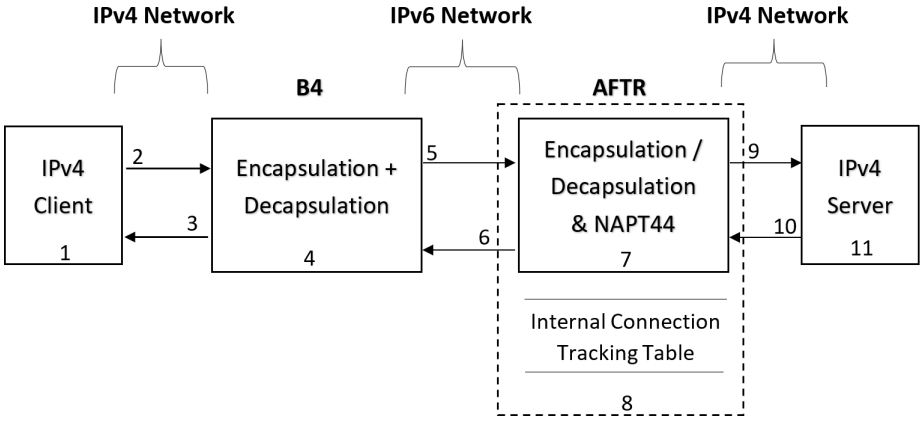

Figure 17: DFD of DS-Lite

7.2 Attacking Possibilities

7.2.1 IPv4 Client

(i) Spoofing: a malicious user might spoof the IP address of the client and attack the B4 router by sending many useless requests to the B4 and fully utilizing its computation power;

(ii) Repudiation: an attacker (after spoofing the client's IP address) denies the fact that he made the request in the first place. By doing so, a malicious user might attack the B4 router and then deny the fact that he was the one who sent this specific request such as an echo request,

DNS resolution request, etc.

7.2.2 Data Flow from the Client to the B4 Router

(i) Tampering: an adversary might tamper (change or modify) the content of the flowing packets such as IP address, source port number, etc. The attacker might also diverge the packet toward a fraudulent server which might cause an FoS attack, which is the prevention of the legitimate requester from receiving a response [50];

(ii) Information Disclosure: an attacker might get hold of confidential information such as some sensitive information sent by the client as plain text, the TTL (Time To Live) value, or the browsing habits of the client [50];

(iii) Dos: an attacker sends too many useless requests to the B4 router and blocks the legitimate requester from having his query processed.

7.2.3 Data Flow from B4 Router to IPv4 Client

(i) Tampering: in this case, the potential victim is the client, who is vulnerable to an attack at the application level, such as sending a TCP RST signal that will end the already established TCP connection [5] and also a de-authentication attack [47] that results in disconnecting the device from the router's WIFI (Wireless Fidelity), in addition to a plain-text injection attack [47] that allows the B4 router to send misleading information to the client, such as false routing details;

(ii) Information Disclosure: an attacker gaining access to sensitive information in a malicious way, such as an important text sent back by the IPv4 server behind the AFTR;

(iii) DoS: sending a high number of forged replies to overwhelm the client and prevent it from sending more requests to the B4.

7.2.4 B4 Router

(i) Spoofing: another machine presents itself as the legitimate B4 router and starts communicating with the rest of the DS-Lite network elements, which puts everyone that deals with this machine at high risk, such as a malicious device initiating a communication with the AFTR router and presenting itself as the B4 router, which leads to whole traffic (from AFTR side) pouring in the wrong direction by changing the destination IP address of the packet, which results in the original sender not receiving a response to his request. Another consequence could be when the malicious server starts recording the data passing through and using it for illegal activities such as blackmailing;

(ii) Tampering: modifying the current source and destination address of the 4in6 traffic that is being processed within the B4 router in order to shift the traffic and prevent it from being sent to the real AFTR, forwarding it to a malicious server. Another possibility could be

tampering with the IP address/port pair of the sender, which eventually results in a different entry in the AFTR translation table, and therefore, the packet reply will not return to the original sender;

(iii) Repudiation: in this case, when the B4 is spoofed, the malicious user will deny the fact that he sent a specific packet even though he did. The obvious solution to this issue is proper logging [5];

(iv) Information Disclosure: an attacker gaining access to sensitive data within the router such as the client's source IP address, TTL value, browsing data, etc.;

(v) DoS: after spoofing the B4 router, an attacker can flood the AFTR router with unnecessary requests in order to overwhelm the AFTR and stop it from processing any further incoming packets;

(vi) Elevation of Privilege: the attacker gains access to high-level (privileged) data, which mainly occurs due to an inside job [50], when an employee is implicit in the act.

7.2.5 Data Flow from the B4 to the AFTR

(i) Tampering: 4in6 traffic might be altered and the source IP address or port number is liable to be modified by the attacker;

(ii) Information Disclosure: the access to confidential information in many ways, please refer to Section 7.2.2.(ii);

(iii) DoS: the flood of useless queries exhausts the resources of the AFTR and hinders its main job, please refer to Section 7.2.2.(iii).

7.2.6 Data Flow from the AFTR to the B4

(i) Tampering: modifying the traffic details before it reaches the B4 router, such as the source IP address or port number of the 4in6 flowing traffic;

(ii) Information Disclosure: the unauthorized access to sensitive data, please refer to Section 7.2.2.(ii);

(iii) DoS: flooding the B4 router with useless traffic, please refer to Section 7.2.2.(iii).

7.2.7 AFTR Router

(i) Spoofing: an attacker impersonates the AFTR and starts communicating with the rest of the machines around him, such as the B4 router and IPv4 server, which might lead to exchanging sensitive data with the wrong person, please refer to Section 7.2.4.(i);

(ii) Tampering: an attacker sniffs on the communication channel and changes the content of the 4in6 packet payload, please refer to Section 7.2.4.(ii);

(iii) Repudiation: hiding the packet sender identity, please refer to Section 7.2.4.(iii);

(iv) Information Disclosure: the possibility of an attacker gaining access to confidential data, please refer to Section 7.2.4.(iv);

(v) DoS: an attacker tamper with the data flow from the B4 to the AFTR to exhaust the AFTR with too many useless requests and hinder the process of encapsulating the desired packet;

(vi) Elevation of Privilege: the act of unauthorized access to a very confidential data center of a specific folder in a malicious way, please refer to Section 7.2.4.(vi).

7.2.8 Internal Connection Tracking Table of the AFTR

(i) Tampering: an attacker can manipulate the connection tracking table of the AFTR router that saves the incoming 4in6 traffic packet with its source address and port number and maps it with a destination IPv4 device according to the stateful NAT rule within the router (see Table 1). The manipulation can cause a faulty destination address and disturb the encapsulation process and result in a packet loss;

(ii) DoS: flooding the connection tracking table with too many unnecessary entries (false connections) might overwhelm the table beyond its capabilities and cause it to flush its data or saturate the AFTR itself, which means the AFTR will no longer be able to process any incoming packet, or it will at least lose some legitimate connections.

7.2.9 Data Flow from AFTR to IPv4 Server

(i) Tampering: The destination IP of the traffic might be altered, causing the packet to be rerouted to a malicious server instead of its intended destination, the legitimate IPv4 server;

(ii) Information Disclosure: unauthorized access to sensitive data of the IPv4 traffic;

(iii) DoS: overwhelming the host with too many requests, please refer to Section 7.2.3.(iii).

7.2.10 Data Flow from IPv4 Server to AFTR

(i) Tampering: the IPv4 packets might be altered in terms of the IP source address or port number, which will definitely change the corresponding 4in6 traffic details;

(ii) Information Disclosure: unauthorized access to IPv4 traffic before it reaches the AFTR router, which might leak the confidential browsing habits of the client;

(iii) DoS: exhausting the AFTR and its main encapsulation function, please refer to Section 7.2.2.(iii).

7.2.11 IPv4 Server

(i) Spoofing: an attacker impersonates the IPv4 server and initiates communication with AFTR and the rest of the DS-Lite topology;

(ii) Repudiation: the host (IPv4 sever) might hide his identity, perform all sorts of malicious acts and then deny any responsibility for them. For example, in the previous point of spoofing, the attacker might spoof the IPv4 server, perform an attack against AFTR and then deny the fact that he initiated the communication in the first place.

7.3 Attacking Scenarios

7.3.1 Spoofing Attack from within VMnet-11

The proposed attack shown in Fig. 18 is spoofing the ingress tunnel endpoint (ens39), which is the softwire tunnel interface of the B4 router, and then sniffing/logging traffic or performing MITM attack against the egress tunnel endpoint (AFTR). The attack was based on the "`thc-ipv6`" toolkit [81], which has several tools within it. One powerful tool was chosen to perform the attack (four2six). It manually creates a crafted packet of IPv4 inside IPv6 (encapsulated packet) and sends it over the channel to the victim (AFTR).

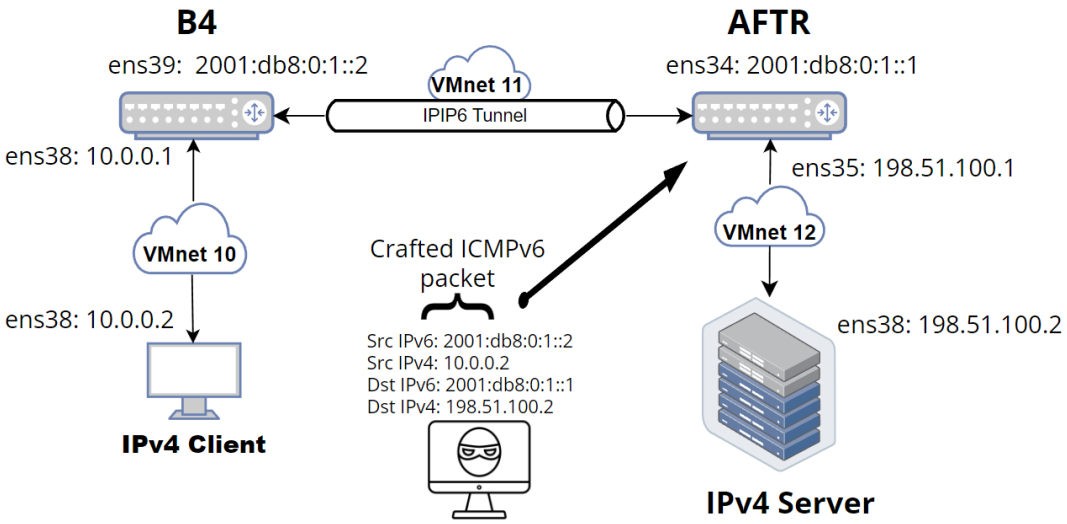

Figure 18: DS-Lite Attack Scenario-1

This crafted packet is actually a spoofed one, imitating that it was sent by the B4 router, having the same specifications as a genuine B4-originated packet:

- IPv6 Src: 2001:db8:0:1::2;
- IPv6 Dst: 2001:db8:0:1::1;
- IPv4 Src: 10.0.0.2;
- IPv4 Dst: 198.51.100.2.

In fact, the spoofed attack went through AFTR and then to the end receiver (IPv4 server). A series of ICMP echo requests and replies were exchanged between the attacker and the victim. In more detail, what happened is the following:

- The AFTR received the malicious packet;
- The AFTR saw the packet as a 4in6 packet, and therefore decapsulated the packet;
- The AFTR then forwarded the extracted IPv4 packet with the source address of 198.51.100.1 towards the IPv4 server (198.51.100.2);
- IPv4 server sent ICMPv4 echo reply back;
- The AFTR saw the later packet and encapsulated it in an IPv6 packet;
- The echo reply from the AFTR, which is an IPv6 packet with an IPv4 packet in its payload, was sent back to its destination IPv6 of address 2001:db8:0:1::2;
- Table. 10 shows three different terminals: the top one is the incoming packets at the attacker interface, the one in the middle is the B4 incoming traffic from the AFTR side, and at the bottom is the incoming traffic at the IPv4 client;
- B4 received the IPv6 packet carrying an IPv4 packet with ICMP echo reply because it is the legitimate owner of the 2001:db8:0:1::1 IPv6 destination address;
- The attacker saw the same reply because its NIC (Network Interface Controller) runs in promiscuous mode;
- The interesting thing here is illustrated by the IPv4 client receiving an ICMP4 echo reply for a request that he did not request in the first place, which was issued by the attacker initially;
- Echo replies in Table. 10 are highlighted in red.

Table 10: Spoofed packet echo reply

Packet capture at Attacker machine ens38
[root@Attackers]# tcpdump -n -i ens38
10:46:19.849015 IP6 2001:db8:0:1::22 > ff02::1:ff00:1: ICMP6, neighbor solicitation, who has 2001:db8:0:1::1 length 32
10:46:19.853245 IP6 2001:db8:0:1::1 > 2001:db8:0:1::22: ICMP6, neighbor advertisement, tgt is 2001:db8:0:1::1 length 32
10:46:19.858023 IP6 2001:db8:0:1::2 > 2001:db8:0:1::1: IP 10.0.0.2 > 198.51.100.2: ICMP echo request, id 16916, seq 772, length 16
10:46:19.860357 IP6 2001:db8:0:1::1 > 2001:db8:0:1::2: IP 198.51.100.2 > 10.0.0.2: ICMP echo reply, id 16916, seq 772, length 16
Packet capture at B4 machine ens39
[root@B4]# tcpdump -n -i ens39
10:46:19.849015 IP6 2001:db8:0:1::22 > ff02::1:ff00:1: ICMP6, neighbor solicitation, who has 2001:db8:0:1::1 length 32
10:46:19.853245 IP6 2001:db8:0:1::1 > 2001:db8:0:1::22: ICMP6, neighbor advertisement, tgt is 2001:db8:0:1::1 length 32
10:46:19.858023 IP6 2001:db8:0:1::2 > 2001:db8:0:1::1: IP 10.0.0.2 > 198.51.100.2: ICMP echo request, id 16916, seq 772, length 16
10:46:19.870357 IP6 2001:db8:0:1::1 > 2001:db8:0:1::2: IP 198.51.100.2 > 10.0.0.2: ICMP echo reply, id 16916, seq 772, length 16
Packet capture at IPv4 Client machine ens38
[root@ipv4-clinet]# tcpdump -n -i ens38
10:46:19.860364 IP 198.51.100.2 > 10.0.0.2: ICMP echo reply, id 16916, seq 772, length 16

7.3.2 Attack from within VMnet-10 (Compromised B4 Network)

It is worth mentioning that B4 by design is also capable of routing native IPv6 packets, which takes us to Fig. 19, where the IPv6 address was added to interface ens38 at the B4 machine in order to process and forward incoming IPv6 packets.

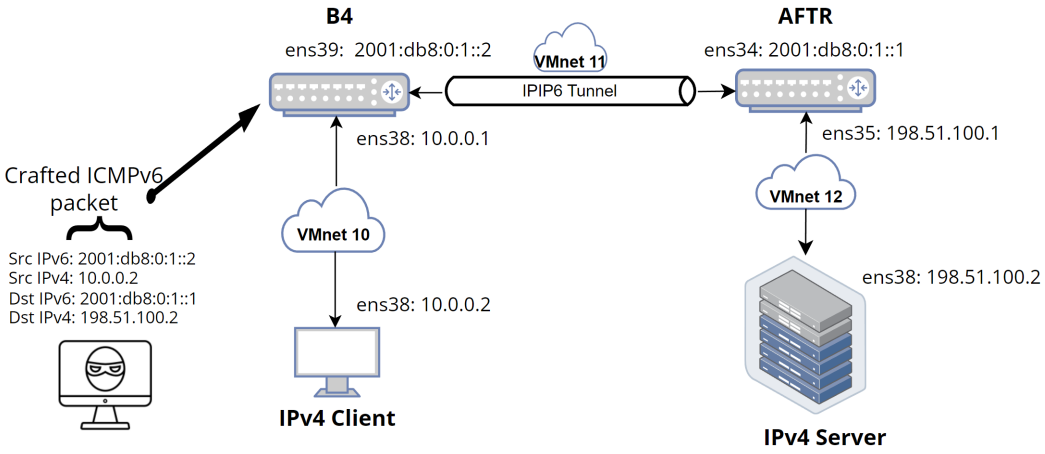

Figure 19: DS-Lite test-bed (attack Scenario 2)

A real-life spoofing scenario could also be stopped before reaching the internal network of AFTR (VMnet-11 in the current example). Therefore, another attacking scenario was proposed in Fig. 19, where the attacker is not sitting directly in the AFTR network, but outside it. The idea behind such an attack is that an attacker has access to the home router network, which gives him direct contact with the CPE. This access allows the attacker to send a crafted packet with a spoofed IPv6 address of a B4 router or any other IPv6 address from the internal network of AFTR (VMnet-11 in the current example). Fortunately for the user, the attack did not go through—the B4 router was always dropping any incoming packet from the attacker-2 machine that has an IP address (a spoofed one) from the pool of (VMnet-11). The reason for that is simple: there was no route configured on the B4 router to forward the crafted IPv6 packet to the AFTR router. The tunnel was hardcoded in the B4 machine by creating an IPIP6 tunnel interface to forward any IPv4 packet (heading towards 198.51.100.0/24 network) to the IPIP6 tunnel interface which has been configured. However, the attacker packet has no route to the tunnel interface, and that is why it was dropped.

7.3.3 Source Port Exhaustion Attack from within VMnet-10

At first, the attack was conducted using the same platform, a regular specs Windows-based computer; however, the results were not stable and not reproducible due to limited cores on the host PC. Therefore, the decision was made to use the resources of NICT StarBED, Japan. A “P” series node [60] was used, which is a Dell PowerEdge 430 server with the following specifications: two Intel(R) Xeon(R) CPU E5-2683 v4 @ 2.1 GHz CPUs having 16 cores each and 348 GB 2400MHz DDR4 SDRAM. Windows 10 Pro operating system was installed, and the same process was repeated by building the test-bed using a VMware workstation player and CentOS-7-based virtual machines. The specifications for each machine were as follows:

- RAM: 4 GB;
- Cores: 4 cores, except for AFTR having 6 cores;
- Disk: 20 GB.

Before disclosing the attack details, it is important to explain the mathematical sense behind it. In the networking world, ports are counted based on a 16-bit standard. The 16 bits were chosen when the TCP and UDP standards were designed [75]. As a result, the maximum number of ports equals $2^{16} = 65,536$. This number applies to TCP and UDP ports. In this experiment, the focus was on the UDP ports and how to exhaust them by sending too many queries (UDP requests) in a short period of time. However, not all ports in the range are usable for establishing new connections, because ports between 1 and 1024 are called well-known ports and they are reserved for specific functions such as FTP, HTTP, DNS, etc.

Therefore, the total range of ports that the NAPT (Network Address and Port Translation) device can use as UDP source port numbers equals 65,536 - 1024 = 64,512 ports.

Fig. 20 shows attacking Scenario 3 of sending too many AAAA queries from "IPv4 Client-1" and "IPv4 Client-2" machines. In order for the attack to function, a few files must be correctly configured:

- `/sys/module/nf_conntrack/parameters/hashsize`: The hash size is actually the size of the hash table storing the lists of conntrack entries [62], is better as a base of 2, which means that hash size could be $2^{14} = 16,384$;
- `/proc/sys/net/netfilter/nf_conntrack_max`: It represents the maximum number of allowed conntrack entries that netfilter can keep running simultaneously. It was set to hashize×8 = 16,384×8 = 131,072. This number will have a significant impact when the attack is run later so that the NAT table does not get full easily;
- `/proc/sys/net/netfilter/nf_conntrack_udp_timeout`: This timeout means that after 30 s, the already used UDP ports are ready to be re-assigned by the kernel, and the default value of 30 s was left as it is;
- `/etc/security/limits.conf`: This file controls the number of maximum open files at the same time, where the below line was added:

  ```
  root hard nofile 1000000
  ```

To perform this kind of attack, another tool was used, where a huge number of DNS queries can be sent to the target machine (AFTR), which is called "dns64perf++" [14] (described in [49]). When all parameters (on the attacking machine side) are set correctly (number of requests, the delay between requests, etc.), the AFTR pool of source ports will be exhausted, and the machine will not be able to process incoming packets anymore until the UDP timeout (port allocation timeout in AFTR machine) is up. The below command was used on each of the attacking clients:

```
/dns64perf++v4 198.51.100.2 53 0.0.0.0/5 60000 1 1 60000 400000 0.1
```

This command sends 60,000 queries from each client. For two clients, this number reaches 120,000 queries, which is still under 131,072 (`nf_conntrack_max`). As a result, the NAT table will not be overfilled with packets that exceed its capacity. Another interesting fact in the command is the delay of 400,000 nanoseconds between queries, resulting in 2500 queries per second,

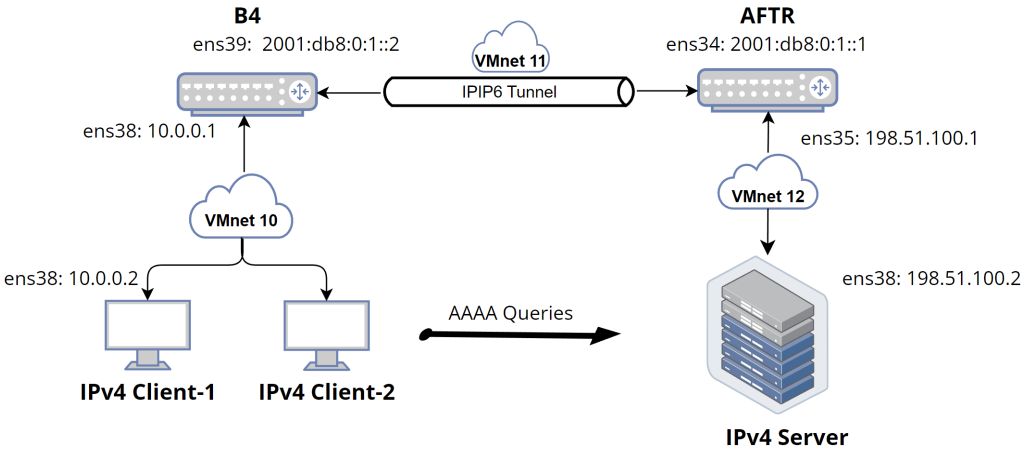

Figure 20: Attacking Scenario 3 (AAAA queries)

which means that for two clients, a total of 5000 queries per second can be sent. The attack was executed by an automated script from client 1, carrying out the following:

- Accesses AFTR remotely (via SSH) and runs `tshark` (traffic monitoring software) and captures the traffic at the ens35 interface;
- Runs the attack from IPv4-client-1 (the same machine);
- Accesses IPv4 client-2 remotely (via SSH) and runs the same attack with the same parameters;
- Waits until attacks are finished on both clients;
- Accesses AFTR remotely (via SSH) and stops `tshark`;
- Sends the `tshark` results as a pcap file to the host machine, so it can be read in the Wireshark software.

Fig. 21 shows the exhaustion of the source ports clearly to prove that a total of 64,512 sent queries must be visible, which comes from subtracting the well-known 1024 ports from the total of 65,536 ports. Fig. 21 also shows that AFTR processed only 64,536 packets at the ens35 interface. After digging deeper, more packets embedded within Wireshark results were found, such as 20 ICMP and 4 ARP packets. As a result, the remaining packets are 64,536 - 20 - 4 = 64,512, which is exactly the anticipated number.

No.	Time	Src. IP	Dst. IP	Protocol	Src. port	Dst. port	Info.
64534	14.336138714	198.51.100.1	198.51.100.2	DNS	38628	53	Standard query 0x8131 AAAA 000-000-129-049
64535	14.339424635	198.51.100.1	198.51.100.2	DNS	38624	53	Standard query 0x8139 AAAA 000-000-129-057
64536	14.386624374	198.51.100.1	198.51.100.2	DNS	38625	53	Standard query 0x81b0 AAAA 000-000-129-176

Figure 21: Last lines for Wireshark capture at ens35 on AFTR

In principle, 60,000 queries were sent from each client (120,000 in total), and AFTR was able to process no more than 64,512 packets and ran out of ports in around 14 s. This failure of service is supposed to last until the UDP session timeout at AFTR is over and new source ports are available

to be reallocated. Moreover, Fig. 22 shows the interaction between all nodes of DS-Lite topology, where the port exhaustion process at AFTR is clearly illustrated with a sequence of packets and when exactly the pool of ports is exhausted.

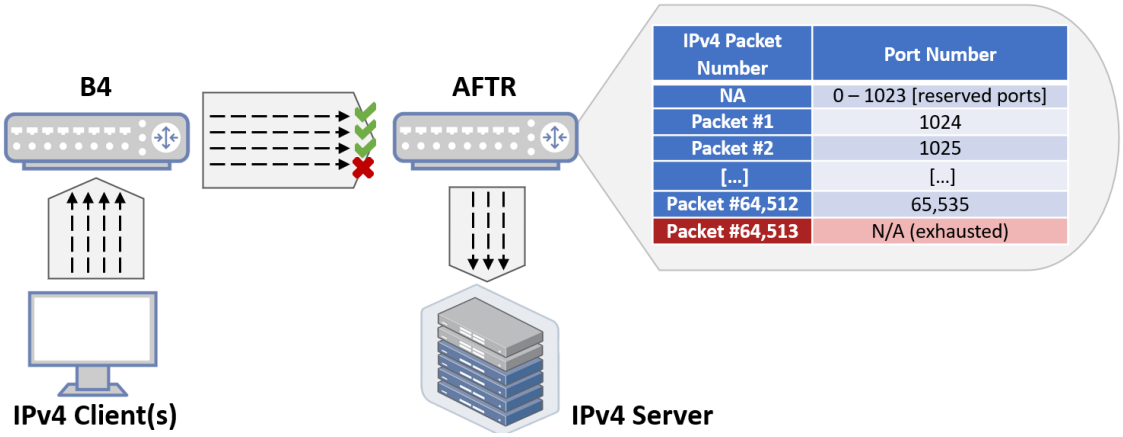

Figure 22: Illustration of port exhaustion at the AFTR

7.4 Mitigation Methods

Several IDS and IPS (Intrusion Detection and Prevention Systems) were tested to identify and drop spoofed packets from legitimate ones. The process proved to be quite hard to implement, especially when the attacker machine is located inside the access network of the ISP (VMnet-11). Therefore, a workaround has been put in place and `iptables` rules were placed to perform the job.

7.4.1 Compromised B4 Network Attack Mitigation

Despite the fact that this attack did not go through and the spoofed packet had been dropped, a proposed solution is presented here in case of building the topology using another machine (different Linux distribution for instance), where such an attack could be prevented at the B4 router side with the following rules:

```
iptables -A FORWARD -i ens38 -s 10.0.0.0/24 -j ACCEPT
iptables -A FORWARD -i ens38 -j DROP
```

The first rule accepts packets with source addresses from the private IPv4 address network, specifically (10.0.0.0/24), while the second one drops every packet that approaches the ens38 interface and needs to be forwarded. These two rules take care of a malicious packet as it reaches the B4 side and before it heads toward the AFTR (see Fig. 19). However, restricting access to the private IP address network of VMnet-10 (10.0.0.0/24) will not fully protect from a DoS attack, because the attack might originate from a machine that resides within the network of the IPv4 client (an inside job).

7.4.2 Source Port Exhaustion Attack Mitigation

One of the ways to mitigate this attack or at least reduce its impact is by adding two (or more) IP addresses to the AFTR ens35 interface (see Fig. 20), where the whole calculation will be double (or triple, depending on how many IP addresses were added) due to the fact that extended NAT works by binding the IP/port pair together. The fact that ens35 has two IP addresses now gives the AFTR more flexibility by having one additional pool of exactly the same port numbers of 64,512 ports, but they will be coupled with another source IP address. In case two IP addresses were not enough to mitigate the attack, the system administrator can easily add a third, fourth, or more as needed. On the practical side, two extra IP addresses were added to the ens35 interface on AFTR. As a result, the AFTR was able to process all incoming 120,000 packets, unlike last time, when it stopped after 64,512 packets. Another mitigation method is possible through the rate-limiting process, which limits the number of packets per second that can be sent and received by the NIC [65]; it also enables the administrator to assign bandwidth restrictions to specific traffic such as ICMP, UDP, etc. Some research works [36, 59] have proposed the rate-limiting mitigation mechanism after noticing the difference and the asymmetry between incoming and outgoing traffic in the designated network.

7.5 Results Summary

As the test-bed for DS-Lite investigated in a topic that has rarely been discussed and analyzed, valuable results were collected based on the attacking scenarios, which can be used to enhance the security around DS-Lite infrastructure's vulnerable spots and, therefore, provide a more secure network infrastructure. Therefore, researchers are encouraged to invest more effort in analyzing the potential security threats that face IPv6 transition technologies, especially DS-Lite. It is recommended that the focus be placed on DoS attacks and source IP address spoofing, as it was found through analysis that these attacks are the most common. As for mitigation methods, a sophisticated IDS and IPS tool is highly recommended, such as SNORT [79] or Suricata [70].

Furthermore, we summarized attacks or the vulnerabilities within the DS-Lite infrastructure in Table 11, where we correlated the attack with its DFD element number.

Table 11: Summary of DS-Lite Threats

DFD Element	Threat	Possible Attacks
1	Spoofing & Repudiation	Spoofing against the IPv4 Client
2, 3	Tampering, Information Disclosure and DoS	DoS against the B4 & Information Disclosure against the IPv4 Client
4	All STRIDE Elements	Spoofing the B4, DoS and unauthorized access
5, 6	Tampering, Information Disclosure and DoS	Information Disclosure against the B4 and the AFTR
7	All STRIDE Elements	Spoofing the AFTR, DoS and unauthorized access
8	Only indirect attacks	DoS against the Connection Tracking Table or Tampering with it
9, 10	Tampering, Information Disclosure and DoS	Information Disclosure against the traffic between AFTR and IPv4 Server
11	Spoofing & Repudiation	Spoofing against the IPv4 Server

7.6 Conclusion

In this Chapter, we have demonstrated the significance of transition technologies, specifically through the tunneling method, and how practical they can be. Our test-bed effectively simulated a DS-Lite topology and uncovered its security vulnerabilities. The findings show that the IPv4 client did not need a public IPv4 address to communicate with an IPv4 server. On the other hand, every element within the DS-Lite topology is vulnerable to various types of attacks, including DoS, tampering and spoofing. As such, further analyses and addressing vulnerabilities are crucial for the successful implementation of IPv6 transition technologies that use tunneling. Our work has benefited from the "`thc-ipv6`" toolkit, which provided us with a range of tools and attacking possibilities. These tools can be used in future research to better understand the weaknesses and threats associated with DS-Lite and other transition technologies. Overall, our study highlights the importance of thorough analysis and vulnerability testing when implementing IPv6 transition technologies that use tunneling. Our findings can benefit network administrators, policymakers and researchers who seek to enhance the security of their networks and prevent attacks.

Chapter 8

Security Analysis of Lw4o6

The lw4o6 technology plays a crucial role in enabling the coexistence of IPv4 and IPv6 networks during the transition phase. However, ensuring the security of lw4o6 deployments is of paramount importance to maintain the integrity and confidentiality of network communications. In this Chapter, we conduct a comprehensive security analysis of the lw4o6 technology and evaluate its potential vulnerabilities and threats.

8.1 Threat Modeling

As we have done with other technologies in previous chapters, we selected STRIDE to be our threat modeling technique to build the DFD of Lw4o6 technology and examine its potential security vulnerabilities.

8.2 Applying STRIDE on lw4o6

According to [78], a DFD of the examined system needs to be drawn for STRIDE to spot the potential vulnerabilities of the given system. Therefore, we have drawn the required diagram as shown in Fig. 23, where the system has 12 vulnerable areas that can be targeted by an attacker.

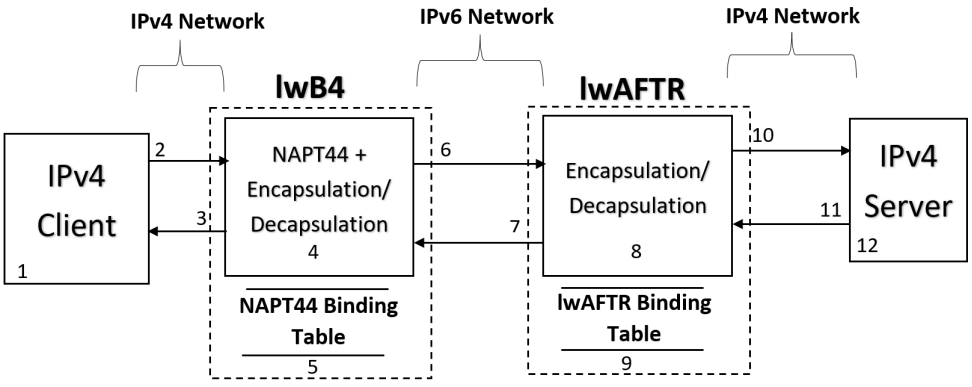

Figure 23: Data Flow Datagram of Lw4o6

8.3 Attacking Possibilities

We conducted a comprehensive analysis of all potential attacking scenarios on lw4o6 by leveraging its DFD using the STRIDE method.

8.3.1 IPv4 Client

(i) Spoofing: an attack spoofs the source IP address of the IPv4 client and misuses the possession of such source IP address by sending harmful packets towards the lwB4 router [78].

(ii) Repudiation: an attacker denies the responsibility of doing any sort of activity such as sending a malicious packet towards the lwB4 router or a packet with a spoofed IP address [78].

8.3.2 Data Flow from IPv4 Client towards the lwB4 Router

(i) Tampering: an attacker intercepts and alters the data being sent from the client to the lwB4 router such as changing the client's original sent information or manipulating an order placed on an e-commerce website [78].

(ii) Information Disclosure: an attacker intercepts and views sensitive information being sent by the client to the lwB4 router, such as viewing a client's credit card information or login credentials [78].

(iii) Denial of Service: an attacker floods the lwB4 router with a large number of requests from the IPv4 client side (or multiple clients), causing the router to become overwhelmed and unable to process legitimate requests. This can cause the router to crash or become unresponsive and deny service to legitimate clients [78]. Moreover, an attacker might send an excessive amount of DNS queries in order to exhaust the allocated pool of ports for the lwB4 router.

8.3.3 Data Flow from the lwB4 Router to the IPv4 Client

(i) Tampering: an attacker alters data sent from the lwB4 router to the client in order to disrupt or gain unauthorized access to the IPv4 client's system. For example, an attacker might modify a software update file sent from a server to the IPv4 client in order to include malware [64].

(ii) Information Disclosure: an attacker intercepts or otherwise gains access to sensitive information sent from a server to the IPv4 client. For example, an attacker might use a MITM attack to intercept and read login credentials sent from a server (via lwB4 router) to the IPv4 client in clear text [78].

(iii) Denial of Service: an attacker floods the IPv4 client with an excessive amount of requests from the lwB4 router side, causing the client to become unavailable to legitimate incoming requests. For example, an attacker might launch a distributed denial of service (DDoS) attack against the client in order to disrupt its access to a specific website [78].

8.3.4 The lwB4 Router

(i) Spoofing: an attacker spoofs the source IP address of the lwB4 router (impersonates it) and initiates communication with neighboring (or remote) devices, while sending all sorts of malicious packets, that could harm the reputation of the organization that operates the router itself [78]. Moreover, another attack scenario is possible such as the potential for an ARP (Address Resolution Protocol) cache poisoning attack. The attack involves the interception of network traffic between the lwB4 router and the lwAFTR router, enabling an attacker to exploit the situation. By sending deceptive ARP messages to the lwB4 router, the attacker can assume the identity of the lwAFTR router. Consequently, the lwB4 router updates its ARP cache with the attacker's MAC (Media Access Control) address, erroneously associating it with the lwAFTR router. Once the ARP cache has been successfully poisoned, the attacker gains the ability to intercept, manipulate, or extract sensitive information from the network traffic. This form of attack poses a significant threat, capable of compromising the security of the system [1].

(ii) Tampering: an adversary might alter the actual data that is being processed or stored within the router such as IP addresses, port numbers, TTL values, etc. It may lead to redirecting the packet to a malicious server and prevent the legitimate recipient of the packet from getting a response to his request [78].

(iii) Repudiation: after spoofing the source IP address of the lwB4 router and sending harmful packets, the attacker will be able to deny the fact that he was behind those suspicious packets [78].

(iv) Information Disclosure: an attacker having unauthorized access to confidential data within the lwB4 router such as packet's payload or the routing table of the lwB4 router itself [78]. Another type of information disclosure attack can be performed by an attacker gaining access to the network topology and the number of hops to reach a specific target.

(v) Denial of Service: an attacker might target the lwB4 router with the well-known DoS attack. This can be done either from the IPv4 client side or from the lwAFTR side, where a huge amount of useless packets can be sent to the lwB4 router with the aim to overwhelm its processing power [78], please refer to Section 8.3.3.(iii).

(vi) Elevation of Privileges: an adversary bypasses the authority matrix within the organization to gain access (such as read/write permission) and therefore pushes a destructive configuration to the lwB4 router, which affects the whole network [78].

8.3.5 The NAPT44 Binding Table of the lwB4 Router

(i) Tampering: an attacker could potentially tamper with the NAPT44 binding table (see Table 3 in Sub-section 2.3.2) by altering the information stored in it, such as the IP addresses, ports, or protocol information. This causes the lwB4 to misroute or block legitimate traffic, leading to a communication breakdown [78].

(ii) Denial of Service: an attacker could launch a Denial of Service attack on the NAPT44 binding table by overloading it with a large number of fake or malformed connection entries. This could cause the table to become full, preventing the lwB4 router from tracking legitimate connections and resulting in a communication breakdown [78].

8.3.6 Data Flow from the lwB4 to the lwAFTR

(i) Tampering: an attacker has the capability to introduce a malevolent packet into the network traffic, leading to detrimental consequences for the the lwAFTR. This act can be classified as a packet injection attack [22]. As an example, the malicious packet could manifest as a TCP RST packet, which promptly terminates an existing TCP connection. Moreover, the attacker might hijack the TCP session by eavesdropping on the communication between the lwB4 and the lwAFTR, then start communicating with the IPv4 server via the lwAFTR.

(ii) Information Disclosure: an attacker intercepts and views sensitive information, such as the TCP/UDP packet's payload, please refer to Section 8.3.2: (ii).

(iii) Denial of Service: an attacker floods the lwAFTR router with useless packets to overwhelm it, please refer to Section 8.3.2: (iii).

8.3.7 Data Flow from the lwAFTR to the lwB4

(i) Tampering: an attacker performs a MITM attack and alters the packet's details such as the destination IP address, which will re-direct the packet to a potentially malicious server and deprive the legitimate lwB4 router of the response it was anticipating.

(ii) Information Disclosure: as the attacker performs a MITM attack, he also exposes the content of the sent data, which is a data confidentiality breach.

(iii) Denial of Service: an attacker floods the lwB4 router with an excessive amount of packets in order to overwhelm its computation power, please refer to Section 8.3.2: (iii).

8.3.8 The lwAFTR Router

(i) Spoofing: an attacker might impersonate the lwAFTR router and initiate a communication with the IPv4 Server or the lwB4, which will make all of those network elements liable to the risk of exchanging sensitive data with an adversary [78].

(ii) Tampering: an attacker alters the content of the sensitive data within the lwAFTR router, such as the provisioned PSID value assigned for a specific lwB4 router, please refer to Section 8.3.4: (ii).

(iii) Repudiation: the packet sender hides his own identity, please refer to Section 8.3.4: (iii).

(iv) Information Disclosure: an attacker getting access to confidential data inside the lwAFTR router, such as packet's payload [78], please refer to Section 8.3.4: (iv).

(v) Denial of Service: an attacker can exhaust the computation power of the lwAFTR router by sending too many useless packets, which will prevent it from being able to process any incoming packets from the lwB4 router.

(vi) Elevation of Privileges: an attacker getting high privilege access to the lwAFTR router such as admin permission [78]. Such attacks happen most of the time due to inside job [50].

8.3.9 LwAFTR Binding Table

This table stores all of softwires (tunnels) configured into the lwAFTR, see Table 7 in Sub-section 5.3.2.

(i) Tampering: an attacker alters the content of an entry within the table such as PSID value or lwB4's public IPv4 address. Such changes will pave the way for a malicious spoofed packet to be processed by the lwAFTR, while dropping the legitimate requests.

8.3.10 Data Flow from the lwAFTR Router towards the IPv4 Server

(i) Tampering: please refer to Section 8.3.3: (i)

(ii) Information Disclosure: please refer to Section 8.3.3: (ii)

(iii) Denial of Service: please refer to Section 8.3.3: (iii)

8.3.11 Data Flow from the IPv4 Server towards the lwAFTR Router

(i) Tampering: please refer to Section 8.3.2: (i).

(ii) Information Disclosure: please refer to Section 8.3.2: (ii).

(iii) Denial of Service: please refer to Section 8.3.2: (iii).

8.3.12 IPv4 Server

(i) Spoofing: please refer to Section 8.3.1: (i).

(ii) Repudiation: please refer to Section 8.3.1: (ii).

8.4 Vulnerability Assessment

Upon conducting a comprehensive examination of the potential security threats associated with lw4o6, we have reached the conclusion that the exploitable threats accessible to an attacker can be succinctly summarized in Table 12. To facilitate a clearer understanding, we have classified the severity of these attacks into distinct levels, namely low, medium, high and critical. The categorization is based on the detrimental impact inflicted upon the targeted system, as well as the intricacy involved in performing and mitigating the respective attack.

Table 12: Summary of the Potential Vulnerabilities of Lw4o6

Attack Name	Intricacy of Performing the Attack	Intricacy of Mitigation	Attack Impact (Severity)
TCP RST Signal	Average	Average	Low
IP Address Spoofing	Average	Difficult	Medium
Packet Injection	Average	Difficult	Medium
Information Disclosure	Average	Easy	Medium
Packet's Payload Tampering	Difficult	Difficult	Medium
ARP Poisoning	Average	Difficult	High
Source Port Exhaustion	Easy	Average	High
TCP Session Hijacking	Easy	Average	Medium
Network Mapping	Easy	Easy	Low
DoS using TCP SYN Flood	Easy	Difficult	Critical

8.5 Results and Analysis

8.5.1 Normal lw4o6 Process

During pinging the IPv4 server from IPv4 client side (see Fig. 11 in Section 5.3.1), we monitored the traffic with the "`tcpdump`" command on different locations as shown in Table 13, where the NAT44 + encapsulation / decapsulation on lwB4 side and encapsulation / decapsulation processes on lwAFTR side are quite obvious.

Table 13: Lw4o6 Packet Encapsulation/Decapsulation Process

Packet capture at lwB4 ens34
17:22:15.047747 IP 10.0.0.2 > 192.0.2.2: ICMP echo request, id 21142, seq 1, length 64
17:22:15.054865 IP 192.0.2.2 > 10.0.0.2: ICMP echo reply, id 21142, seq 1, length 64
Packet capture at lwB4 ens35
17:22:15.047840 IP6 2001:db8:0:1::2 > 2001:db8:2::2: IP 203.0.113.1 > 192.0.2.2: ICMP echo request, id 1026, seq 1, length 64
17:22:15.054630 IP6 2001:db8:2::2 > 2001:db8:0:1::2: IP 192.0.2.2 > 203.0.113.1: ICMP echo reply, id 1026, seq 1, length 64 1
Packet capture at lwAFTR ens35
17:22:15.050075 IP 203.0.113.1 > 192.0.2.2: ICMP echo request, id 1026, seq 1, length 64
17:22:15.051435 IP 192.0.2.2 > 203.0.113.1: ICMP echo reply, id 1026, seq 1, length 64

8.5.2 PSID Test

In our test-bed, we send packets from lwB4 with a specific port set [1024 - 2047]. As a result, we calibrated that on the lwAFTR side with the below details:

- PSID length : 6
- PSID : 1

Packets went through without any drop, however, when we changed the PSID value on lwaftr softwire configuration to 2, while keeping the port range on lwB4 the same, packets did not go through lwAFTR and an ICMP6 "Destination Unreachable" message was sent back to lwB4. The reason behind the packet drop is that PSID 2 refers to port set [2048 - 4095], while we kept sending packets using the old ports range [1024 - 2047].

8.6 Attacking Scenarios

8.6.1 DoS Attack

As shown in Fig. 24, an attacker machine was deployed to flood the lw4o6 infrastructure with too many TCP synchronization requests and thus perform a DoS attack against lwB4 and lwAFTR.

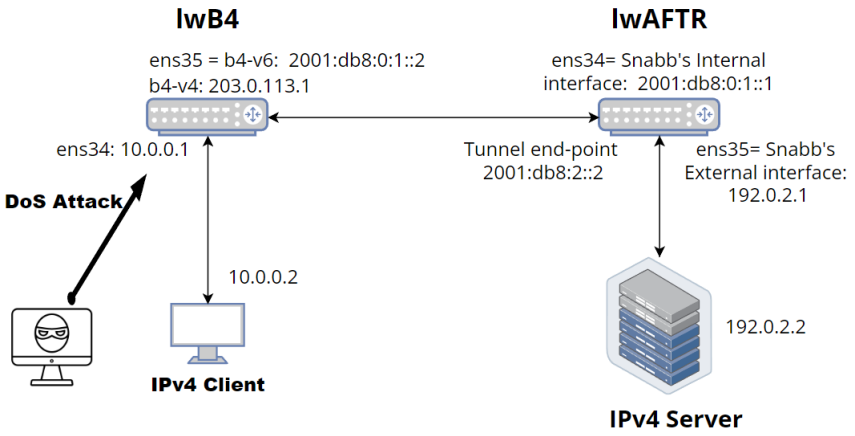

Figure 24: DoS Attack Against Lw4o6 Infrastructure.

The attack was executed while the IPv4 client communicating with the IPv4 server normally, and it took around 3-5 seconds for the IPv4 client to show 75% packet loss. The attack was performed using "`hping3`" package:

```
hping3 -S --flood -V -p 80 192.0.2.2
```

In this situation, we explored a scenario where an intruder gains entry to the client's local network and engages in harmful actions aimed at overwhelming the access point. The objective is to obstruct the client(s) from receiving responses to their requests.

Moreover, Fig. 25, shows the CPU utilization of the lwB4 machine before and after the attack, where the CPU of the machine was fully utilized within 5 seconds.

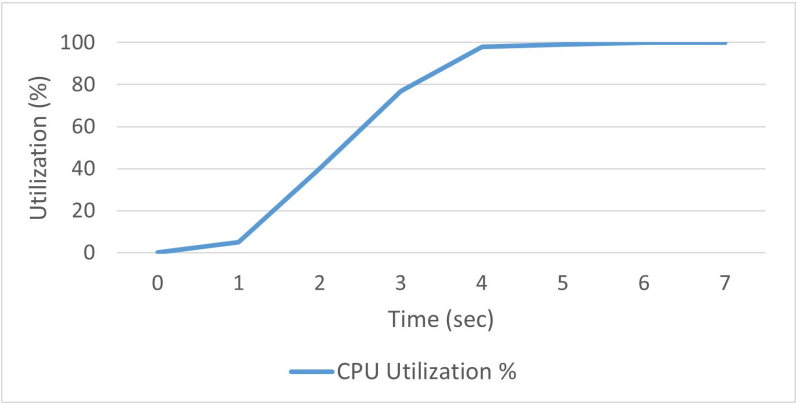

Figure 25: CPU utilization for lwB4 Machine

8.6.2 Information Disclosure

As shown in Fig. 26, we carried out an information disclosure attack against the traffic between lwB4 and lwAFTR, where we used scapy script to sniff the communication channel and print out the payload of the TCP/UDP packets. The attack was successful, as it printed out the content of the payload in plain text. The script can be found under the name of "`info-disclosure.py`" in our GitHub repository [2].

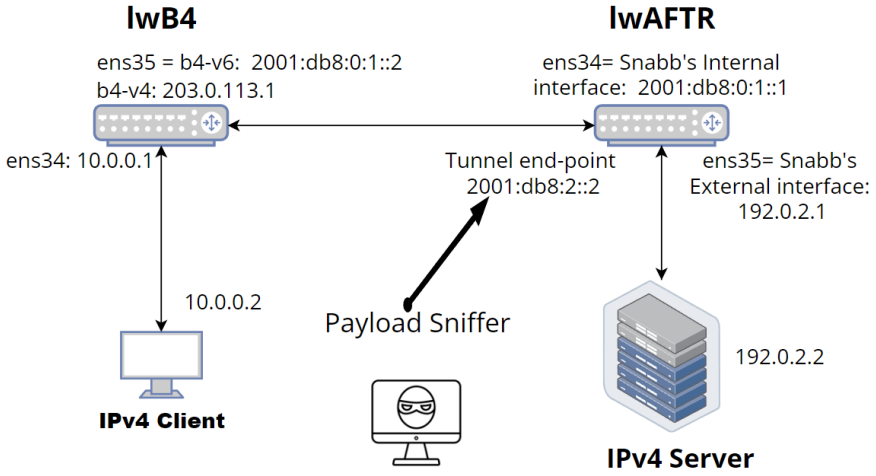

Figure 26: Information Disclosure attack using Scapy script

8.6.3 ICMP Spoofing

Since ICMP packets do not have port numbers, lw4o6 handles ICMP packets differently than it does with TCP/UDP packets, especially when it comes to packet filtering at lwB4 and lwAFTR routers. The solution was presented in RFC-7596 Section 8.1 [21], where the lwB4 router encapsulates a port number (out of the assigned pool of ports) into the ICMP ID field. Therefore, when the ICMP packet reaches the lwAFTR router, its ICMP ID field will be inspected and treated as the source port number, where the same process of packet filtering (Section 2.3.2) will be applied to the packet.

The attack was made possible by a script based on a powerful Python library called Scapy [15], which is an interactive packet manipulation program. The script can be accessed under the name of "`icmp-spoofer.py`" in our GitHub repository [2].

As illustrated in Fig. 27, the idea behind this attack is to sniff the communication channel for any ICMP packet being forwarded from the lwB4 towards the lwAFTR, then send a crafted ICMP packet to the lwAFTR. Before sending the crafted packet, Scapy makes sure that the new packet has similar details to the original one while altering the payload's content (the transmitted data), uses the same ICMP ID (for the sake of port mapping) and then forwards it to the lwAFTR.

The packet journey of the spoofed packet follows the below path:
Attacker $\Rightarrow$ lwAFTR $\Rightarrow$ IPv4 Server $\Rightarrow$ lwAFTR $\Rightarrow$ lwB4 $\Rightarrow$ IPv4 Client. As a result, the IPv4 client received a reply for a packet that it never sent.

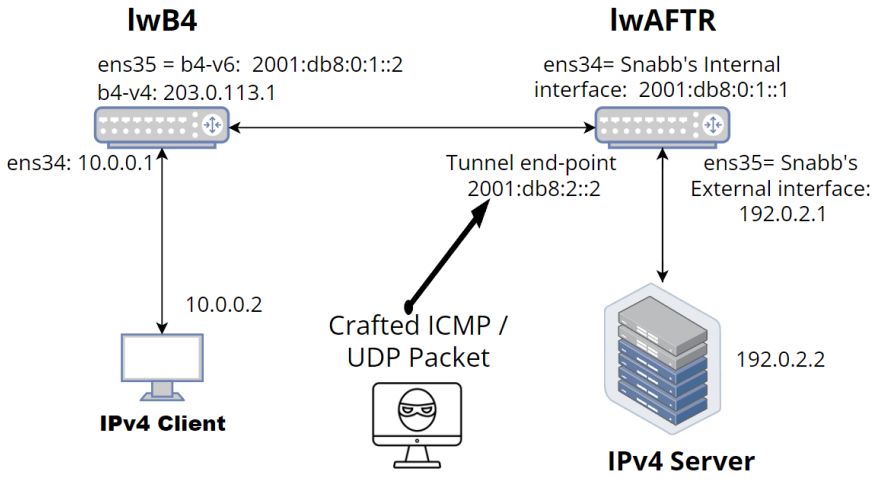

Figure 27: ICMP / UDP packet Spoofing Attack.

8.6.4 UDP Spoofing

As shown in Fig. 27, we repeated the same attack but targeted UDP traffic. Therefore, we applied a different Python script, which can be accessed under the name of "`udp-spoofer.py`" in our GitHub repository [2]. The script sniffs the traffic between lwB4 and lwAFTR routers, looks for UDP traffic and crafts new UDP packets based on the same details, especially the source port number. In addition, the newly crafted packet has the wrong payload (random text generated by the script). Eventually, the attacker machine forwards the crafted packet to the lwAFTR router. The attack was successful and the lwAFR router processed the malicious packet normally and forwarded it to the IPv4 server.

On the other hand, we reiterated the same attack with a minor adaptation. In contrast to the initial method of extracting the source port from the UDP packet, we devised a customized packet with a randomly generated source port number and subsequently directed it toward the lwAFTR. Consequently, the lwAFTR promptly discarded the packet owing to the absence of the appropriate source port. Further details and implementation of this script can be accessed in our GitHub repository under the file name "`random-source-port.py`" [2]

8.6.5 Source Port Exhaustion

Our lwB4 router was equipped with a specific port range: [1024-2047]. Therefore, we decided to exploit this vulnerability. As illustrated in Fig. 28, we used a tool called "`dns64perf++`" that generates an excessive amount of DNS queries toward the IPv4 server [14]. DNS queries are UDP packets and they require a UDP port to be assigned for each packet. As a result, we managed to exhaust the pool ports in less than one second. Dns64perf++ tool generated 2500 packets/second, which exhausted the pool of source ports within the lwB4 router. The script can be found under the name of "`port-exhaust.sh`" in our GitHub repository [2].

Fig. 29 shows the last three lines of Wireshark capture on lwB4 ens35, where we managed to exhaust the source port pool [1024-2027] in less than one second. It also shows traffic stopped in

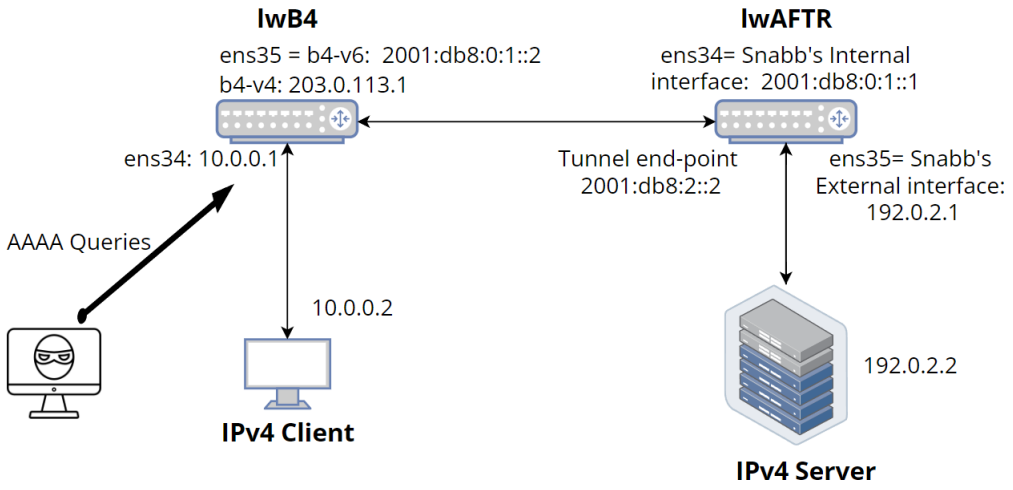

Figure 28: Source Port exhaustion

less than half of a second and then resume at the 30th second. This was due to the default timeout of the UDP connection, where ports are re-assignable after 30 seconds.

It is worth mentioning that "`DNS64perf++`" was designed to be used as a measurement tool, however, we used it as an attacking method.

No.	Time	Src. IP	Dst. IP	Protocol	Src. port	Dst. port	Info.
1028	0.561305919	203.0.113.1	192.0.2.2	DNS	2045	53	Standard query 0x03fd AAAA 000-000-003
1029	0.562221123	203.0.113.1	192.0.2.2	DNS	2046	53	Standard query 0x03fe AAAA 000-000-003
1030	0.562592491	203.0.113.1	192.0.2.2	DNS	2047	53	Standard query 0x03ff AAAA 000-000-003
1046	30.010075937	203.0.113.1	192.0.2.2	DNS	1024	53	Standard query 0xd524 AAAA 000-000-213

Figure 29: Wireshark Capture on lwB4 ens35

8.6.6 TCP Reset Signal

The idea behind the attack is to send a TCP RST signal to the IPv4 server (via the lwAFTR router) and terminate an existing TCP connection with the IPv4 client. We achieved our goal by writing a Python script that sniffs the traffic between the lwB4 and the lwAFTR routers, searches for TCP packets with activated SYNC+ACK flags and crafts new TCP RST packet accordingly. The crafted TCP RST packet was forwarded by the attacker to the lwAFTR router, which forwarded the packet again to the IPv4 server. The crafted packet forced the IPv4 server to terminate the TCP connection with the IPv4 server (until the next TCP Sync packet comes again from the IPv4 client).

The script can be found under the name of "`lw-tcp-reset.py`" in our GitHub repository [2].

8.6.7 TCP Session Hijacking

TCP session hijacking attack is a cyberattack where an unauthorized party intercepts and takes control of an established TCP connection between two communicating entities, potentially gaining unauthorized access and control over the communication or data exchange [76].

We conducted this attack by employing the Scapy software to intercept communication between the lwB4 and the lwAFTR machines, as depicted in Fig. 30. The attack is initiated when the "attacker-1" machine detects a TCP ACK packet moving from the IPv4 client to the IPv4 server. Upon identification of this packet, the attacker extracts pertinent information such as IP addresses, port numbers, sequence and acknowledgment numbers from the exchanged TCP packets and sends crafted packets to the IPv4 server accordingly.

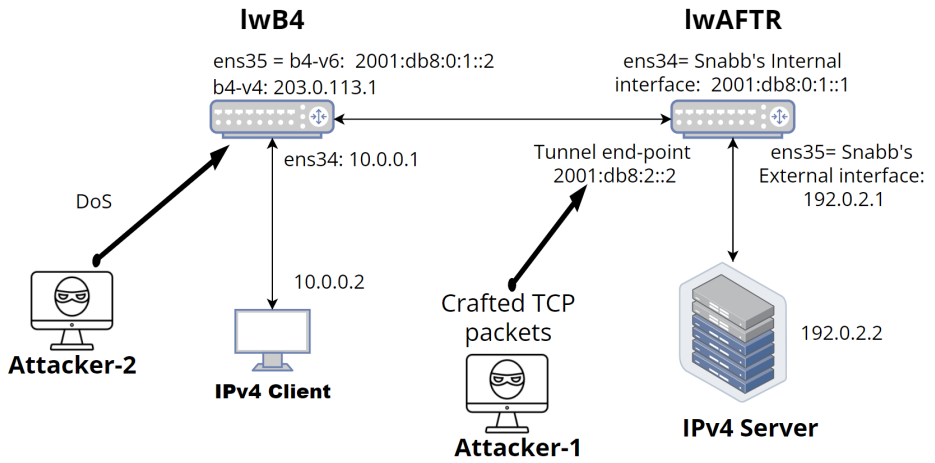

Figure 30: TCP Session Hijacking Attack

Subsequently, the "attacker-1" machine fabricates a TCP packet with the "`RST`" flag and relays it to the IPv4 client via the lwB4 machine in order to deceive the IPv4 client with a fake TCP session abortion signal. Furthermore, the attack script continuously monitors the communication channel, responding to relevant TCP packets and adapting responses with accurate information and appropriate flags.

Concurrently, the script establishes an SSH connection from the "attacker-1" to the "attacker-2" machine and initiates a DoS attack against the lwB4 router with the goal of exhausting its CPU resources. This action aims to obstruct any future communication between the IPv4 client and the IPv4 server via the lwB4 router. Consequently, the "attacker-1" machine gains the ability to communicate with the IPv4 server and hijacks the ongoing TCP session.

Notably, the script is also designed to send meticulously crafted TCP packets containing the "`PSH`" and "`ACK`" flags (with falsified payloads) to the IPv4 server. This process is iterated each time the IPv4 server responds with a TCP packet bearing an "`ACK`" flag. The script can be found under the name "`tcp-session-hijack.py`" in our GitHub repository [2].

It's important to note that the use of the "`PSH`" and "`ACK`" flags is not mandatory in all TCP connections. By utilizing it, the server can ensure that data is delivered to the application as soon as possible, reducing latency and improving the user experience. Results of the attack were made public under this path: "`files/tcp-session-hijack-results.txt`" in our GitHub repository [2], where it shows the full chain of TCP communication between the attacker and the IPv4 server via the lwAFTR machine.

8.6.8 Network Mapping

In order to expose the topology of lw4o6, we conducted an experiment with a simple attacking command: `traceroute 192.0.2.2`
The result was as follows:

```
30 hops max, 60 byte packets
1 10.0.0.1 (10.0.0.1) 1.154 ms  0.739 ms  1.911 ms
2  192.0.2.2 (192.0.2.2)  36.101 ms  36.302 ms 36.627 ms
```

Such a result tells the attacker that there are two hops till he can reach the target. It also can help the attacker to map the network topology and reveals the network infrastructure, routers and servers in between the source and target. This knowledge can aid in identifying potential points of entry or weak links in the network.

8.7 Mitigation Methods

8.7.1 DoS Attack Mitigation

We mitigated the attack by deploying "`iptables`" rules on the lwB4 machine to perform a rate-limiting mechanism and setting a filter with two rules, one that allows packets to be forwarded with certain limits (10 p/s for instance), while the second rule drops everything else.

```
iptables -A FORWARD -p tcp--syn -m limit --limit 10/s -j ACCEPT
iptables -A FORWARD -p tcp--syn -j DROP
```

8.7.2 Information Disclosure Mitigation

We have successfully mitigated the information disclosure attack, which we presented in Section 8.6.2. We achieved that by encrypting the payload of TCP/UDP packets generated by the IPv4 client using the Python Scapy script. The script leverages a cryptographic Python library called "`Fernet`", which provides a straightforward and secure way to encrypt and decrypt data [43]. It is specifically designed for symmetric encryption, which means the same key is used for both encryption and decryption processes.

The script can be found in our GitHub repository under the name of "`payload-encryptor.py`" [2]. Therefore, after running the same attacking script, the actual payload of the packet was not visible, only its encrypted value.

While the "Fernet" Python library was utilized for encryption in this thesis for demonstration purposes, it's important to note that industry standard alternatives like TLS (Transport Layer Security) offer more advanced key management techniques and heightened security measures.

8.7.3 ICMP Spoofing Mitigation

An advanced packet crafting software packages, such as Scapy [15], can create a very realistic crafted packet with almost the exact anticipated values (IP addresses, port numbers, Packet se-

quence and even MAC addresses). Therefore, a sophisticated tool such as SNORT [79], which functions as IDS (Intrusion Detection System) and IPS (Intrusion Prevention System) as well, would be required. The tool performs a deep inspection mechanism, where it detects the suspicious (spoofed) packet and drops it eventually. SNORT proved to be complicated to configure. Therefore, we omitted it as it is out of our scope.

8.7.4 UDP Spoofing Mitigation

Please refer to the above Sub-section (8.7.3).

8.7.5 Source Port Exhaustion Mitigation

To counter the attack, we implemented rate limiting for DNS queries, ensuring that only 100 packets per second are allowed. We achieved this by configuring "`iptables`" rules to drop incoming DNS query packets by default while permitting the specified rate (100 packets per second).

As a result, the pool of the allocated ports within the lwB4 router could not be exhausted with such a low rate of DNS queries. The complete script can be found in our GitHub repository under the name of "`exhaust-mitigate.sh`" [2].

8.7.6 TCP Reset Signal Mitigation

The mitigation proved to be very complicated, an IDS / IPS device is required for deep inspection, please refer to the above Sub-section 8.7.3.

8.7.7 TCP Session Hijack

Please refer to Sub-section (8.7.3).

8.7.8 Network Mapping

There are several possible mitigation methods for such an attack. For example, disabling ICMP echo replies at the network perimeter or on specific routers, will prevent outsiders from using "`traceroute`" to discover the internal network structure. Moreover, a network segmentation strategy can help protect against network mapping attacks and enhance overall network security. Network segmentation involves dividing a large network into smaller, isolated segments or subnetworks. Each segment is logically separated from the others and may have its own security policies, access controls and communication rules [63].

8.8 Results Summary

Upon conducting six distinct attack scenarios, several were successfully mitigated, while alternative mitigation strategies were proposed for the remaining scenarios. Consequently, our findings are encapsulated in Table 14, offering a comprehensive overview encompassing the attacks, their

corresponding mitigation measures and an assessment of the severity imposed upon the victim. The severity scale was determined by evaluating the intricacy of executing each attack and the technical prerequisites necessary for their mitigation.

Table 14: Implemented attacks against the lw4o6 infrastructure

Attack Name	Mitigation Method	Severity
DoS	Rate-Limiting	Critical
Information Disclosure	Payload Encryption	High
ICMP Spoofing	IDS / IPS [79]	Medium
UDP Spoofing	IDS / IPS	Medium
Source Port exhaustion [14]	iptables rules	High
TCP RST Signal	IDS / IPS	Medium
TCP Session Hijacking	IDS / IPS	High
Network Mapping	Network Segmentation	Low

8.9 Conclusion

In this Chapter, we conducted a security analysis for lw4o6 IPv6 transition technology by applying the STRIDE threat modeling technique. As a result, we listed the most common attacks based on their severity and impact on the targeted machine. Consequently, we built a test bed for lw4o6 technology, leveraged the common attacks list and performed several attacking scenarios against our testbed. Lw4o6 can be built using open-source software packages such as VPP, Snabb, etc. Moreover, the Snabb proved to be reliable software to build the lwAFTR over Linux-based virtual machines. Our test-bed was a success, where we performed the normal packet NAT44 operation + packet encapsulation/decapsulation smoothly.

We managed to identify vulnerabilities using our attack scenarios, such as DoS, spoofing, Tampering, Information Disclosure, source port exhaustion, TCP Session Hijack, etc. Eventually, we implemented and proposed mitigation methods against these potential attacks. However, some attacks were not mitigated such as UDP packet spoofing and TCP Session Hijack, where an IDS is required to be installed to filter such packets and detect similar sophisticated attacks. Finally, Scapy packer manipulation software proved to be an effective tool for crafting packet with high precision that can deceive routers and impersonates any sort of packet (TCP, UDP, ICMP, etc.).

Chapter 9

Security Analysis of MAP-T / MAP-E

9.1 Applying STRIDE on MAP-T

To inspect the vulnerabilities of MAP-T technology, we built a DFD of the system's topology to identify the threat points one by one. As shown in Fig. 31, we identified 11 points of vulnerabilities, which we intend to inspect in the next section.

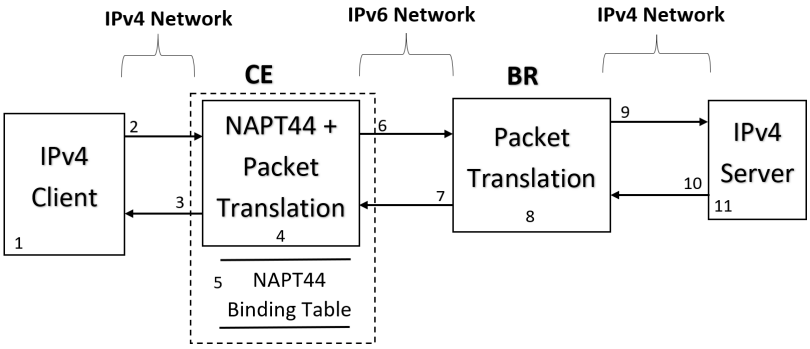

Figure 31: Data Flow Diagram of MAP-T

9.2 Attacking Possibilities

9.2.1 IPv4 Client

(i) Spoofing: an adversary spoofs the IP address of the IPv4 client and starts sending a huge number of packets toward the CE router to fully utilize its processing power.

(ii) Repudiation: an adversary denies the fact that he sent a specific packet from the IPv4 client machine. As a result, the attacker might send a harmful packet (echo request, DNS resolution request, etc.) to the CE router and then deny his responsibility [78].

9.2.2 Data Flow from the IPv4 Client to the CE Router

(i) Tampering: an attacker sniffs the communication channel and tampers (alters) the content of the exchanged data like destination IP address, TTL, etc. Changing the destination IP address would diverge the packet to a malicious server and cause an FoS attack [50], which means a legitimate user does not get a response to his request.

(ii) Information Disclosure: an adversary getting access to confidential information such as online banking login credentials, user browsing habits, etc. [78].

(iii) Denial of Service: an attacker sends a huge number of useless queries to the CE router to overwhelm it and prevent the legitimate packets from being processed [78].

9.2.3 Data Flow from the CE Router to the IPv4 Client

(i) Tampering: in this direction, the victim is the IPv4 client himself, while the attacker performs MITM attack and alters the content of the sent packets [78]. Another potential risk is an injection attack, where the attacker injects malicious packets into an existing network flow [47]. Moreover, an adversary might inject a TCP RST signal to the client causing a termination of the already established TCP connection.

(ii) Information Disclosure: an attacker getting unauthorized access to sensitive information from the packets headed from the CE router towards the client [78].

(iii) Denial of Service: an adversary flooding the IPv4 client with an excessive amount of requests with the intention to overwhelm it and deprive it of the ability to process any more packets [78].

9.2.4 The CE Router

(i) Spoofing: an attacker's machine spoofs the source IP address of the CE router and starts communicating with other network elements of MAP-T infrastructure such as the IP4 client or the BR router. This action endangers every machine that talks to him (the attacker) by exposing sensitive information that is meant to be sent to the legitimate CE router [78]. On the other hand, ARP Cache Poisoning attack is also a possibility here, where an attacker can intercept network traffic between the CE router and BR router (for instance). The attacker can impersonate the BR router by sending fake ARP messages to the CE router, causing it to update its ARP cache with the attacker's MAC address instead of the correct MAC address of the BR router. Once the attacker has successfully poisoned the ARP cache of the target device, he can intercept and modify network traffic, steal sensitive information or launch other attacks [1].

(ii) Tampering: simply modifying the actual content of the packet's information such as the destination IPv6 address and redirecting the packet towards a dangerous recipient instead of the BR router [78]. Moreover, a change in the source port will change the entry in the

NAPT table (see Table 4) and result in forwarding the returned packet to a malicious server instead of the original requester [78].

(iii) Repudiation: an attacker spoofs the source IPv6 address of the CE router and performs all sorts of illegal activities, then denies the fact that he did so because it was not his IP address, but the actual CE's IP address.

(iv) Information Disclosure: an attacker having unauthorized access to confidential data within the CE router such as the whole routing table of the CE router or the payload's content of some packets [78].

(v) Denial of Service: an attacker flooding the CE router with a huge number of useless packets and putting it out of service for at least several seconds (depending on the computation power of the CE router) [78].

(vi) Elevation of Privileges: an adversary gets highly privileged permissions inside the CE router such as root permission, which makes it easier for the attacker to perform his malicious activities [78]. Those kinds of attacks happen mostly due to an inside job [50].

9.2.5 The NAPT44 Binding Table of the CE Router

(i) Tampering: an attacker alters the NAPT44 table (see Table 4), such as the public IPv4 address, port number, etc. Such an act will lead to a wrong translation process by the CE router and eventually re-direct the traffic to a malicious server.

(ii) Denial of Service: an attacker inundates the CE router with superfluous packets. This flood of packets results in the addition of unnecessary entries to the NAPT44 table, eventually overwhelming it and causing legitimate entries to be dropped.

9.2.6 Data Flow from the CE Router to the BR Router

(i) Tampering: an attacker sniffs the communication channel and performs MITM attack, where he alters the content of the transmitted data, causing packet's re-direction [78], please refer to Section 9.2.2.(i).

(ii) Information Disclosure: an attacker sniffs the channel and exposes the sensitive/confidential information, which puts both CE and BR routers at risk [78], please refer to Section 9.2.2.(ii).

(iii) Denial of Service: an attacker floods the BR router with an excessive stream of packets to overwhelm its processing power [78], please refer to Section 9.2.2.(iii).

9.2.7 Data Flow from the BR Router to the CE Router

(i) Tampering: an attacker intercepts the flowing data and alters the packet content such as IP addresses and port numbers, which shifts the traffic to a malicious recipient [78], please refer to Section 9.2.2.(i).

(ii) Information Disclosure: an attacker gets access to the exchanged data between CE and BR routers, which might contain confidential information such as online banking credentials [78].

(iii) Denial of Service: an attacker gets access to the BR router and sends a huge amount of packets towards the CE router to overwhelm its processing power [78], please refer to Section 9.2.2.(iii).

9.2.8 The BR router

(i) Spoofing: an attacker impersonates the BR router by spoofing its source IP address and starts communicating with the CE router or the IPv4 server, which endangers all of the involved machines with the possibility of sharing sensitive data with the attacker [78]. Another risk is the ARP Cache Poisoning attack, where the attacker sends a falsified ARP message to the BR router to associate the attacker's MAC address with the spoofed IP address of another host (IPv6 address of CE for example) [1].

(ii) Tampering: the attacker might alter the data that is being processed within the BR router itself such as the mapped IP addresses, port numbers, DMR and FMR rules, etc. Tampering with such data will lead to altering the IPv6 destination address and eventually forwarding the packet to the malicious server [78].

(iii) Repudiation: when the attacker spoofs the source IP address of the BR router, he can send harmful packets to a recipient and then claim he has not done so because the packet's source IP address does not belong to him, but to the BR router.

(iv) Information Disclosure: an attacker gets access to the BR router and therefore exposes confidential data such as the packet's payload, that might contain confidential information.

(v) Denial of Service: an attacker executes the infamous process of DoS attack, which involves flooding the BR router with a large number of packets to exhaust its CPU and prevent the BR router from processing incoming packets from the CE router or the IPv4 server for a specific amount of time.

(vi) Elevation of Privileges: an attacker getting high privilege access inside the BR router such as read and write permission, which will allow him to run harmful scripts such as altering the routing table.

9.2.9 Data Flow from the BR router towards IPv4 Server

(i) Tampering: an attacker alters the content of the exchanged data. This could involve injecting malicious code, and commands or manipulating the data in a way that leads to unintended consequences or security vulnerabilities, please refer to Section 9.2.3.(i).

(ii) Information Disclosure: an attacker exposes the content of the exchange packets' payload, such as online banking credentials, which were originally sent by the IPv4 client, please

refer to Section 9.2.3.(ii).

(iii) Denial of Service: an attacker exhausts the IPv4 server with an extreme rate of packets to overwhelm it and prevent it from replying to the legitimate request of the BR router and therefore to the original request of the IPv4 client, please refer to Section 9.2.3.(iii).

9.2.10 Data Flow from IPv4 Server towards BR router

(i) Tampering: an attacker creates fraudulent data packets masquerading as legitimate requests by forging source addresses, manipulating headers, or injecting malicious code within the packets, please refer to Section 9.2.2.(i).

(ii) Information Disclosure: an attacker eavesdrops on the exchanged data and extracts sensitive information, such as a token sent by the IPv4 server, which can be used to access a particular service, please refer to Section 9.2.2.(ii).

(iii) Denial of Service: an attacker overwhelms the BR router with a flood of requests to make it unavailable or unresponsive to legitimate requests from both the IPv4 server and the CE router, please refer to Section 9.2.2.(iii).

9.2.11 IPv4 Server

(i) Spoofing: an attacker sends a packet with a spoofed source IP address (IPv4 server IPv4 address) and communicates with the BR router, which endangers the BR router by trusting and exchanging sensitive information with a malicious machine, please refer to Section 9.2.1.(i).

(ii) Repudiation: after spoofing the source IP address of the IPv4 server, an attacker sends the harmful packet to the BR router, then denies his responsibility for such an act, because the logged source IP address doesn't belong to him but to the IPv4 server, please refer to Section 9.2.1.(ii).

After conducting a thorough analysis of potential security threats in MAP-T, we evaluated the severity of each identified threat. Table 15 categorizes those threats according to three key parameters: Intricacy of Performing the Attack, Intricacy of Mitigation and Attack Impact (Severity). Furthermore, Table 15 provides a structured classification of threats based on the complexity involved in executing the attack, the level of complexity in mitigating it and the severity of its impact on the targeted system or data. The "Intricacy of Performing the Attack" column denotes the difficulty or complexity an attacker faces in executing a specific threat. This factor considers the technical expertise, resources and effort required to launch the attack successfully.

Conversely, the "Intricacy of Mitigation" column assesses the level of complexity involved in detecting and mitigating the identified threats. It accounts for the challenges and intricacies faced by defenders in identifying and effectively neutralizing these threats once they occur.

Lastly, the "Attack Impact (Severity)" column highlights the potential severity of each threat's

impact on the targeted system or data. This categorization considers the potential harm caused by the attack, encompassing factors such as data integrity compromise, system availability disruption and confidentiality breaches.

Table 15: Summary of the potential vulnerabilities of MAP-T

Attack Name	Intricacy of Performing the Attack	Intricacy of Mitigation	Attack Impact
DoS	Easy	Difficult	Critical
Man-in-the-Middle (MITM)	Average	Difficult	High
Information Disclosure	Average	Average	Medium
Source IP address Spoofing	Easy	Difficult	Critical
Source Port exhaustion	Average	Average	Medium
TCP RST Signal	Easy	Easy	Low
TCP SYNC Flood	Easy	Average	High
Packet's Payload Tampering	Average	Difficult	High
ARP Poisoning	Average	Difficult	High

9.3 Applying STRIDE on MAP-E

As shown in Fig. 32, MAP-E infrastructure also has 11 attacking possibilities, and it has almost identical DFD to MAP-T, which makes the potential security threats (to some extent) similar to the ones that MAP-T faces. However, unlike MAP-T, MAP-E adapts the packet's encapsulation, which will cause relatively different sorts of attacking possibilities. We have summarized the main possible attacks against MAP-E infrastructure:

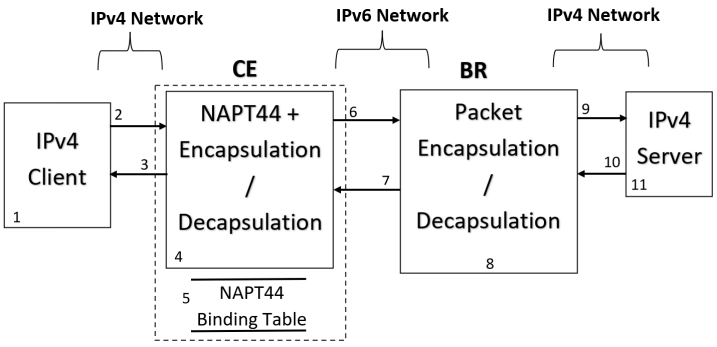

Figure 32: Data Flow Diagram of MAP-E

- DoS: an attacker sends a large amount of traffic to the BR router to overwhelm it and prevent it from processing legitimate traffic, which disrupts network connectivity and services. DoS can also be harmful to the CE router as it overflows the NAPT44 table with too many useless entries and erases the legitimate ones, see Table 4
- Spoofing: an attacker can spoof the IPv6 source address of the CE router and craft an IPv4-embedded IPv6 packet to bypass access controls or launch a reflection or amplification

attack against the BR router and vice versa [37].

- MITM: an attacker can intercept traffic between the CE and BR routers, potentially allowing them to eavesdrop on sensitive information or modify traffic in transit.
- Unauthorized access: an attacker can gain unauthorized access to the CE router and modify or extract sensitive information such as the content of the NAPT44 table (see Table 4). Such data altering attack could lead to wrong encapsulation and forwarding of the IPv4-embedded IPv6 packet to the wrong recipient.

9.4 Attacking Scenarios

As we summarized the list of attacks in Table 15, we have implemented some of those attacks against multiple machines, especially the CE and BR routers.

9.4.1 UDP Packet Spoofing

As shown in Fig. 33, we spoofed the source IPv6 address of the outgoing traffic (CE ⇒ BR). To achieve that, we used a powerful interactive packet manipulation program called Scapy [15], which has multiple use cases such as sniffing and spoofing. The attack was based on a Python script that we wrote (`udp-spoofer.py`) [3].

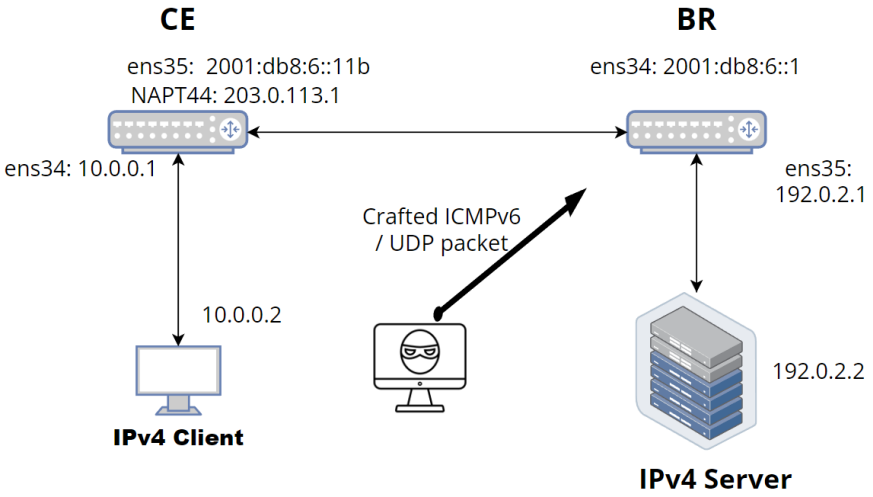

Figure 33: ICMPv6 / UDP Spoofing Attacks

The script sniffs the communication channel between the CE and the BR, looks for UDP traffic (CE ⇒ BR), grabs the source and destination port number, then crafts a new UDP packet with the same details (IP addresses, MAC addresses and port numbers) and sends the crafted packet to the BR router. However, the script alters the payload of the packet with a random text to imitate a real-life attack scenario.

The importance of the attack lies in the fact that the crafted UDP packet was received and processed by the BR router, then forwarded to the IPv4 server. The BR router was deceived and

processed a malicious packet as if it came from a legitimate CE machine. Such action endangers the BR router and the IPv4 server by receiving a stream of data from an attacker while believing it is a trustworthy one.

9.4.2 ICMPv6 Packet Spoofing

A similar procedure was repeated for the ICMP packet: sniffing, monitoring and sending a similar and crafted ICMP packet, while we monitored the ICMP traffic this time.

It is worth mentioning that ICMP packets do not have a port field in their headers. On the other hand, the CE router encompasses the NAPT44 function that assigns ports to the outgoing packets. As a result, MAPT-T has developed a workaround for ICMP packets [57], where it replaces the ICMP packet's ID field with a port number (out of the assigned pool). That's how it filters the ICMP packets against the allocated ports in CE and BR.

As shown in Fig. 33, we managed to spoof the source IPv6 address of the outgoing traffic from CE towards the BR router and send a crafted ICMPv6 packet. This was made possible by another Python script that we wrote for such an attack (`icmpv6-spoofer.py`) [3]. The script also sniffs the communication channel between CE and BR, searches for ICMP traffic, grabs the ICMPv6 packet, which was sent originally by the IPv4 client and saves the ICMP ID field as a variable. Eventually, the script crafts a new ICMPv6 packet with the same saved details (using the same ICMPv6 ID value) of the found ICMPv6 packet and then sends it accordingly to the BR router. The attack went through with these steps: Attacker $\Rightarrow$ BR $\Rightarrow$ IPv4 server $\Rightarrow$ BR $\Rightarrow$ CE $\Rightarrow$ IPv4 client.

In conclusion, the CE and BR routers processed a fake crafted ICMPv6 packet, while the IPv4 client received a reply to a packet that he hadn't sent. This type of attack has a similar effect as the UDP spoofing, but the impact of ports assigning with UDP packets is clearer due to the fact that every UDP packet includes a source port in its header, unlike ICMP packet, where ports are assigned as ID fields value in the ICMP layer of the packet.

9.4.3 DoS Attacks

We carried out a DoS attack to overwhelm the CE / BR machines and exhaust their commutation power. We achieved our goal by applying two methods: `hping3` package and `Scapy` crafting script. As illustrated in Fig. 34, we used the `hping3` package, where we sent a flood of TCP SYN packets from the IPv4 client to the IPv4 server using the below command:

```
hping3 -S --flood -V -p 80 192.0.2.2
```

Furthermore, we monitored the CPU utilization of the CE machine before and after the attack. In Fig. 35, we show that the CPU utilization was very low until we start our attack in the third second. We kept the attacking script running till the tenth second when the CPU was fully utilized.

While the attack was up and running, we tried reaching the IP4 server from the IPv4 client side, we recorded 70% packet loss, which proved that our attack caused more than two-thirds of the

legitimate packets to be drooped.

In the second method for a DoS attack, we used our Scapy script (`tcp-sync-dos.py`) [3]. As illustrated in Fig. 36, the script sniffs the traffic between the CE and the BR machines looking for TCP traffic. When the attacker finds a TCP packet headed in this direction: CE ⇒ BR, it saves the source port as a variable, then crafts a similar TCP sync packet with the same source port number and eventually sends the crafted packet again to the BR machine as if it came from the CE.

The attack went through with the below steps:
Attacker ⇒ BR ⇒ IPv4 server ⇒ BR ⇒ CE ⇒ IPv4 client.

However, the script sends an extreme rate of TCP sync packets (10,000 packets per second), which results in a DoS attack against the BR machine by consuming a large amount of its computation power. The rate of the packets can be modified in the script, assuming that the attacker's machine is able to perform such an attack in terms of its physical resources.

In fact, the attack was relatively successful as it caused 11% packet loss when the IPv4 client tried to ping the IPv4 server during the attack period.

Moreover, we repeated the same attack (with `Scapy` script) with a different approach, where the new script is identical to the original one but it applies a random TCP source port number. We found out that all of the crafted TCP sync packets were dropped by the BR machine because they hadn't used a port from the pre-defined ports range. As a result, we conclude that the source port number has to be extracted from the allocated port range for the attack to succeed. The script can be found under the name of "`tcp-sync-dos-random-sport.py`" in our GitHub repository [3]. The second DoS method can also be considered a spoofing attack, as it spoofs the source IPv6 address of the CE machine and communicates with the BR accordingly.

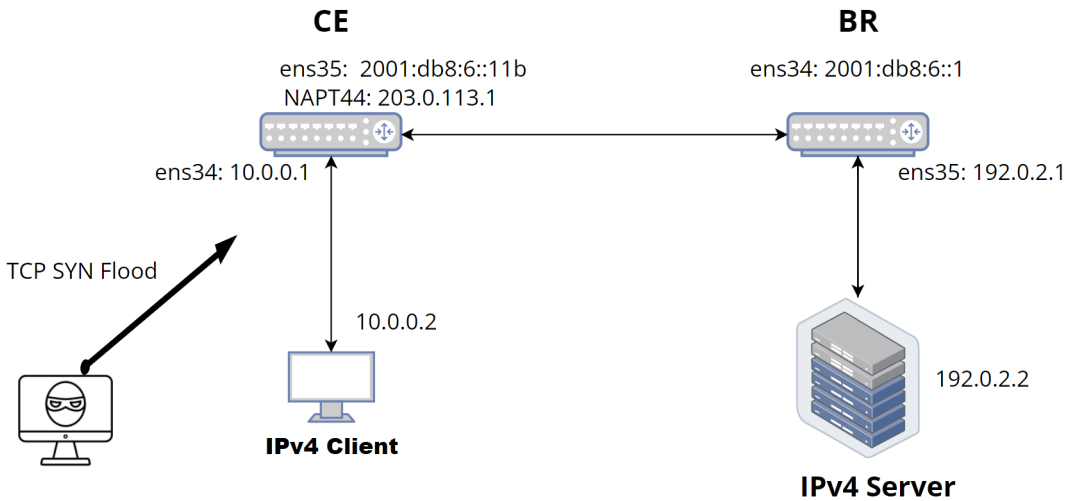

Figure 34: DoS attack using `hping3` package

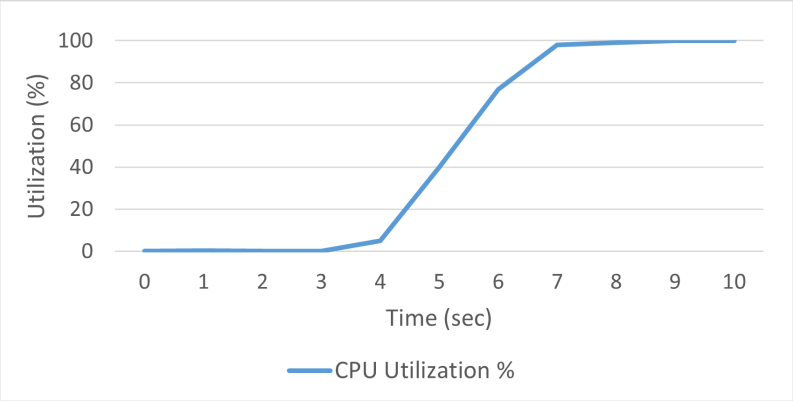

Figure 35: CPU utilization for CE machine

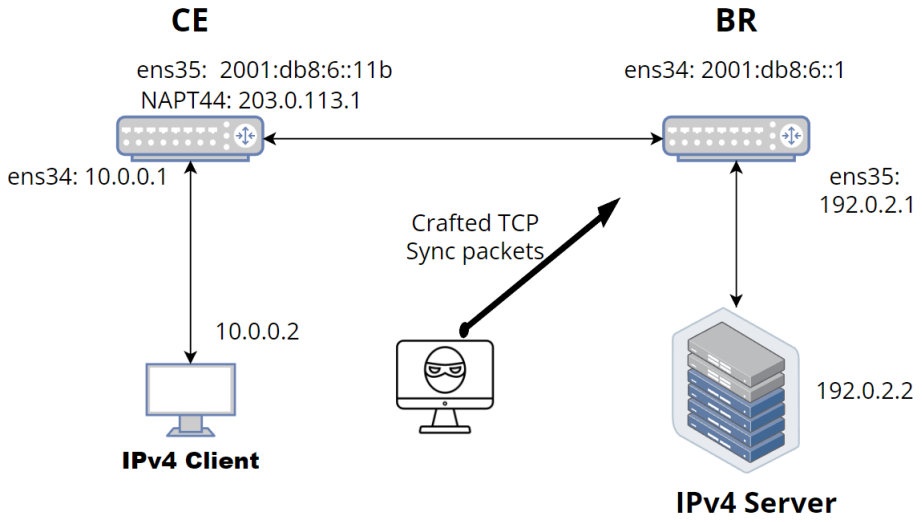

Figure 36: DoS attack using `Scapy` script (`tcp-sync-dos.py`) [3]

9.4.4 Information Disclosure Attack

As shown in Fig. 37, we performed an information disclosure attack using a specific Python script that leverages the Scapy module (`info-disclosure.py`) [3], where we monitored the communication channel between CE and BR machines. The attack was successful. Our script detected the traffic (TCP, UDP or ICMP), then applied a specific Python function (depending on the detected traffic type) and printed out the payload of the TCP or UDP packet. In the case of ICMP traffic, the script printed out the content of the data field of the ICMPv6 layer.

9.4.5 Man-in-the-Middle (MITM) Attack

To perform such an attack, we wrote a Python script (`mitm-attack.py`) [3], which does the followings:

- sniffs the communication channel between CE and BR machines, looks for UDP traffic headed in this direction: CE $\Rightarrow$ BR,

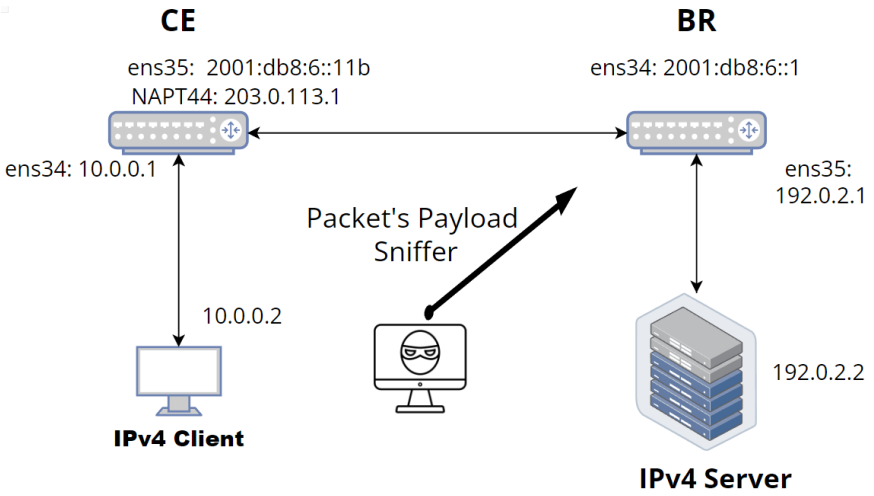

Figure 37: Information Disclosure attack using Scapy script

- when matched traffic is found, the attacker machine saves the packet details (IP addresses, MAC addresses, source port number, etc.),
- crafts a new packet with the same details, but with different UDP payload content,
- sends the newly crafted packet to the BR, claiming to be the CE machine.

The attack went through and the BR router processed the crafted packet (which has the wrong payload) as if it came from the CE machine, which constitutes MITM Attack.

9.4.6 Source Port Exhaustion

The port numbers are stored in 16-bits format, which means that the maximum number of UDP ports equals 65,536 [0-65,535]. This number applies to TCP and UDP ports. By design, every CE router has a built-in pool of assigned port numbers. However, it is recommended not to assign any of the reserved and well-known port numbers [0-1023]. In our test-bed configuration, the CE router is pre-assigned with 2,048 port numbers [55,296-57,343].

As a result, our attack's goal is to exhaust this pool for the CE router to stop functioning to the level that it will not be able to process any incoming packets anymore.

As shown in Fig. 38, we sent an excessive amount of "AAAA" record queries from the attacker toward the IPv4 server. To implement such an attack, we have used a tool called `dns64perf++` [14]. It was designed to be used as a testing tool for measuring the performance of DNS64 servers. As it can send DNS queries at a high rate, we used it as an attacking tool. In the meantime, the assigned ports for CE in our test-bed were 2,048 ports [55,296-57,343]. After installing the `dns64perf++` tool, we applied the below command on the attacker's machine:

```
./dns64perf++v4 192.0.2.2 53 0.0.0.0/5 60000 1 1 60000 400000 0.1
```

There are multiple parameters within the command, and we will provide a detailed explanation of each parameter below:

- 192.0.2.2: IPv4 address of the DNS server, here: destination IP address;
- 53: port number of the DNS server, here: destination port number;
- 0.0.0.0/5: the subnet for generating a label for the DNS queries, here: redundant;
- 60,000: total number of queries to send;
- 1: the burst size;
- 1: the number of threads;
- 60,000: number of ports per thread;
- 400,000: the delay between queries in nanoseconds, which means, we send 2500 queries/s;
- 0.1: the timeout (specifying the maximum waiting time for a response), here: redundant.

After sending an excessive amount of AAAA queries (2500 per second) from the attacker toward the IPv4 server, we managed to exhaust the pool of source port numbers in less than 2 seconds. As a result, we stopped the CE router from further processing incoming packets for several seconds. Fig. 39 shows the wire-shark capture on the CE router (ens35 interface), where it illustrates that in 1.22 seconds (the second column), we managed to exhaust the pool of allocated ports, where the packet flow stops. Only after the 30th second (which is the default timeout for UDP connection), the flow continued because the ports were re-assignable by then.

9.5 Mitigation Methods

9.5.1 ICMP and UDP source address Spoofing Mitigation

The crafted packets via Scapy are generated in a sophisticated manner that makes them look almost identical to the original ones. Therefore, it is technically very hard for the BR and CE routers to

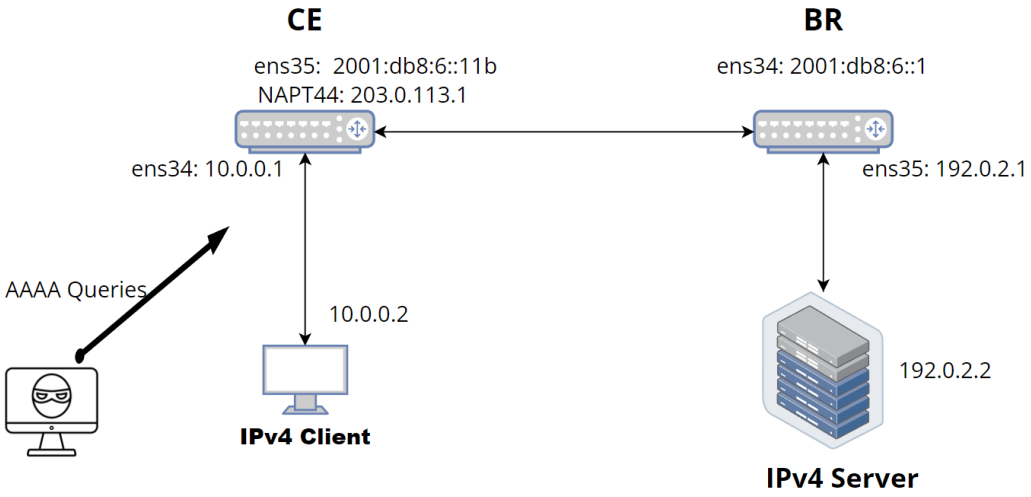

Figure 38: Source port exhaustion attack using AAAA DNS Queries

No.	Time	Src. IP	Dst. IP	Protocol	Src. port	Dst. port
2053	1.213747489	2001:db8:ce:11b:0:cb00:7108:1b	64:ff9b::c000:202	DNS	55356	53
2054	1.214886657	2001:db8:ce:11b:0:cb00:7108:1b	64:ff9b::c000:202	DNS	55357	53
2055	1.228824597	2001:db8:ce:11b:0:cb00:7108:1b	64:ff9b::c000:202	DNS	56367	53
2064	29.998550310	2001:db8:ce:11b:0:cb00:7108:1b	64:ff9b::c000:202	DNS	57152	53

Figure 39: Wireshark Capture on CE ens35.

realize that the received packets are IPv6 address packets with spoofed source addresses, especially when the crafted packets have identical IP and MAC addresses to the original ones.

As a result, a sophisticated tool could be deployed to act as an IDS or an IPS to intercept malicious packets, filter them and eventually drop them. For instance, Snort [79], which is an open-source software package can be deployed and act as IDS and IPS as well.

Snort, with its deep inspection method, can detect the spoofed ICMPv6 packets by examining the packet headers and looking for anomalies that are indicative of spoofing. In particular, Snort can use the following methods to detect spoofed ICMPv6 packets:

- IP address matching: Snort can compare the source IP address of the ICMPv6 packet against the known IP address of the CE router. If the source IP address does not match, Snort can trigger an alert indicating that the packet is likely spoofed.
- TTL value checking: Snort can check the TTL value in the packet header to determine if the packet has traveled a realistic distance from its source. If the TTL value is too low or too high, Snort can infer that the packet has been spoofed.
- Protocol anomalies: Snort can look for other protocol-level anomalies in the packet header that suggest spoofing.

We have installed and configured Snort on the BR router in order to detect and prevent spoofing attacks by leveraging the Snort rule, which we inserted in a file we named “icmp.rules”, which has the following line configured:

```
alert icmp any any -> any any (msg:"Possible  Spoofed Address Detected"; \
  ip6_saddr:!<CE-IPv6-prefix>::/64; icmpv6_type:echo-request; sid:1; rev:1;)
```

Snort looks at a packet that originated from a machine that resides outside the network, if such a packet has a source IPv6 address that belongs to the internal network of the BR ens34 interface, then it will be tagged as a malicious packet. Below is the command that we used to run snort in order to monitor the traffic and detect the filtered ICMP packets:

```
snort -c $snort_path/etc/snort/snort.lua -R icmp.rules -Q --daq \
 afpacket -i "ens34:ens35"
```

However, the mitigation using Snort did not work in this scenario. The reason behind that is that Snort is not able to detect inside jobs, where the attacker is within the network. For more information regarding Snort installation and configuration, please refer to our GitHub repository, where we installed and configured Snort on Debian 10 machine [4]. It is worth mentioning that such software packages might cause lower performance [74].

9.5.2 DoS Mitigation

In general, DoS can be mitigated by having a list of IP addresses of potential malicious adversaries who have committed such acts before and blocking those IP addresses. However, some attackers have the ability to hide their own identities and spoof an innocent machine's IP address. Therefore, the rate-limiting [65] technique can be deployed on the CE router to control/limit the traffic of packets per second that passes through the NIC.

As a result, we wrote a shell script "`rate-limiting.sh`" [3], which we executed on the CE machine to set a severe rate limit of 10 packets/second (for testing purposes). It did prevent CPU exhaustion on the CE machine while under DoS attack. The script is composed of two fundamental rules: the first permits the processing and forwarding of incoming TCP sync packets that occur at a rate of 10 packets per second, whereas the second rule discards all TCP sync packets:

```
/sbin/ip netns exec napt iptables -A FORWARD -p tcp --syn \
  -m limit --limit 10/s -j ACCEPT
/sbin/ip netns exec napt iptables -A FORWARD -p tcp --syn -j DROP
```

9.5.3 Information Disclosure Mitigation

As presented in Fig. 37, we were able to expose the payload of the UDP / TCP packet. Therefore, we wrote a Python script to mitigate such an attacking scenario. The script leverages a cryptographic Python library called "Fernet" for data encryption / decryption purposes [43]. We repeated the attack of Section 9.4.4, which sniffs the communication channel between the CE and the BR router and prints the payload content of the packet. However, this time we send a TCP packet that has an encrypted payload from the IPv4 client toward the IPv4 server. As a result, the attacker was only able to see the encrypted value of the payload, which is useless in that regard. The full Python script for the mitigation method can be found in our GitHub repository under the name of "`tcp-crafter-enc.py`" [3].

9.5.4 MITM Attack Mitigation

Unfortunately, the mitigation of MITM attack is also dependent on installing an IDS / IPS tool such as SNORT, which was also discussed in the mitigation of ICMP/UDP source IP address Spoofing. Please refer to Section 9.5.1. Other methods can be applied to mitigate MITM attacks by enhancing the security of communication and endpoints, such as establishing a secure VPN (Virtual Private Network) connection between the CE and BR routers or enabling MFA (Multi-Factor Authentication).

Multi-factor authentication (MFA) is an essential security measure that augments the login process by necessitating not only the usage of a username and password but also an additional form of verification. This supplementary verification can take various forms, such as the input of a PIN (Personal Identification Number) or the utilization of a unique code transmitted via SMS (Short Messaging Service) to a designated mobile device [71].

9.5.5 UDP Source Port Exhaustion Mitigation

To address this issue, we plan to add more ports. However, this method alone may not be enough to fully mitigate the attack, as the attacking script could still exhaust the new, larger pool of ports within seconds. In addition, adding more ports will not be a practical approach, because there will not be enough ports to allocate for other CEs.

Therefore, to drop DNS queries that have extremely high packet rates, we have utilized the `tc` (traffic control) command to shape the traffic on the ingress interface. Here is a snippet from the script that we run on the CE router to limit DNS queries to 10 per second per IP address and drop any queries that exceed that limit:

```
/sbin/ip netns exec napt tc qdisc add dev to_global root handle 1: htb default 1
/sbin/ip netns exec napt tc class add dev to_global parent 1: classid 1:1 htb \
 rate 1gbit
/sbin/ip netns exec napt tc class add dev to_global  parent 1:1 classid \
 1:10 htb rate 10kbit ceil 10kbit
/sbin/ip netns exec napt tc filter add dev to_global parent 1:0 protocol ip \
  prio 2 u32 match ip protocol 17 0xff match ip dport 53 0xffff match \
  u32 0 0 flowid 1:10 action drop
```

The script does the followings:

- creates a hierarchical token bucket (HTB) qdisc on the "to_global" interface with a default class (1:1) and a child class(1:10) for DNS traffic,
- limits the maximum bandwidth for the parent class to 1Gbps,
- limits the maximum bandwidth for the child class to 10 kilo-bits per second (htb rate 10kbit),
- finally, `tc filter` command applies a filter on the ingress interface to drop packets that exceed the configured limits.

As a result, after running the same attack of Section 9.4.6, the attacker was not able to exhaust the pool of allocated ports within 30 seconds (UDP connection timeout). However, we had to configure a radical rate limit of 10 packets/second. Therefore, it is a trade-off between protecting your infrastructure from DoS/source port exhaustion attacks and causing high latency and service disruption. The full mitigation shell script can be found under the name of "`traffic-controller.sh`" in our GitHub repository [3].

After implementing our attacking scenarios and presenting the corresponding mitigation methods, we summarize them in Table 16, which includes the applied/proposed attack mitigation techniques. Additionally, we classify the severity of the attacks based on the complexity of execution and mitigation.

Table 16: Implemented attacks against MAP-T infrastructure

Attack Name	Complexity of Executing	Complexity of Mitigation	Mitigation Method	Severity
DoS	Easy	Difficult	Rate-Limiting	Critical
Man-in-the-Middle (MITM)	Average	Difficult	VPN / MFA [71]	Critical
Information Disclosure	Average	Average	Payload Encryption	High
ICMP Spoofing	Average	Difficult	IDS / IPS [79]	Medium
UDP Spoofing	Average	Difficult	IDS / IPS [79]	Medium
Source Port exhaustion [14]	Medium	Difficult	Traffic-Control	High

9.6 Conclusion

In this Chapter, we highlighted the importance of the MAP-T IPv6 transition technology, particularly its translation method and showcased its practicality.

We have shown that the Jool software can be applied as a solution for deploying MAP-T and providing customers with IPv4aas (IPv4-as-a-Service), particularly for ISPs operating with an IPv6-only infrastructure. However, it is important to acknowledge that this solution exhibits several security vulnerabilities. Through the utilization of the STRIDE threat modeling technique, we identified various potential attacks, including DoS, IP address spoofing, information disclosure and ARP Cache Poisoning, which pose risks to the MAP-T infrastructure, especially the CE and BR routers.

Furthermore, our testing platform was subjected to different types of attacks, such as IP address spoofing of ICMP/ TCP/UDP packets, TCP SYN DoS and source port exhaustion. While some of these attacks were successfully mitigated, others were not. Specifically, the Jool-based CE and BR routers were found to be inadequate in preventing or detecting a crafted packet with a spoofed source IP address, necessitating the deployment of an additional layer of firewall or IDS / IPS software alongside Jool.

Chapter 10

Proposed Security Analysis Method

10.1 General Structure

Our proposed methodology aims to refine and enhance existing approaches by considering the unique characteristics and topologies inherent to IPv6 transition technologies. We define three layers, on which our new method is based: IPv4aaS Technologies, Place of Statefulness and Individual IPv6 Transition Technologies.

10.1.1 IPv4aaS Technologies

This layer focuses on the five most important IPv4aaS technologies for the reasons explained in the previous chapters. These technologies share a common architectural pattern, consisting of Client, CE device, PE (Provider Edge) device and Server components, resulting in analogous DFDs as shown in Fig. 40. This categorization is a foundational step in our methodology, facilitating a structured approach to evaluate the security posture of IPv6 IPv4aaS transition technology.

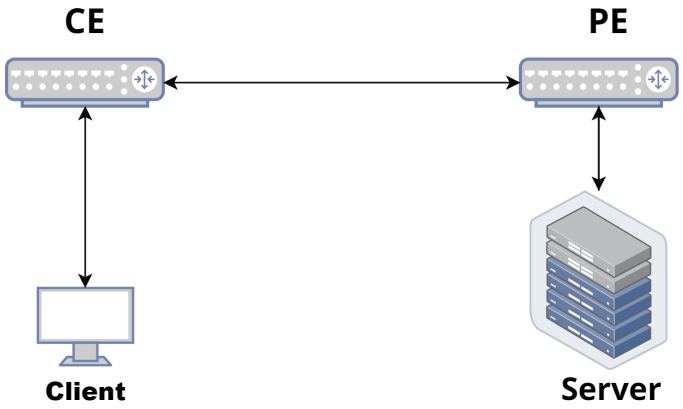

Figure 40: Abstract Layer: Client, CE, PE and Server

Table 17: Classification of IPv6 transition technologies based on the newly proposed method [55]

Technology is stateful on the	Service provider network traversal technology	
	Double translation	Encapsulation / decapsulation
Operator Side	464XLAT	DS-Lite
Client Side	MAP-T	Lw4o6, MAP-E

10.1.2 Place of Statefulness

Generally, 464XLAT and DS-Lite technologies are referred to as "stateful" ones, as their PE devices (PLAT and AFTR) store information about every single network flow that traverses them. Conversely, Lw4o6, MAP-T and MAP-E are referred to as "stateless" ones, as their PE devices (lwAFTR and BR) do not store information about every single network flow that traverses them. However, they are stateful in their CE devices. Table 17 gives a summary of the high-level operation of the five IPv4aaS technologies regarding their statefulness and method used for service provider network traversal. Please refer to [55] for more details.

Therefore, we distinguish the two groups based on where the given technology has a state. The "stateful" ones have a state in their PE device, and they do not have a state in their CE device. The so-called "stateless" ones do not have a state in their PE device, but they have a state in their CE device. The DFDs of the stateful and stateless technologies are shown in Fig. 41 and Fig. 42, respectively.

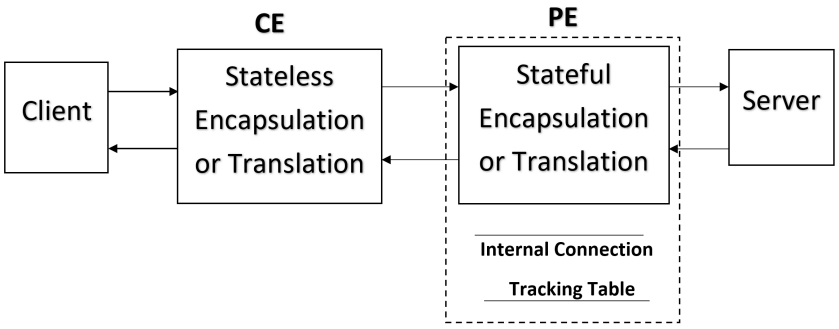

Figure 41: DFD for *stateful PE*-based technologies

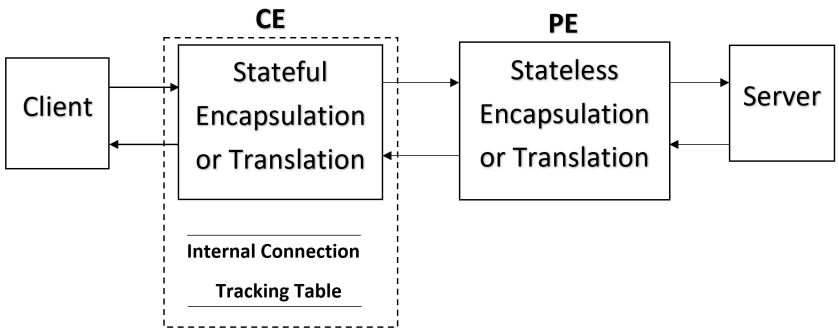

Figure 42: DFD for *stateless PE*-based technologies

10.1.3 Individual IPv6 Transition Technology Analysis

We further refine our analysis by examining each IPv6 transition technology individually. Each technology may be susceptible to different sets of vulnerabilities and may require specific deployment environments. By conducting a detailed analysis of each technology, we can better understand its unique security challenges and requirements.

Fig. 43 illustrates the layered structure of our proposed method for the security vulnerability analysis, which comprises three hierarchical layers of categorization. At the initial layer of general categorization, termed "IPv4aaS Technologies", the selected technologies are grouped into a single overarching category based on their shared architectural pattern.

Subsequently, a more in-depth analysis is conducted, focusing on the place of statefulness of IPv4aaS technologies (CE or PE) and exploring its implications for potential attacks.

Finally, at the third layer, the analysis delves into individual technologies such as DS-Lite, MAP-T, etc., each presenting a unique set of vulnerabilities.

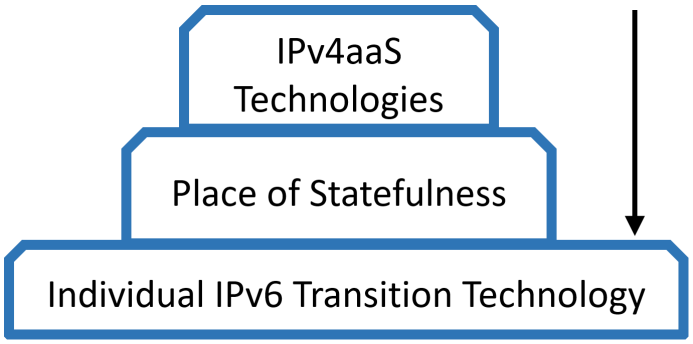

Figure 43: New proposed method hierarchy

10.2 Results and Analysis

While our methodology typically advocates for a top-down analytical approach, we opted to reverse this sequence in this instance due to the availability of results concerning the bottom layer from our prior chapters' results. This approach involved a form of abstraction whereby vulnerabilities at higher layers were inferred by leveraging insights derived from the examination of vulnerabilities at lower layers. This section provides the outcomes of our prior research endeavors, wherein we conducted a comprehensive security analysis of the four leading IPv4aaS technologies. Employing the STRIDE method, we systematically evaluated the security posture of each technology. Furthermore, we constructed a test-bed environment tailored to each technology and executed various attack scenarios targeting the specific CE and PE devices of the given technology.

10.2.1 Method of Abstraction

In this subsection, we define a formal method for abstraction to derive the potential vulnerabilities of a more general level (higher layer) from the potential vulnerabilities discovered at a more specific level (lower layer). Let us denote the set of the considered IPv4aaS technologies (T) by the below formula:

$$T = \{t_i\} = \{464XLAT, DS-Lite, Lw4o6, MAP-E, MAP-T\} \qquad (1)$$

As a result, the set of the DFD elements of technology t_i can be categorized by the below formula:

$$E_i = \{e_{i,j}\} = \{\text{the j-th DFD element of technology } t_i\} \qquad (2)$$

Furthermore, the attack sets of the individual IPv4aaS technologies have already been determined in previous Chapters (6, 7, 8 and 9) and can be denoted by the below formula:

$$A_i = A(t_i, e_{i,j}) = \{a_{i,j}\} = \{\text{ Potential attacks identified for technology } t_i \; at \; DFD \; element \; e_{i,j}\} \quad (3)$$

The general DFD of the IPv4aaS technologies contains only those elements that occur in the DFDs of all IPv4aaS technologies. Formally:

$$E_{\text{IPv4aaS}} = \{e_k\} = \cap_i\{E_i\} \qquad (4)$$

Finally, the potential attacks against the general DFD of the IPv4aaS technologies can be expressed as:
$$A_{\text{IPv4aaS}} = \{a_k\} = \cap_i\{a_{i,k}\} \qquad (5)$$

10.2.2 Individual IPv4aaS Level

At this level, we identified the potential security vulnerabilities within each of the targeted individual IPv4aaS technologies. Such data we already gathered all the data through our stride analysis in Chapters 6, 7, 8 and 9, where we documented our results for 464XLAT, DS-Lite, lw4o6 and MAP-T technologies in Tables 9, 11, 12 and 15 respectively.

10.2.3 Place of Statefulness Level

In this layer, attacks have been classified according to the statefulness of edge-routers (CE & PE), as well as informed by findings from our previous research endeavors documented in [6], [7], [8] and [9]. While recognizing the potential existence of supplementary attack vectors, our analysis emphasizes the inclusion of prominent attacks identified through the STRIDE methodology and practical attack scenarios.

Building on formula (5), which we presented in Section 10.2.1, we continued the same path by splitting the main set into two unique sets of attacks. We differentiate between those attacks based on the statefulness of edge-routers (CE & PE). For example, the set of potential vulnerabilities of *stateful PE*-based technologies can be described as below:

$$A_{\text{IPv4aaS, stateful}} = \cap_{i \in \text{Stateful-PE}}\{a_{i,k}\} \qquad (6)$$

Similarly, the set of potential vulnerabilities of stateless PE-based technologies can be represented as:

$$A_{\text{IPv4aaS, stateless}} = \cap_{i \in \text{ Stateless-PE}} \{a_{i,k}\} \qquad (7)$$

The results of the last two equations are illustrated in Tables 18 and 19 respectively, which is categorized based on the statefulness of the edge-routers (CE & PE).

Table 18: Summary of potential vulnerabilities *stateful PE*-based technologies

Attack Name
DoS against the PE device
Tampering with PE Connection Tracking Table
Source port exhaustion attack against the PE
MITM attack against the PE & CE
Source IP Address spoofing against the PE & CE
Information Disclosure against the CE–PE Traffic
Packet's Payload Tampering
ARP cache poisoning against the PE
Buffer overflow attack against the PE
Network Mapping against the CE
Packet redirection against the CE–PE Traffic

Table 19: Summary of potential vulnerabilities of *stateless PE*-based technologies

Attack Name
DoS against the CE device
Tampering with CE Connection Tracking Table
Source port exhaustion against the CE
Source IP Address spoofing against the PE & CE
Information Disclosure against the CE–PE Traffic
MITM against the CE & PE
TCP Session Hijack against the CE
Network Mapping against the CE
Packet's Payload Tampering against CE–PE Traffic
ARP cache poisoning against the CE
Packet Injection against the CE–PE Traffic
Packet redirection against the CE–PE Traffic

10.2.4 IPv4aaS Level

In this layer, we provide a concise summary of attacks at the IPv4aaS level, where we identify common attack vectors shared among the targeted technologies. These technologies exhibit a parallel structure comprising Client, CE, PE and server components.

To identify the common potential vulnerabilities between the stateful and stateless PE-based technologies, we intersected the two subsets that we gathered from formulas (6) & (7) in Section 10.2.3:

$$A_{\text{IPv4aaS, common}} = A_{\text{IPv4aaS, stateful}} \cap A_{\text{IPv4aaS, stateless}} \quad (8)$$

The results of this intersection, which include only the common attacks among the two subsets, are illustrated in Table 20. Those results are also represented by intersecting the common attacks from Table 18 and 19, where they can be considered as outcomes of our current analysis. As a result, Table 20 shows common attacks identified across several technologies on the IPv4aaS level.

The severity categorization presented in Table 20 was determined through an assessment of two pivotal metrics: the intricacy inherent in executing the attack and the complexity involved in its mitigation. These metrics were informed by empirical data acquired from our practical engagements with attack scenarios and mitigation strategies, as documented in our previous research contributions [6], [7], [8] and [9].

Table 20: Summary of the Potential Vulnerabilities at IPv4aas Level

Attack Name	Targeted Area	Severity
Source IP Address Spoofing	CE & PE	Average
Information Disclosure	CE–PE Traffic	Average
Traffic Packet's Payload Tampering	CE–PE Traffic	High
MITM	PE & CE	Average
Packet redirection	CE–PE Traffic	Average
Network Mapping	CE	Low

10.3 Conclusion

We recognize that many IPv6 transition technologies have been proposed over time; however, the landscape has shifted. Earlier methods like 6to4, Teredo and 6rd have become largely obsolete or see limited adoption today due to security issues and reduced usage. Our focus was on IPv4aaS technologies, which are better suited to handle IPv4 address exhaustion and support IPv6 address adoption. These are expected to be widely used in the coming years, ensuring that our methodology addresses the most relevant security concerns in real-world scenarios.

As we examined various IPv4aaS technologies, we discovered common vulnerabilities across all of them. We found that, while these vulnerabilities are similar across these technologies, their impact and location differ; some vulnerabilities affect the ISP side, while others affect the customer side. Although some technologies are marketed as stateless, they actually maintain states within their infrastructure. For instance, lw4o6, MAP-T and MAP-E, while labeled as stateless, hold states at the customer edge. Consequently, DoS attacks in such cases are more likely to affect specific customers rather than the entire system.

In contrast, a DoS attack on a technology with statefulness at the ISP side, like DS-Lite or 464XLAT, can impact all subscribers, causing more extensive damage and network downtime. In Table 21, we summarized the impact (severity) of the DoS attack on the customer and ISP sides based on the statefulness of the technology itself.

In synthesis, our research underscores the complexities and opportunities inherent in IPv6 transition technologies. As network administrators, policymakers and researchers strive to embrace IPv6, addressing

Table 21: DoS impact based on the technology statefulness

State at Provider Side	DoS Impact at client side	DoS Impact at ISP side
Stateful	Low	High
Stateless	High	Low

security concerns and refining implementations will be paramount.

While each IPv6 transition technology has its own set of advantages and disadvantages, including considerations related to security, it is important to acknowledge the nuanced nature of these evaluations. However, after careful analysis, we have concluded that stateless technologies are the optimum choice for security reasons.

As demonstrated in Table 21, stateless technologies offer distinct advantages over stateful ones. A stateless approach, exemplified in this context, ensures heightened resilience against certain attacks such as DoS. Unlike stateful implementations, which concentrate state information on centralized devices, stateless architectures distribute this information across the network. Consequently, in the event of a DoS attack, targeting a stateful central device would have broader repercussions, potentially affecting a larger number of end-users due to packet loss or service shutdown. In contrast, stateless architectures mitigate such risks by dispersing the impact, limiting it to individual clients rather than compromising the entire network.

Chapter 11

Discussion

11.1 Research Gaps

The study of IPv6 transition technologies, including 464XLAT, DS-Lite, lw4o6, MAP-T and MAP-E, has revealed several critical gaps in the existing research landscape. Two primary deficiencies stand out prominently:

11.1.1 Absence of Comprehensive Security Analysis

One critical research gap lies in the absence of comprehensive security analyses conducted for the identified IPv6 transition technologies. Despite their increasing relevance in modern network infrastructures, there remains a notable dearth of in-depth examinations into the security posture of those technologies. Such analyses are essential for identifying potential vulnerabilities, understanding attack vectors and devising robust security measures to safeguard network infrastructures against emerging threats.

11.1.2 Lack of Real Built Test-Beds

Another significant research gap pertains to the absence of real built test-beds tailored to the specific nuances of IPv6 transition technologies. While theoretical analyses and simulations have provided valuable insights, the lack of practical experimentation on functional test-beds limits our ability to comprehensively evaluate the real-world performance and security implications of these technologies. The absence of such test-beds hinders researchers, network administrators and policymakers from conducting empirical assessments, identifying vulnerabilities and implementing effective mitigation strategies. There were some trials to build such test-bed, for example in [52], where the authors built a test-bed on Debian-based machines, utilizing Jool open-source software [41] to implement CE and BR routers. The authors tested the scalability of the performance of CE and BR routers while adding more CPU cores and monitoring the performance. However, the focus of the authors was merely on performance and not on security.

11.1.3 Lack of Research on Reliable Open-Source Software

A significant gap is the absence of comprehensive research surveying reliable open-source software for deploying IPv6 transition technologies. The lack of systematic evaluations hampers network administrators and researchers in selecting suitable software for deployment. Assessing open-source options could provide valuable insights into their features, performance and security considerations, facilitating more informed

decision-making. This gap impedes the efficient deployment and adoption of IPv6 transition technologies, highlighting the need for further investigation in this area. There is a recent survey on IPv4aaS technologies, their implementations and performance analysis [25] , which lists several implementations, however it does not contain an analysis if the listed implementations are still usable and some of them are way obsolete (more than ten years old and not maintained). These research gaps underscore the pressing need for further investigation and scholarly inquiry into the security implications and operational challenges associated with IPv6 transition technologies. Addressing these gaps will not only enhance our understanding of these technologies but also contribute to the development of more secure and resilient network infrastructures in the face of evolving cybersecurity threats.

11.2 Finding Benefits

As network administrators, policymakers and researchers strive to embrace IPv6, addressing security concerns and refining implementations will be paramount. By addressing the pressing security concerns associated with IPv6 adoption, my research provides invaluable insights for network administrators, policymakers and researchers. Through the examination of five distinct technologies, each revealing unique vulnerabilities, my work offers practical guidance for safeguarding network infrastructure. For instance, in the deployment of DS-Lite, where DoS attacks pose a significant threat, system administrators might conclude that there is a necessity of implementing tailored anti-DoS measures, such as sourcing traffic from trusted third-party providers offering "Clean Traffic" under defined SLA (Service Level Agreement). Moreover, by shedding light on overlooked areas of security vulnerabilities within computer network security research, my work contributes to enhancing the overall understanding and mitigation of risks associated with IPv6 transition technologies.

Moreover, I underscore the pioneering nature of my research methodology by applying the STRIDE methodology to analyze potential attacking points within five targeted IPv6 transition technologies. This approach represents the first-ever endeavor globally to systematically assess the security implications of these technologies using such a comprehensive framework. By emphasizing this unprecedented aspect of my research, I highlight its significance in addressing a critical gap in the existing literature and providing valuable insights for both the scientific community and network engineers. Moreover, by being the first to conduct such a vulnerability analysis, my research offers a foundation upon which future studies can build, potentially leading to advancements in understanding and securing IPv6 transition technologies.

In addition to providing practical guidance for safeguarding network infrastructure, my research makes a valuable contribution to the scientific community by addressing security issues that facilitate a safer transition to IPv6. By identifying and mitigating vulnerabilities within IPv6 transition technologies, my work alleviates concerns about the complexities associated with IPv4 and its complicated NAT processes. This not only enhances the security posture of networks but also fosters a smoother transition to IPv6, ultimately benefiting network administrators, policymakers and researchers alike.

Finally, through my contribution to the discussion on IPv6 transition, I pave the path toward improved network connectivity and resilience within an ever-evolving technological landscape.

Results Summary

My research has delved into the intricacies of various IPv6 transition technologies, unveiling their functionalities, efficiencies and potential vulnerabilities. By scrutinizing these technologies in detail, I have contributed insights that advance the understanding of IPv6 adoption and its challenges. Throughout my investigation, I explored the capabilities of the five prominent IPv6 transition technologies. Each one of them presents a distinctive approach to the IPv6 transition, accompanied by its own strengths and concerns.

In this dissertation, I've contributed the following advancements to this particular field.

THESIS 1.1 Utilizing the STRIDE method, I conducted a comprehensive analysis of vulnerabilities inherent in the 464XLAT technology. This scrutiny revealed the existence of multiple vulnerabilities, including DoS and spoofing. For a detailed compilation of potential attacks identified, please refer to Table 9.

THESIS 1.2 I established a 464XLAT test-bed using Linux-based virtual machines and TAYGA open-source software. While conducting a DoS attack on this infrastructure, I discovered an anomaly in the Netfilter framework, detailed in [J2]. This issue pertained to inconsistencies in the IP masquerade feature's functionality during high-frequency ping requests from eight clients to the IPv4 server via the CLAT and the PLAT using the `hping3` command. Subsequently, I reported this anomaly to the Netfilter framework developers for further investigation.

THESIS 1.3: I successfully countered the DoS attack by implementing a defense strategy on the CLAT machine using `iptables` rules. This strategy involved the imposition of a strict rate-limiting mechanism, which was achieved through the creation of a filter containing two key rules.

THESIS 2.1 I examined the vulnerabilities of the DS-Lite technology using the STRIDE method and found out that it contains several vulnerabilities such as DoS, tampering and source IP address spoofing. The full list of potential vulnerabilities can be found in Table 11.

THESIS 2.2 As shown in Fig. 10, I built a test-bed for DS-Lite technology, employing Linux-based virtual machines. Utilizing the IPIP6 tunnel, I facilitated connectivity between the B4 and the AFTR router, encapsulating and transmitting packets through this conduit. Subsequently, I executed two attacks successful against the DS-Lite infrastructure: source IP address spoofing and source port exhaustion.

THESIS 3.1 Through the utilization of the STRIDE method, I conducted a comprehensive assessment of vulnerabilities within the Lw4o6 technology, uncovering multiple weaknesses. I gathered security vulnerabilities of the lw4o6 in Table 12, where I rated the attack's severity, considering how complex each is to carry out and how hard it is to mitigate it.

THESIS 3.2 I established a test-bed, detailed in [J4], dedicated to evaluating the lw4o6 technology, using the Snabb open-source software [31]. This involved the creation of a tunnel connecting the primary routers integral to the lw4o6 framework, namely lwB4 and lwAFTR. Furthermore, I configured and assembled the lwAFTR router by employing the sophisticated setup procedures inherent in Snabb.

THESIS 3.3: I executed several attacks against lw4o6 main routers such as DoS, Information Disclosure, Spoofing, Source Port Exhaustion, etc. The full list of implemented attacks can

be found in Table 14. Furthermore, I demonstrated the application of Scapy, which is an interactive packet manipulation program, in attacking the lw4o6 infrastructure.

THESIS 3.4: As illustrated in Fig. 30, I conducted a TCP Session Hijacking attack by employing two attacking machines against lw4o6 infrastructure using the Scapy software [J4], where an unauthorized party intercepts and takes control of an established TCP connection between two communicating entities, potentially gaining unauthorized access and control over the communication or data exchange.

THESIS 3.5: Upon subjecting the lw4o6 infrastructure to eight distinct attack scenarios, I proceeded to implement diverse mitigation strategies targeting the majority of these scenarios. For instance, methods such as rate limiting were employed to counteract DoS, while an additional encryption layer was introduced to mitigate information disclosure threats. Subsequently, I summarized the results of my attacks and their corresponding mitigation approaches within Table 14.

THESIS 4.1 I conducted an in-depth analysis of MAP-T technology vulnerabilities employing the STRIDE method, uncovering multiple susceptibilities including potential MITM, ARP Cache Poisoning, etc. Subsequently, a consolidated summary outlining these identified potential attacks has been presented in Table 15.

THESIS 4.2 I built a dedicated test-bed to evaluate the MAP-T technology, utilizing Linux-based virtual machines as the foundational framework. Within the MAP-T topology, the critical components include two primary routers, referred to as CE and BR. The implementation of the Jool open-source IPv4/IPv6 translator [41] was pivotal in establishing this setup.

THESIS 4.3 I conducted various attacks targeting MAP-T main routers, including DoS, Information Disclosure, source port exhaustion, etc. Table 16 contains the complete list of executed attacks. The majority of these attacks showcased the utilization of Scapy software in targeting the MAP-T infrastructure.

THESIS 4.4: After executing attack scenarios, I used Traffic Control, rate-limiting and encryption to counter information disclosure, source port exhaustion and DoS threats. For unmitigated attacks, I suggested IDS/IPS tools like Snort or Suricata for deep traffic filtering. I then assessed attack severity based on complexity metrics, outlined in [J5] and summarized them in Table 16.

THESIS 5.1 I proposed a new security analysis methodology that aims to refine and enhance existing approaches by considering the unique characteristics and topologies inherent to IPv6 transition technologies. In this method, I defined three layers, such as the place of statefulness for the PE device within the topology as shown in Fig. 43.

THESIS 5.2 Building on the new proposed method, I defined a formal method for abstraction to derive the potential vulnerabilities of a more general level (higher layer) from the potential vulnerabilities discovered at a more specific level (lower layer).

THESIS 5.3 Based on the proposed abstraction method, I summarized the potential vulnerabilities of (statefull & stateless) PE-based technologies as shown Tables 18 and 19. Furthermore, I intersected the common attacks among those two horizons and presented them in Table 20.

List of own Publications

Conferences

[C1] A. Al-Azzawi, “Plans for the security analysis of IPv4aaS technologies”, 14th International Symposium on Applied Informatics and Related Areas, University of Óbuda, Székesfehérvár, Hungary (2019) pp. 101–105.

[C2] A. Al-Azzawi and G. Lencse, “Towards the Identification of the Possible Security Issues of the 464XLAT IPv6 Transition Technology,” 43rd International Conference on Telecommunications and Signal Processing (TSP), Milan, Italy, 2020, pp. 439-444, doi: 10.1109/TSP49548.2020.9163487.

[C3] A. Al-Azzawi and G. Lencse, “Testbed for the Security Analysis of the 464XLAT IPv6 Transition Technology in a Virtual Environment”, 44th International Conference on Telecommunications and Signal Processing (TSP), Brno, Czech Republic, 2021, pp. 5-9, doi: 10.1109/TSP52935.2021.9522598.

Journals

[J1] A. Al-Azzawi, “Towards the Security Analysis of the Five Most Prominent IPv4aaS Technologies”, Acta Technica Jaurinensis 13 (2) (2020) 85–98. doi:10.14513/actatechjaur.v13.n2.530

[J2] A. Al-Azzawi and G. Lencse, “Identification of the Possible Security Issues of the 464XLAT IPv6 Transition Technology”, Infocommunications Journal 13 (4) (2021) 10–18. doi:10.36244/ICJ.2021.4.2. (WOS, Scopus Q3)

[J3] Al-Azzawi and G. Lencse, “Analysis of the Security Challenges Facing the DS-Lite IPv6 Transition Technology”, Electronics 12 (10) (2023). doi:10.3390/electronics12102335 (WOS IF =2.6, Scopus Q2)

[J4] A. Al-Azzawi and G. Lencse, “Lightweight 4over6 Test-bed for Security Analysis”, Infocommunications Journal 15 (3) (2023) 30–41. doi:10.36244/ICJ.2023.3.4. (WOS IF =0.9, Scopus Q3)

[J5] A. Al-Azzawi and G. Lencse, “Security Analysis of the MAP-T IPv6 Transition Technology”. The Computer Journal (2024), doi:10.1093/comjnl/bxae059. (WOS IF =1.5, Scopus Q2)

[J6] A. Al-Azzawi and G. Lencse, “Methodology for the security analysis of IPv4-as-a-Service IPv6 transition technologies”. The Computer Journal (2025), doi:10.1093/comjnl/bxaf048. (WOS IF =1.5, Scopus Q2)

Acronyms

AFTR	Address Family Transition Router 6, 100
ARP	Address Resolution Protocol 63, 66, 76, 78, 90, 101
B4	Basic Bridging Broadband 6, 100
BR	Border Relay 16, 101
CAPEC	Common Attack Patterns Enumeration and Classification 20
CE	Customer Edge 16, 91, 101
CLAT	Customer-Side Translator 5, 100
CPE	Customer Premises Equipment 6, 8, 12, 55
CPU	Central Processing Unit 29–31, 38, 55, 67, 71, 78, 82, 88
DFD	Data Flow Diagram 41, 42, 59, 61, 62, 75
DHCP	Dynamic Host Configuration Protocol 5
DNS	Domain Name System 5, 69, 73, 86, 89
DoS	Denial of Service i, 2, 24, 42–44, 50–52, 59, 60, 100, 101
DS-Lite	Dual-Stack Lite i, 2
FoS	Failure of Service 42, 50, 76
HTTPS	Hypertext Transfer Protocol Secure 28
ICMP	Internet Control Message Protocol 28, 53, 54, 57, 68, 73, 74
IDS	Intrusion Detection System 73, 74, 87, 88, 90, 101
IoT	Internet of Things i, 3
IPS	Intrusion Prevention System 73, 87, 88, 90, 101
ISP	Internet Service Provider 6, 8, 12, 58, 96
Lw4o6	Light weight 4over6 i, 2
MAC	Media Access Control 63, 73, 76, 78, 81, 85, 87
MAP-E	Mapping of Address and Port using Encapsulation i, 2
MAP-T	Mapping of Address and Port using Translation i, 2
MITM	Man-in-the-middle 27, 53, 62, 64, 76, 77, 85, 101

NAPT44 Network Address and Port translation between private and public IPv4 addresses 16, 17
NIC Network Interface Controller 54, 59, 88

PE Provider Edge 91–93
PIN Personal Identification Number 88
PLAT Provider-Side Translator 5, 6, 100
PSID Port Set Identifier 16, 64–66

SLA Service Level Agreement 99
SMS Short Messaging Service 88
SNMP Simple Network Management Protocol 26
SSH Secure Shell 45, 57

TCP Transmission Control Protocol 24, 42, 44, 45, 50, 56, 64, 66–68, 70, 71, 74, 101
TLS Transport Layer Security 72
TTL Time To Live 50, 51, 63, 76, 87

UDP User Datagram Protocol 27, 56, 57, 59, 64, 68–70, 72, 74

WAN Wide-Area Network 9
WIFI Wireless Fidelity 50

Bibliography

[1] Abad, C. L. and Bonilla, R. I. [2007], An Analysis on the Schemes for Detecting and Preventing ARP Cache Poisoning Attacks, *in* '27th International Conference on Distributed Computing Systems Workshops (ICDCSW'07)', IEEE, pp. 60–60.

[2] Al-Azzawi, A. [2022*a*], 'Lightweight 4 over 6 test-bed'.
URL: https://github.com/ameen-mcmxc/lw4o6-automation

[3] Al-Azzawi, A. [2022*b*], 'MAP-T Jool Testbed Installation'.
URL: https://github.com/ameen-mcmxc/MAP-T-builder

[4] Al-Azzawi, A. [2023], ' Snort Installation and Configuration using Ansible Automation Software '.
URL: https://github.com/ameen-mcmxc/SNORT-Automation

[5] Al-Azzawi, A. and Lencse, G. [2020], Towards the Identification of the Possible Security Issues of the 464XLAT IPv6 Transition Technology., *in* 'TSP', pp. 439–444.

[6] Al-Azzawi, A. and Lencse, G. [2021], 'Identification of the Possible Security Issues of the 464XLAT IPv6 Transition Technology', *Infocommunications Journal* **13**(4), 10–18.

[7] Al-Azzawi, A. and Lencse, G. [2023*a*], 'Analysis of the Security Challenges Facing the DS-Lite IPv6 Transition Technology', *Electronics* **12**(10).

[8] Al-Azzawi, A. and Lencse, G. [2023*b*], 'Lightweight 4over6 Test-bed for Security Analysis', *Infocommunications Journal* **15**(3), 30–41.

[9] Al-Azzawi, A. and Lencse, G. [2024], 'Security Analysis of the MAP-T IPv6 Transition Technology', *Computer Journal* .

[10] Al-hamadani, A. and Lencse, G. [2021], Design of a Software Tester for Benchmarking Lightweight 4over6 Devices, *in* '2021 44th International Conference on Telecommunications and Signal Processing (TSP)', pp. 157–161.

[11] Al-hamadani, A. and Lencse, G. [2022], 'Towards Implementing a Software Tester for Benchmarking MAP-T Devices', *Infocommunications Journal* **14**(3), 45–54.

[12] Bagnulo, M., Matthews, P. and van Beijnum, I. [2011], Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers, RFC 6146, IETF.

[13] Bagnulo, M., Sullivan, A., Matthews, P. and Van Beijnum, I. [2011], DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers, RFC 6147, IETF.

[14] Bakai, D. [2016], 'DNS64perf++ Measurement Tool'.
URL: https://github.com/bakaid/dns64perfpp

[15] Biondi, P. [2003], 'Scapy, Packet Manipulation Program'. Accessed: 2023-01-02.
URL: https://scapy.net/index

[16] Boucadair, M., Jacquenet, C. and Sivakumar, S. [2019], A YANG Data Model for Dual-Stack Lite (DS-Lite), RFC 8513, IETF.

[17] Carpenter, B. and Jung, C. [1999], Transmission of ipv6 over ipv4 domains without explicit tunnels, RFC 2529, IETF.

[18] Center, C. [2013], 'Open Source MAP Implementation'.
URL: https://github.com/cernet/MAP

[19] Conta, A. and Deering, S. [1998], Generic Packet Tunneling in IPv6 Specification, RFC 2473, IETF.

[20] Cordeiro, E., Carnier, R. and Moreiras, A. [2015], Experience with Testing of Mapping of Address and Port Using Translation (MAP-T), RFC 7703, IETF.

[21] Cui, Y., Sun, Q., Boucadair, M., Tsou, T., Lee, Y. and Farrer, I. [2015], Lightweight 4over6: An extension to the dual-stack lite architecture, RFC 7596, IETF.

[22] Deng, S., Gao, X., Lu, Z. and Gao, X. [2018], 'Packet Injection Attack and Its Defense in Software-Defined Networks', *IEEE Transactions on Information Forensics and Security* **13**(3), 695–705.

[23] Durand, A., Droms, R., Woodyatt, J. and Lee, Y. [2011], Dual-stack lite broadband deployments following IPv4 exhaustion, RFC 6333, IETF.

[24] D'yab, O. and Lencse, G. [2022], Testbed for the Comparative Analysis of DS-Lite and Lightweight 4over6 IPv6 Transition Technologies, *in* '2022 45th International Conference on Telecommunications and Signal Processing (TSP)', IEEE, pp. 371–376.

[25] D'yab, O. [2022], 'A Comprehensive Survey on the Most Important IPv4aaS IPv6 Transition Technologies, their Implementations and Performance Analysis', *Infocommunications Journal* **14**(3), 35–44.

[26] Engelhardt, J. and Bouliane, N. [2011], 'Writing Netfilter Modules', *Revised, February* **7**.

[27] FDio [2016], 'VPP Software'.
URL: https://github.com/FDio/vpp

[28] Ferdous, M. S., Chowdhury, F. and Acharjee, J. C. [2007], An Extended Algorithm to Enhance the Performance of the Current NAPT, *in* 'International Conference on Information and Communication Technology', pp. 315–318.

[29] Fu, Y., Jiang, S., Dong, J. and Chen, Y. [2016], Dual-Stack Lite (DS-Lite) Management Information Base (MIB) for Address Family Transition Routers (AFTRs), RFC 7870, IETF.

[30] Garcia, D. [n.d.], 'Lightweight 4-over-6: a compelling IPv4+IPv6 architecture'. Accessed: 2023-04-24.
URL: https://www.igalia.com/project/lw4o6

[31] Garcia, D. P. [2013], 'Snabb Explained in less than 10 Minutes'. Accessed: 2022-09-04.
URL: http://blogs.igalia.com/dpino/2017/11/13/snabb-network-toolkit/

[32] García, D. P. [n.d.], 'The B4 network function'. Accessed: 2022-08-15.
URL: https://blogs.igalia.com/dpino/2018/02/15/the-b4-network-function/

[33] Georgescu, M., Hazeyama, H., Kadobayashi, Y. and Yamaguchi, S. [2014], Empirical analysis of IPv6 transition technologies using the IPv6 Network Evaluation Testbed, *in* 'Testbeds and Research Infrastructure: Development of Networks and Communities: 9th International ICST Conference, TridentCom 2014, Guangzhou, China, May 5-7, 2014, Revised Selected Papers 9', Springer, pp. 216–228.

[34] Georgescu, M., Hazeyama, H., Okuda, T., Kadobayashi, Y. and Yamaguchi, S. [2016], The STRIDE Towards IPv6: A Comprehensive Threat Model for IPv6 Transition Technologies, *in* '2nd International Conference on Information Systems Security and Privacy'.

[35] Georgescu, M., Pislaru, L. and Lencse, G. [2017], Benchmarking methodology for IPv6 transition technologies, RFC 8219, IETF.

[36] Gil, T. M. and Poletto, M. [2001], A Data-Structure for Bandwidth Attack Detection, *in* '10th USENIX Security Symposium (USENIX Security 01)'.

[37] Gupta, B. B., Dahiya, A., Upneja, C., Garg, A. and Choudhary, R. [2020], 'A Comprehensive Survey on DDoS Attacks and Recent Defense Mechanisms', *Handbook of Research on Intrusion Detection Systems* **13**, 186–218.

[38] Hankins, D. and Mrugalski, T. [2011], Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Option for Dual-Stack Lite, RFC 6334, IETF.

[39] Hat, R. [2012], 'Ansible, Open Source IT Automation Tool'.
URL: https://www.ansible.com/

[40] Hong, H., Choi, H., Kim, D., Kim, H., Hong, B., Noh, J. and Kim, Y. [2017], When Cellular Networks Met IPv6: Security Problems of Middleboxes in IPv6 Cellular Networks, *in* 'IEEE European Symposium on Security and Privacy (EuroS&P)', pp. 595–609.

[41] ITESM [2014], 'Jool, Open Source IPv4/IPv6 Translator'.
URL: https://www.jool.mx/en/index.html

[42] ITU/APNIC/MICT [2016], 'Securing the Transition Mechanisms'.
URL: https://www.itu.int/en/ITU-D/Regional-Presence/AsiaPacific/Documents/s11-ipv6-securingtransitionmechanisms.pdf

[43] Jain, A. and De, P. [2021], Enhancing Database Security for Facial Recognition using Fernet Encryption Approach, *in* '2021 5th International Conference on Electronics, Communication and Aerospace Technology (ICECA)', IEEE, pp. 748–753.

[44] Jara, A. J., Ladid, L. and Gómez-Skarmeta, A. F. [2013], 'The Internet of Everything through IPv6: An Analysis of Challenges, Solutions and Opportunities.', *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.* **4**(3), 97–118.

[45] Keromytis, A. D. [2011], *Buffer Overflow Attacks*, Springer US, Boston, MA, pp. 174–177.

[46] Krishnan, S., Thaler, D. and Hoagland, J. [2011], Security concerns with IP tunneling, RFC 6169, IETF.

[47] Kristiyanto, Y. and Ernastuti, E. [2020], 'Analysis of Deauthentication Attack on IEEE 802.11 Connectivity Based on IoT Technology Using External Penetration Test', *CommIT (Communication and Information Technology) Journal* **14**(1), 45–51.

[48] Lee, Y., Maglione, R., Williams, C., Jacquenet, C. and Boucadair, M. [2013], Deployment considerations for dual-stack lite, RFC 6908, IETF.

[49] LENCSE, G. and BAKAI, D. [2017], 'Design and Implementation of a Test Program for Benchmarking DNS64 Servers', *IEICE Transactions on Communications* **E100.B**(6), 948–954.

[50] Lencse, G. and Kadobayashi, Y. [2018], 'Methodology for the identification of potential security issues of different IPv6 transition technologies: Threat analysis of DNS64 and stateful NAT64', *Computers & Security* **77**, 397–411.

[51] Lencse, G. and Kadobayashi, Y. [2019], 'Comprehensive survey of IPv6 transition technologies: A subjective classification for security analysis', *IEICE Transactions on Communications* **102**(10), 2021–2035.

[52] Lencse, G. and Nagy, N. [2022], 'Towards the scalability comparison of the Jool implementation of the 464XLAT and of the MAP-T IPv4aaS technologies', *International Journal of Communication Systems* **35**(18), e5354.

[53] Lencse, G., Palet Martinez, J., Howard, L., Patterson, R. and Farrer, I. [2022], Pros and Cons of IPv6 Transition Technologies for IPv4-as-a-Service (IPv4aaS), RFC 9313, IETF.

[54] Lencse, G. and Répás, S. [2013], Performance analysis and comparison of the TAYGA and of the PF NAT64 implementations, *in* '36th International Conference on Telecommunications and Signal Processing (TSP)', pp. 71–76.

[55] Lencse, G. and Ádám Bazsó [2024], 'Benchmarking methodology for IPv4aaS technologies: Comparison of the scalability of the Jool implementation of 464XLAT and MAP-T', *Computer Communications* **219**, 243–258.

[56] Li, X., Bao, C., Dec, W., Troan, O., Matsushima, S. and Murakami, T. [2015*a*], Mapping of Address and Port using Translation (MAP-T), RFC 7599, IETF.

[57] Li, X., Bao, C., Dec, W., Troan, O., Matsushima, S. and Murakami, T. [2015*b*], Mapping of address and port using translation (MAP-T), RFC 7599, IETF.

[58] Lutchansky, N. [2011], 'TAYGA: Simple, no-fuss NAT64 for Linux'.
URL: http://www.litech.org/tayga

[59] Mahajan, R., Bellovin, S. M., Floyd, S., Ioannidis, J., Paxson, V. and Shenker, S. [2002], 'Controlling high bandwidth aggregates in the network', *ACM SIGCOMM Computer Communication Review* **32**(3), 62–73.

[60] *Making a Synthesis Emulation in IoT ERA Possible, Starbed5 Project. StarBED5 Project website* [n.d.]. Accessed: 2022-09-05.
URL: https://starbed.nict.go.jp/en/equipment/

[61] Mawatari, M., Kawashima, M. and Byrne, C. [2013], 464XLAT: Combination of Stateful and Stateless Translation, RFC 6877, IETF.

[62] McHardy, P. [2001], 'Netfilter Conntrack Performance Tweaking '.
URL: https://wiki.khnet.info/index.php/Conntrack_tuning

[63] Mhaskar, N., Alabbad, M. and Khedri, R. [2021], 'A Formal Approach to Network Segmentation', *Computers & Security* **103**, 102162.

[64] Naval, S., Laxmi, V., Rajarajan, M., Gaur, M. S. and Conti, M. [2015], 'Employing program semantics for malware detection', *IEEE Transactions on Information Forensics and Security* **10**(12), 2591–2604.

[65] Noormohammadpour, M. and Raghavendra, C. S. [2018], 'Datacenter Traffic Control: Understanding Techniques and Tradeoffs', *IEEE Communications Surveys & Tutorials* **20**(2), 1492–1525.

[66] Nordmark, E. and Gilligan, R. [2005*a*], Basic Transition Mechanisms for IPv6 Hosts and Routers, RFC 4213, IETF.

[67] Nordmark, E. and Gilligan, R. [2005*b*], Basic Transition Mechanisms for IPv6 Hosts and Routers, RFC 4213, IETF.

[68] Nweke, L. O. and Wolthusen, S. D. [2020], 'A Review of Asset-Centric Threat Modelling Approaches', *International Journal of Advanced Computer Science and Applications* **11**(2).

[69] of Homeland Security, U. D. [2007], 'Common Attack Pattern Enumeration and Classification (CAPEC™)'.
URL: https://capec.mitre.org/

[70] OISF [2020], 'Open Source Network Analysis and Threat Detection Software'.
URL: https://suricata.io/

[71] Ometov, A., Bezzateev, S., Mäkitalo, N., Andreev, S., Mikkonen, T. and Koucheryavy, Y. [2018], 'Multi-Factor Authentication: A Survey', *Cryptography* **2**(1).

[72] *OpenWrt Software* [2018], http://openwrt.org.

[73] Palet Martinez, J., Liu, H.-H. and Kawashima, M. [2019], Requirements for IPv6 Customer Edge Routers to Support IPv4-as-a-Service, RFC 8585, IETF.

[74] Park, W. and Ahn, S. [2017], 'Performance Comparison and Detection Analysis in Snort and Suricata Environment', *Wireless Personal Communications* **94**(2), 241–252.

[75] Postel, J. [1980], User Datagram Protocol, RFC 768, IETF.

[76] Qian, Z. and Mao, Z. M. [2012], Off-path TCP Sequence Number Inference Attack - How Firewall Middleboxes Reduce Security, *in* 'IEEE Symposium on Security and Privacy', pp. 347–361.

[77] Répás, S., Hajas, T. and Lencse, G. [2015], Application compatibility of the NAT64 IPv6 transition technology, *in* '38th International Conference on Telecommunications and Signal Processing (TSP)', pp. 1–7.

[78] Shostack, A. [2014], *Threat Modeling: Designing for Security*, John Wiley & Sons.

[79] Sourcefire, I. [2021], 'Snort 3, Open Source Intrusion Detection and Prevention System'.
URL: https://snort.org/

[80] Troan, O., Dec, W., Li, X., Bao, C., Matsushima, S., Murakami, T. and Taylor, T. [2015], Mapping of Address and Port with Encapsulation (MAP-E), RFC 7597, IETF.

[81] van Hauser Heuse, M. [2020], 'thc-ipv6'.
URL: https://github.com/vanhauser-thc/thc-ipv6