ASTES

Advances in Science, Technology and Engineering Systems Journal
Vol. 6, No. 2, 722-732 (2021)
www.astesj.com

ASTES Journal
ISSN: 2415-6698

Securing IPv6 Neighbor Discovery using Pre-Shared Key

Rezaur Rahman*, Hossen Asiful Mustafa

Institute of Information and Communication Technology (IICT), Bangladesh University of Engineering and Technology (BUET), Palashi, Dhaka, 1205, Bangladesh

ARTICLE INFO

Article history:
Received: 29 November, 2020
Accepted: 03 March, 2021
Online: 27 March, 2021

Keywords:
Internet Protocol Version 6
Neighbor Discovery Protocol
Neighbor Cache Entry poison

ABSTRACT

Neighbor Discovery Protocol (NDP) is used to discover the MAC address of the connected hosts in Internet Protocol Version 6 (IPv6) in a networked environment. Neighbor Cache Entry (NCE) table holds the association between a host's IP address and MAC address. However, according to the protocol, the MAC address could be overwritten by sending a single fake packet to its victim. This is a serious security loophole as traffic can easily be sniffed by the attacker. In this paper, we present a scheme to address this problem. Our proposal suggests that when Neighbor Solicitation (NS) and Neighbor Advertisement (NA) process completes, a randomly generated key can be exchanged between them so that, in case of an attack, that key can be used to verify the request. We implemented the proposed scheme in NS3 and simulation results show that our proposed scheme can perform effectively while circumventing the attack that uses override flag of IPv6.

1 Introduction

Internet Protocol Version 6 (IPv6) is the latest technology in addressing networked devices. Similar to Internet Protocol Version 4 (IPv4), IPv6 is an addressing mechanism that provides network identification to a device at the network layer of Open Systems Interconnection (OSI). Its predecessor IPv4 had an address space limitation which has been mitigated using the technologies like Network Address Translation (NAT) [1]. But, such measures were only temporary and could not provide a permanent solution. Which is why IPv6 was developed with a large address space with 128-bit address, compared to 32-bit address space of IPv4. Additionally, some methods were used to reduce the complexity of the addressing scheme. Many other advantages from IPv4 was also baked inside IPv6 for better adaptation such as reduce dependency on NAT, and gain connectivity using automatic address allocation [2].

However, some of the common security vulnerabilities are observed in the IPv6 and attack like man-in-the-middle is still possible [3], [4]. Like existing TCP/IP stack, to successfully send a packet from one device to another, data link layer address, also known as Media Access Control (MAC) Address, is required in IPv6. As pre-populating the MAC address table for each of the connected devices is not practical, IPv4 used a mechanism called Address Resolution Protocol (ARP) [5]. The purpose of ARP is to send out a broadcast asking for destination device to reply with its own MAC address, given that the assigned IP on the destination device matches with the IP address in the broadcasted packet. In an ideal scenario, only the device with matched IPv4 holder will reply with its own MAC address so that link layer communication can be established. Once MAC address and IP address are resolved, the source device adds the MAC address in ARP cache for future use as both of them will possibly have to communicate between themselves for a certain period. ARP cache reduces repeated ARP resolution calls.

Unfortunately, in IPv4, the ARP cache in a device can easily be poisoned; an attacker can easily send fake ARP messages and those fake requests will be accepted and processed by the device as per the protocol. This creates a significant security risk as all the traffic can be diverted to the attacker host [6]. Such mechanism paves the way for sophisticated attacks including stealing username and password, obtaining users session ID, redirecting users to fake sites, etc. The attack becomes extremely effective if the client and server communication is not protected by any encryption mechanism [7]. Even a novice attackers can use tools like dsniff [8], cane and able [9], ettercap [10], etc. to launch such attacks.

Although IPv6 does not use ARP to resolve MAC address, it employs a similar protocol called Neighbor Discovery Protocol (NDP) [11]. Using NDP, nodes in the same link advertise their existence as well as learn about the other nodes. Similar to ARP cache, NDP maintains Neighbor Cache Entry (NCE) table to reduce repeated NDP resolution. In NDP, Neighbor Advertisement (NA) and Neighbor Solicitation (NS) packets are used to resolve MAC addresses of the connected hosts. Although unsolicited NS messages are ignored in IPv6, MAC spoofing attack is still possible [12]. The

*Corresponding Author: Rezaur Rahman, Institute of Information and Communication Technology (IICT), Bangladesh University of Engineering and Technology (BUET), Palashi, Dhaka 1205, Bangladesh, Tel # +8801755543315, Email: rezaur@protonmail.com

author showed that the *override* flag in the NS packet could be used to distort the NCE table and perform hijacking of traffic [13]. The primary objective of *override* flag is to announce the changes in the data link layer. For example, hot-swappable Network Interface Cards (NIC), which are usually used by high end routers or switches, can be replaced without powering down the device; in such case, the NIC would send out a NS with *override* flag set to announce its new MAC address [14].

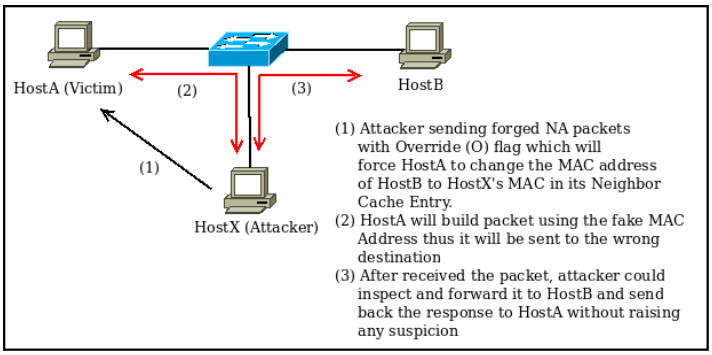

Figure 1: How an attack can be initiated in IPv6

Unfortunately, this override mechanism in IPv6 can be easily exploited by an attacker unless specific countermeasures are taken. The overall attack mechanism is illustrated in Figure 1 where *HostX* is the attacker and *HostA* is the victim. Here, we assume that *HostA* already has MAC address of *HostB* in its NCE table. At first, the attacker *HostX* sends a forged NA packet with *override* (O) flag set, claiming to be *HostB*; this will force *HostA* to change MAC address for *HostB* in its NCE table with the MAC address of *HostX*. Subsequently, all packets for *HostB* from *HostA* will be delivered to *HostX*. Thus, *HostX* can intercept, and inspect the packets and can also act as middle-man between *HostA* and *HostB*. Each and every node in local area network is in risk of such attack irrespective of the number of the nodes in the network. For prevention, we could use mechanisms like SEND or IPSec but they have their own limitations like complex configuration, overhead and difficulty of use.

In this paper, we propose a scheme where the a randomly generated key will be transported when two hosts initiate communication between themselves. This key will provide a layer of protection against unsolicited *override* requests by providing a challenge to the sender. As this protection mechanism will be baked inside the protocol, it will be seamless to the end users as well as to the network administrators; thus it should obtain wide adaptation. We implemented the proposed scheme in NS3 and experimental results show that our proposed scheme can effectively handle this attack without introducing any major performance impact.

The rest of the paper is organized as follows. In Section 2, we will look into the ICMPv6 and will discuss NDP and try to illustrate the exact problem we are trying to address. Then, in Section 3, we will explore current solutions available and what is preventing them from adaptation. We will then discuss proposed solution in Section 4 with some details. In Section 5, we will discuss simulation of the current implementation and show how an attack can be lunched against a host. Section 6 will demonstrate and prove the proposed scheme using simulation. Approximate requirement for storing key will be shown followed by performance comparison in Section 7; memory requirement and shortcomings will be discussed in Section 8. We conclude the paper in Section 9.

2 Background

Previously, ARP was managed as a simple messaging and was not a part of any other protocol[5]. However, in the IPv6, this mechanism of resolving from IP address to MAC address and other functionalities have been confined within Internet Control Message Protocol version 6 (ICMPv6) [15]. The structure of ICMPv6 packet has been given in Figure 2.

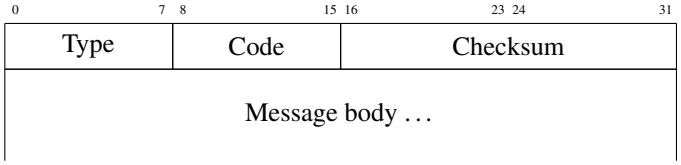

Figure 2: ICMPv6 packet details

In the following, we will give an overview of neighbor discovery protocol protocols to provide a better understanding of the proposed scheme.

2.1 *Neighbor Discovery Protocol (NDP)*

Neighbor Discovery Protocol (NDP) is used to send and receive message from other connected hosts. There are five different ICMPv6 message types and some conceptual demonstration are provided below:

2.1.1 *Router Solicitation (Type 133)*

When a node becomes active, in most of the cases, it needs to locate the gateway on that network because it will be able to communicate with the internet by reaching the gateway. This type of message is being used to request the routers to respond with their gateway address so that the device can communicate with necessary servers immediately.

2.1.2 *Router Advertisement (Type 134)*

In many cases, finding out the gateway for the node is not considered as high priority. Because reaching other networks is not required immediately and the node can wait for a certain period. The router will send out periodical updates to all the hosts in the network with the updated gateway address. This system makes maintaining a complex network easy because in the production environment, changing the gateway can cause downtime. By using this method, such downtime can be reduced and the nodes will be able to maintain connectivity with the gateway.

2.1.3 *Neighbor Solicitation (Type 135)*

To determine the MAC address, this type of packet is used by the hosts. NS packets are responsible for requesting MAC address from a specific target. The detailed description of the packet is given in Figure 3. This packet is sent out to all the hosts in that broadcast

domain; thus, all the hosts will receive this packet when transmitted by a host.

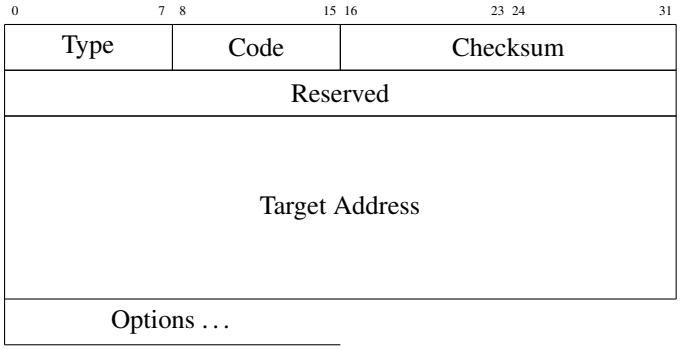

Figure 3: NS packet description

This packet is quite simple and it does not have any specific bits or flags to convey additional information. The value of type field will be 135 as per specification and the target address will contain the IP address of that device with which the initiator wants to communicate with.

2.1.4 Neighbor Advertisement (Type 136)

In response to a received NS packets, NA packets are used. The NA packet structure is shown in Figure 4

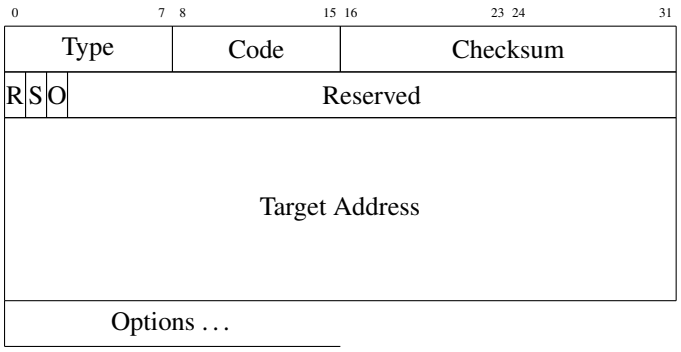

Figure 4: NA packet description

This packet is comparatively complex than the NS packet and have three fields which we will discuss below:

- **Router flag (R)**: Router flag is used to indicate that the device sending this packet is a router.
- **Solicited flag (S)**: When the packet is sent in response to a NA, this flag is set.
- **Override flag (O)**: Override flag is used to overwrite cache entry in the NCE table and update the MAC address.

NA packets can be solicited as well as unsolicited, meaning, a host can receive and process an NA packet even if it did not send out any NS request. Moreover, NA packet has an *override* flag which is sent out if any host has changed its MAC address. This is substantially effective and fast considering how easily a device can announce its new MAC address when it is changed. However, such packets can easily be forged and poison one or more hosts. We are going to focus on this field as exploiting this flag can cause man-in-the-middle attack.

2.1.5 Redirect (Type 137)

Redirect type is used only by routers because in many cases better route to a network has to be advertised to the hosts and by using this type of packet, such objective can be achieved.

3 Related Works

To address the attack for NDP in IPv6, protective measures are required to make the NDP messages secure. SEcure Neighbor Discovery (SEND) was proposed with protection for NDP. It uses Cryptographically Generated Address (CGA) to provide security to the NDP packets. However, lack of solid guideline and procedures make this protocol difficult to implement. Other obstructions include a valid Certification Path and performance impact [16]. There are also lack of tools which can be used in production environment making this system almost impossible to implement [17]. Moreover, it is also known to be resource hungry as well. If such protective measures are to be implemented, it will certainly require advanced knowledge as well as complicated configurations in all of the networked devices in the environment. Such implementations are difficult to adopt and thus, acceptance is low which ultimately leads towards insecure network.

IPSec was proposed to secure the NDP from the possible problems which are caused by the unaddressed complications of the protocol. Unfortunately, the problem with IPSec is multifold. Firstly, the concept of IPSec requires a time to grasp because of its complexities. For example, there are many ways by which it can be configured and being a security feature, any misconfiguration will result in the interoperability. This complex configurations can lead towards frustration and avoid this protocol for something less secure. Moreover, in IPSec, shared key based authentication can be configured. If the administrator wants to distribute the keys manually, then he or she has to perform a complex task of maintaining keys for every host. If the automatic method is preferred over shared key, it would introduce further complex configuration in the host devices. On top of this, not all devices support same IPSec configuration; thus each type of operating system requires its own separate configuration. It appears that IPSec creates more problem then it promises to solve [18], [19].

In [20], the author proposed a scheme that uses Dynamic Host Configuration Protocol Version 6 (DHCPv6); it suggests that a DHCPv6 server should be used to provide IP address to the end hosts based on Stateful Address Autoconfiguration (SAA). But, there are many situations where DHCPv6 server is not practical. Small or medium offices, where technical expert might not be available; thus configuring and managing a DHCP server might seem complex for them. Additionally, it may become a single point of failure for the whole network.

To reduce the difficulties suggested by these mechanisms, certificate authority based system is intentionally avoided so that our proposed scheme can be widely adopted and incorporated. As PSK does not depend on any other hardware, configuration or incur any

additional cost, it is expected that it will be used by many and provide a more secure networking environment.

4 Proposed Pre-Shared Key (PSK) Based Scheme

In this section, we present our proposed PSK based scheme. In our scheme, each pair of hosts in the network will have its own secret key. The proposed length of this key is 16 bytes (128 bits) long. This PSK is communicated to other hosts using a new packet type. In the following, we discuss the details of the scheme.

4.1 New Packet Header and Description

Our proposed header is compliant to RFC 4861 and is shown in Figure 5. Now, we will briefly discuss the header.

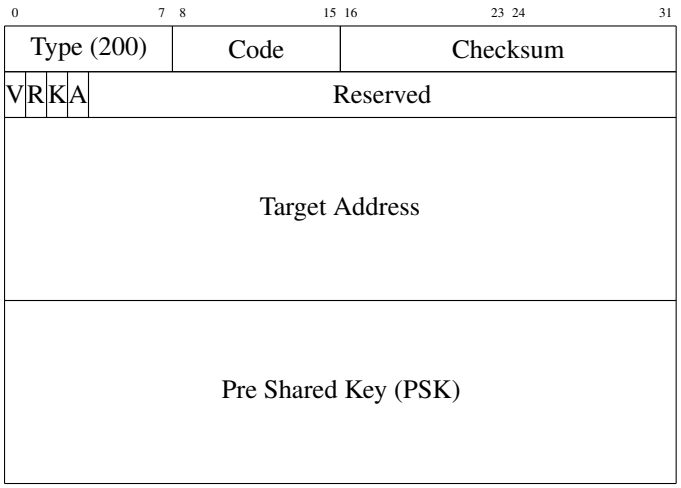

Figure 5: Proposed PSK header

- **Type**: This header is responsible for describing what kind of packet it is and how the node will process it. We propose to use the next available sequence; however, in this paper, we will be using Private Experimentation type which is denoted by 200.

- **Verification (V) flag**: This flag will indicate that the packet is to be sent out using multicast so that all the hosts in the network receive it. This packet is an instruction that the IPv6 address holder should return its key using unicast to the initiator. This flag will be mostly used in situation where there is a conflict or the node which requested the *override* does not have any associated key to present to the recipient; thus the scheme will force the receiver to check the network whether such association is still present in the network.

- **Response (R) flag**: The purpose of R flag is to perform key verification. It is being used to respond to a verification multicast packet which has been received by the node. This flag will be used when responding to the verification packet flag.

- **Key (K) Flag**: When the host is responding to a PSK NS, this flag will be set so that the initiator can understand that the packet contains the key for a particular request.

- **Acknowledge (A) flag**: After the key has been sent to the destination host, by using the K flag, it will reply using this flag enabled so that the initiator can understand that a key has already been sent and subsequent response has been received.

- **Target Address**: This field will hold the target IP address of the host.

- **Pre Shared Key (PSK)**: This field is 16 bytes, i.e, 128 bits and will be used to share the key between two hosts.

4.2 Key Cache Entry (KCE)

This is a table which will be stored in the slower part of the operating system's memory and will hold all the exchanged keys between hosts in a network. The reason for suggesting to store them in the slower part is because this key is not required in per packet operation. This set of data only required to be accessed when there is an *override* request and when new entries are added or removed from this list.

4.3 Key Generation

The key generation process will be done by the operating system during the initial communication phase. The key shall consist of capital letter, small letter, numerical as well as symbols to ensure it cannot be brute-forced in any practical manner. As the networking systems improve, the chances of brute forcing a key also increases. To ensure that this scheme is protected against such attack, 128 bit long key has been chosen. As previously stated, all of the ASCII characters will be used, thus even with a million attempts per second; it will take over billion years to brute force this key thus making it practically impossible to crack [21].

4.4 Key Sharing Phase

When two hosts, e.g., *HostA* and *HostB*, in the network want to communicate, they must have each others MAC address. Below, we discuss the full process for NS, corresponding NA and our proposed shared key mechanism, i.e., *Key Sharing Phase*. The process in shown in Figure 6.

1. By sending an NS message, *HostA* initiates the connection process.
2. *HostB* responds with NA packet containing the link layer address of its own.
3. Both the devices now have MAC address of each other.
4. Now, *HostA* starts the PSK process by generating a key and crafting a packet with type 200 (experimental) with the Key (K) flag set.
5. *HostB* receives packet and saves the key in its memory and responds by returning a key-acknowledge (A) packet back to the *HostA*.

6. *HostA* receives the packet and saves the key sent previously into it's own NCE table.

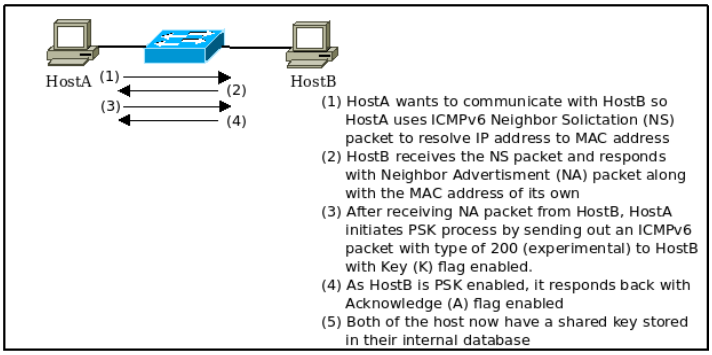

Figure 6: The process of shared key exchange

The full process on how the attack prevention system will work is depicted in Figure 7.

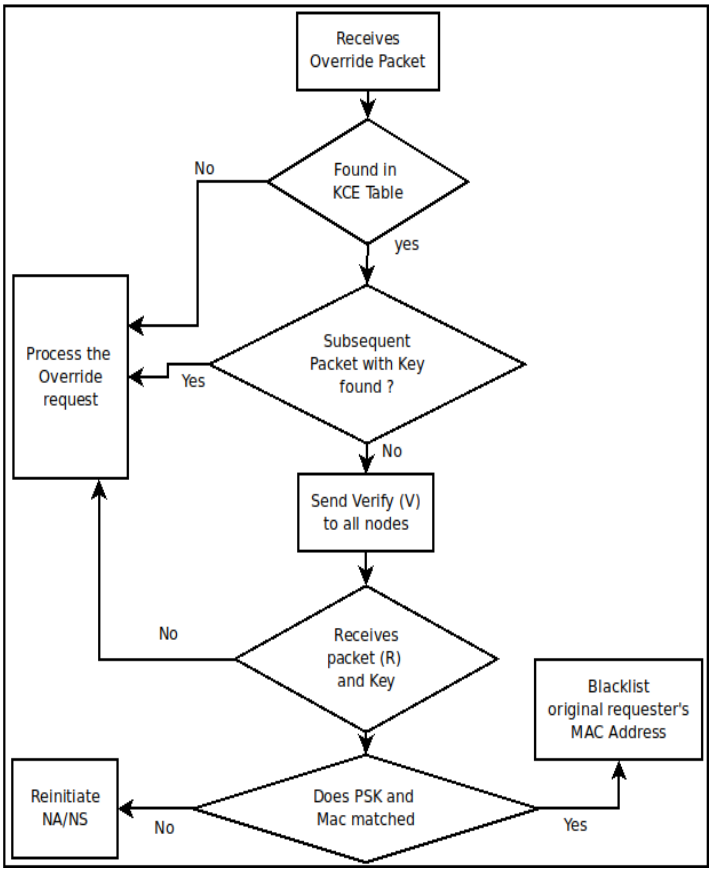

Figure 7: Overview on how the prevention mechanism works.

Now, we will discuss two scenarios where one of these devices are not PSK compliant or does not have our proposed feature enabled.

4.4.1 *Source does not have PSK but destination have it enabled*

As described in Figure 6, after step number two, the process will not continue and the source which is denoted by *HostA* will not send an PSK packet. As no instruction is being sent by the initiator, the destination will also comply and avoid sending any key-acknowledge packet. The communication between these two hosts will continue as usual.

4.4.2 *Source have PSK enabled but the destination does not*

In this situation, the source will send a packet with the Key (K) flag enabled but as per the specification of ICMPv6, that packet will not be recognized by the destination and will be dropped. Because no subsequent packet will be received by the source, it will decide that PSK mechanism is not enabled and will safely remove any generated key allocated for destination. Thus, communication between these two host will remain uninterrupted.

4.5 *Affects of PSK in Normal Scenario*

When a legitimate host, e.g., *HostB*, needs to update its own MAC address, and inform other connected nodes, it will send a separate NA packet to each host with *override* flag set and its own PSK from the KCE table for the target host using Key flag set. Upon receiving the key exchange packet, the host, e.g., *HostA*, will match the PSK in its KCE table and update the corresponding KCE. Thus, a legitimate host will be able to update its MAC address without any security risk.

4.6 *Affects of PSK in Attack Scenario*

We illustrate the attack scenario in Figure 8. Here, *HostA*, and *HostB* are legitimate hosts and *HostX* is the attacker. To initiate an attack, *HostX* sends out an NA packet with the *override* flag set announcing his own MAC address as the new MAC address to reach *HostB*. Since the hosts are using PSK mechanism, the victim host, *HostA*, will request for the PSK from *HostX*. As the attacker does not have the PSK of *HostB* for *HostA*, it may respond with a random PSK or may not reply at all. In case of random PSK, the probability that the PSK matches is $\frac{1}{2^{128}}$ since the PSK is 128-bit long; so, the attack is highly unlikely to be successful. When *HostA* does not receive a valid response, it will multicast, also known as broadcast, a verification packet which will contain the original MAC address. In reply, *HostB* will reply with its PSK using unicast raising the *V* flag and by sharing its key; thus, the authenticity of the *HostB* will be confirmed. So, the attack will fail and the override packet sent by the attacker will be ignored.

The verification packet will be dropped by other hosts in the network as the target address will not match. Only the holder of the target address will respond with Response (R) flag set.

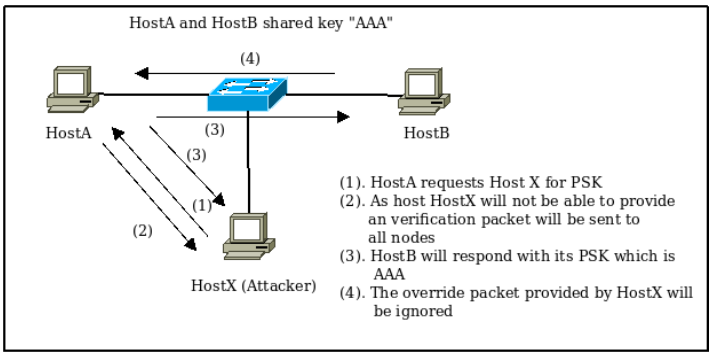

Figure 8: Overview on how the prevention mechanism works.

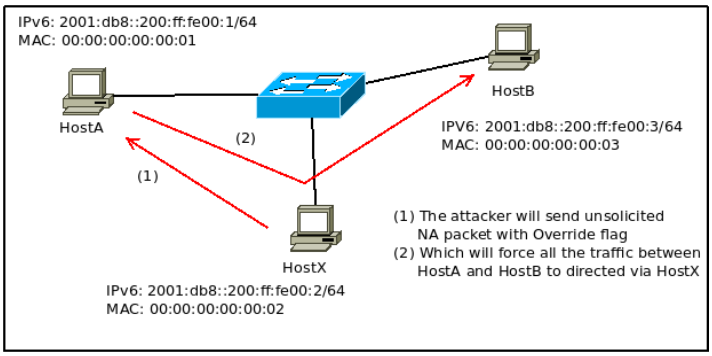

Figure 9: Overview of the network

4.7 Handling Different Scenarios

There might be a situation where a host has lost the PSK key among a pair. If this pair tries to initiate connection, there could be a case where one host will have a key but other host will have no knowledge of previous communication, e.g., sudden power loss. In such case, they will have to re-agree upon a new key as one of the host will have to re-initiate NA and process subsequent NS. After which PSK processes will be initiated and eventually both of the hosts will have a shared key like before.

Now, let us consider an extreme case where the MAC address of a host, HostS, has been hot-swapped. In such case, that device should store the existing PSK(s). After the address has been changed, the device will send out a multicast using the *override* flag. At the same time, an attacker, *HostX*, who was waiting for such an event, may send out another fake packet with *override* flag. As per proposal, all the nodes in the network will check PSK and only HostS which will be successful as the PSK will be available to HostS; thus, the fake messages will not be processed as it will be canceled out by the verification mechanism.

One of the primary focus of this work is to ensure that a non-PSK and a PSK-aware system can coexist. All the keys are being transferred using the proposed packets for a minimum impact. Only a verification packet has been proposed, which will be ignored if received by a non PSK enabled system.

5 Simulation without PSK

We have implemented the proposed scheme and showed how it can prevent an attack in Network Simulator (NS3) [22]. NS3 is a discreet event simulator capable of producing results by simulating the source code of IPv6 protocol. In our simulation, we modified the ICMPv6 layer code of NS3 in order to achieve result.

In this simulation, we are trying to show the scenario that has been graphically described in Figure 9. In our simulation, the IP address and MAC address described in the figure will be used. In the simulation, we will first test the original attack scenario and then, gradually move into modifying the logics of ICMPv6. We will also analyze the part of the code which is responsible for updating the MAC address when a *override* packet arrives and then, we will show two situation where the key is matched against the cache of the host and in another, there is a mismatch.

5.1 Attacker Host

In this section, we discuss an attacker host which we developed for simulation. This host will be responsible for sending out fake packet with malicious intent. This host has been denoted as *HostX* in Figure 9

In the code, a separate AttackApp has been used which inherits from Application class in NS3. This application has two primary features which are discussed below:

- The application will have high packet sending rate as the objective of the attacker app is to flood *HostA* with fake packets. Although such flooding of packets is not necessary, but it is better to keep on sending them so that even if *HostA* recovers from the attack for any reason, the attacker will keep on overriding the actual entry with a fake one.

- As the attacker will send forged NA packets, it will utilize the SendNA method found inside the ICMPv6 implementation in NS3. Using this method, we will able to send fake NA packets.

The portion of code which is responsible for sending out fake NA packets is given below:

```
void AttackApp::SendPacket(void)
{
  m_attacker->SendNA(m_fakeAddr, m_vAddr↩
    , &m_vMac, 1);
  ScheduleTx();
}
```

5.2 Traffic Generator

We have selected ping, also known as Echo, as the traffic generator as it is simple to trace, and implement and it is understood by many. We will run the simulation for ten (10) seconds; during that time, the ping application will send multiple ping requests from *HostA* to *HostB* and if the packet is received by *HostB*, it will send back a response. It should also be noted that if the attack is successful, *HostA* will send packets with *HostX*'s MAC address and that ping request will not receive any response because the attacker host is not configured with ping application.

5.3 Analysis

Under normal situation, we will run our initial simulation where it is expected that the override packet sent by the *HostX* will cause diversion of traffic as discussed.

After running the simulation, NS3 generated network logs in PCAP format. This file can be opened using Wireshark application so that we can analyze the communications between hosts [23]. Now, we will analyze traffic from *HostA*.

As we can see from Figure 10, the packet number 12th is a successful ping response received by *HostA*. We can also see that the source and destination MAC addresses are correct and traffic generated from *HostA* was sent to correct receiver, which is *HostB*, and the expected response was also received from *HostB* to *HostA*.

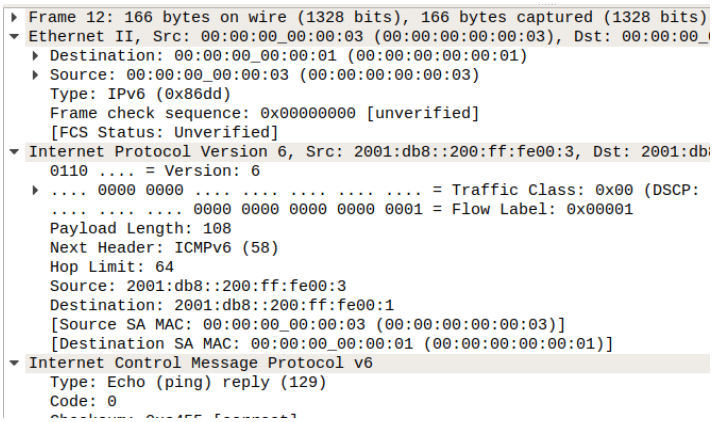

Figure 10: Diversion of traffic

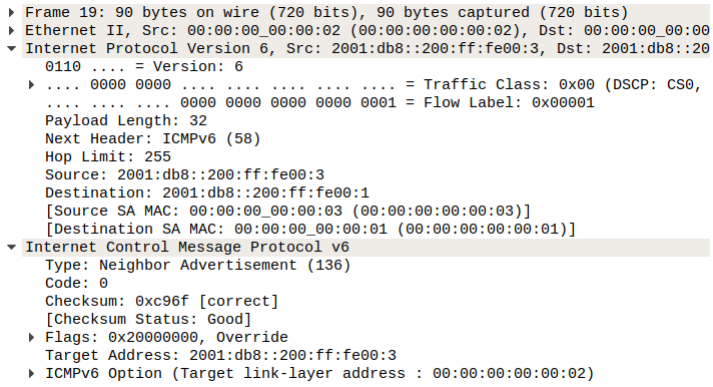

Figure 11: Fake NA packet sent by *HostX*

However, as the AttakApp starts to send fake requests, after some time, the simulator generated many packets and among those, details of one of them can be seen in Figure 11. The 19th packet shows us that the attacker is now active and sent ICMPv6 packet with *override* flag set and requesting to re-write the MAC address to 00:00:00:00:00:02 for the IP address of 2001:db8:ff:fe00:3.

As we can see, immediately after processing the fake NA packet, in the packet number 20th, the *HostA* sends the ping request to the wrong MAC address which is 00:00:00:00:00:02 as per Figure 12. As discussed above, this is the expected result and all the traffic will now be sent to the attacker instead of the correct recipient.

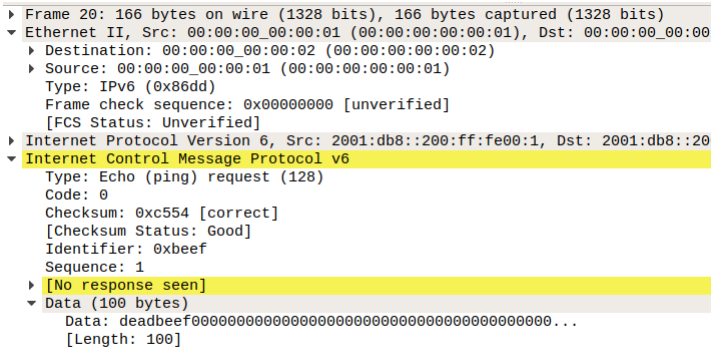

Figure 12: Traffic sent to the wrong MAC address

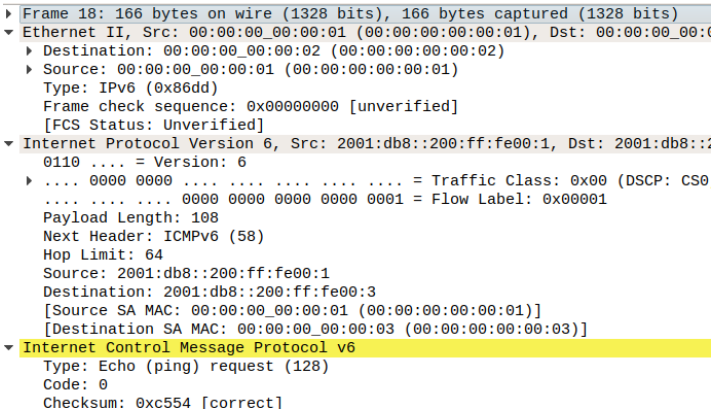

Figure 13: Echo request wrongly received by *HostX*

If we check what traffic was received by the attacker will further solidify our assumption. If we open the traffic generated by NS3 using Wireshark for the attacker or *HostX*, we will see that ping requests has been received by the attacker machine as we can see in Figure 13.

6 PSK Simulation

6.1 Modification of IPv6 Stack

In NS3, there are several files which are being used to simulate the IPv6 networking protocol. In this paper, relevant source code files will be discussed and a brief description will be provided. Below, we will discuss some critical files which are required for ICMPv6 to operate properly.

- icmpv6-l4-protocol.h: This is a C++ header file and it contains the basic structure of this protocol and other definitions which help the programmer to understand the basic structure and use them when necessary. All the code initialization and declarations for ICMPv6 is declared here and name of the functions are also written down in this file.

  This file defines Icmpv6L4Protocol class which has a base class of IpL4Protocol.

- icmpv6-l4-protocol.cc: This file contains all the implementation for ICMPv6. All the definitions declared in the header

file are coded in this file and all the logic is implemented. In this file, we will locate the possible problem and will try to solve this issue using PSK mechanism.

Below, we will discuss some basic modifications which are required:

6.1.1 KCE Initialization

As already discussed, we will require a list to store MAC addresses and its associated keys. Following line was included in the header so that we can achieve our goal.

```
std::map<std::string, int> k_token;
```

Here, map is a standard C++ declaration where a key value pair is declared. The string will store the MAC address for the remote host and the int variable is programmed to store the corresponding key value for that host. Here, to make our proposal more reader-friendly, we have used integer variable so the concept remains clear without altering our ultimate goal.

This segment of the code will be executed when the ICMPv6 layer of IPv6 initializes.

6.1.2 Kay-cache Value Initialization

For simplicity we will insert values into the key cache table and will eventually check against this list when the host receives an *override* request from another host.

In the CreateCache method of the initialization for ICMPv6, we have added against two hosts:

```
k_token.insert(std::make_pair↩
    ("00:00:00:00:00:01",1122));
k_token.insert(std::make_pair↩
    ("00:00:00:00:00:03",1122));
```

As we can see, these are the MAC addresses for *HostA* and *HostB* and corresponding shared keys. The *HostX* will be the attacker which will flood *HostA* with fake NA packets having their *override* flag set and will try to divert the traffic from *HostA* to *HostX* rather then to *HostB*.

6.1.3 Vulnerable Code

In NS3, upon receiving *override* flag, the change in the code is done by the line:

```
entry->SetMacAddress (lla.GetAddress ())↩
    ;
```

Here, the *entry* is the variable which holds the a specific entry from the Neighbor Discovery Cache and this line of code is used to update the value of the MAC address and eventually propagate the changes into the KCE table.

We will eventually change this part of the code where it will only execute when key matches from the key-cache table.

6.2 Key Mismatches Situation

In this section, we will discuss the scenario where an attacker is trying to send fake NA packets to *override* the MAC address for a particular IP address. Now. we will have to extract the values from the incoming *override* packet and they are being done by modifying the a section of the logic and extracting information from the packet and the code for which is given below:

```
io<<entry->GetMacAddress();
std::string mac = io.str().substr(6,23);
```

After we gather information from the received packet, we now look into the key-cache table.

```
bool isFound = false;
bool isTokenMatched = false;

std::map<std::string, int>::iterator it ↩
    = k_token.begin();
while(it != k_token.end())
{
  if(mac.compare(it->first) == 0)
  {/// a match found
  isFound = true;
    if(it->second == receivedToken )
    { isTokenMatched = true;  }
  }
it++;
}
```

As shown in the code above, we iterate over all the entries inside the list and look for a matching MAC address. And if such MAC address is found, we will then again check it against the *isTokenMatched* variable which is in this scenario empty. As the attacker has no knowledge of the key which has been exchanged.

And finally if both the *isTokenMatched* and *isFound* is found, KCE is updated using the following code.

```
if(isTokenMatched == true && isFound == ↩
    true)
{
  entry->SetMacAddress (lla.GetAddress ↩
    ());
}
```

As we can see, this code is now protected with a key and it cannot be updated by any host without knowing the key first.

If we check captured files generated by the NS3 system, we will a see that like in the unmodified version of the code, same NA packets are being sent to *HostA* (Figure 14). In this packet, we can see that it has the *override* flag set and it is being sent from *HostX* to *HostA* in order to perform man-in-the-middle attacks. We also see that even though an *override* packet was sent to *HostA* requesting it to update the MAC address, the request was not honored. In the 69^{th} packet (Figure 15), the response to the request was received successfully from *HostB*; thus making it impossible for the attacker to alter the KCE table and perform any man-in-the-middle attack.

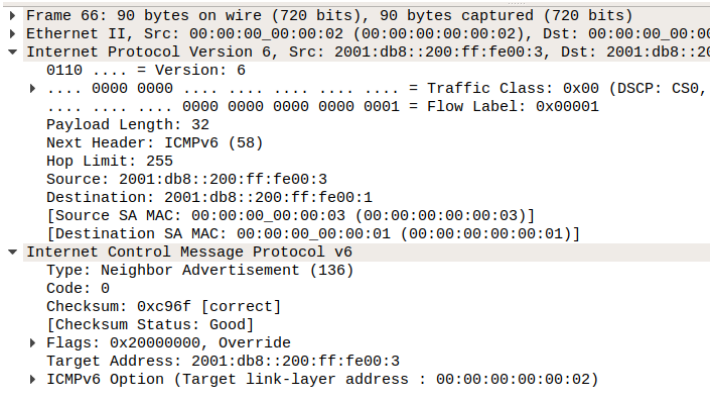

Figure 14: Fake NA packets sent again without matching key

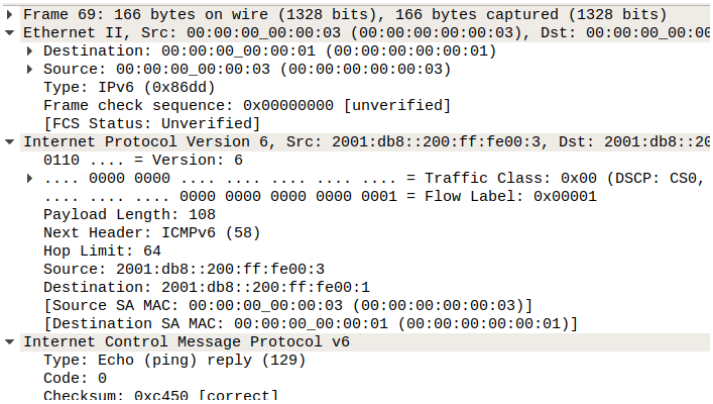

Figure 15: Without same key, communication remains unaffected

6.3 *Key Matching Scenario*

Now, we will discuss the scenario where *HostX* is the legitimate owner of a IP Address and the MAC address behind that IP Address has changed; so, it is required to propagate this change to another host.

To do this, we will set the value to the equal to the value found in the key-cache table. All other aspects of this code will remain same.

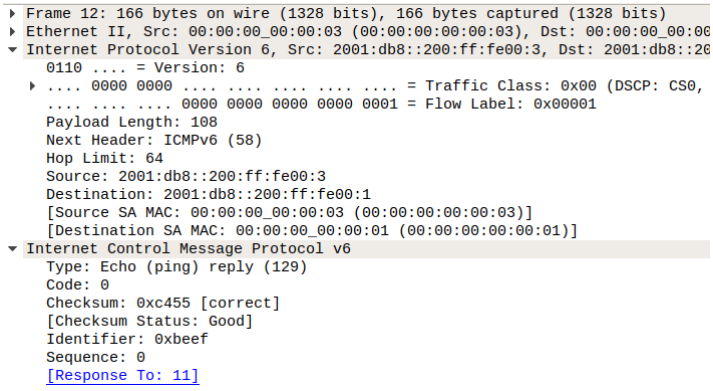

Figure 16: NA packets sent having matching key

If we observe Figure 16, we will see that the 12th packet sent was an response to the 11th packet and the reply was successful.

Further down the timeline of this simulation, we can observe that the ping response to the request has not been received. If we investigate why no such response were received, we will see that the request has been sent to the wrong MAC address which is in this situation the host which have the correct key to alter the KCE table of *HostA* (Figure 17).

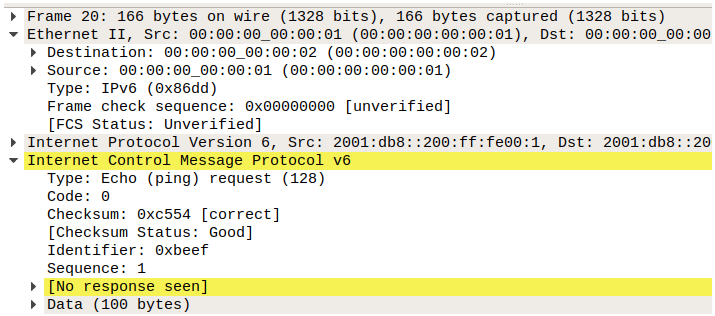

Figure 17: Successfully override request as keys matched

7 Comparison

As already stated, to counter this type of attack, SEND protocol is suggested. Unfortunately, as this protocol uses CGA, such generation and verification have resource overheads. Even though various algorithms have been proposed to improve the response time and General-Purpose computing on Graphical Processing Units (GPGPU) has been suggested to improve generation time, the overhead is significant for devices with low computing resource or limited power source as every packet must follow some cryptographic process. It has been widely accepted that introducing cryptographical elements in each packet will undoubtedly introduce some delay. Such delay may not become apparent to high end computing devices, but devices with fewer resources will have to face performance issues [24].

IPSec also has some adverse impact on the throughput of the associated devices. As various cryptography tasks have be be performed, it will incur various complex calculations for not only data but also for the header of the packet. The impact of such complex calculations does not seem to affect high end computational hardware but the impact on the devices with lower specifications is obvious. To compensate lack of resource, the traffic rate will have to be sacrificed [25].

Compared to above methods, our proposed scheme has no processing per packet basis which makes it faster and “CPU friendly” as it will not introduce any additional burden on its computing unit.

7.1 *Qualitative Analysis*

In this section, we will discuss some aspects of PSK based system and non-PSK based system in order to understand more about the benefit of our proposed model.

- Our suggested method have performance benefit as there is no per-packet based processing. For example, in case of both

IPSec and SEND, some additional calculations are required in order to either validate the content of the packet or to perform cryptographic transformations. However, our proposal has purposefully avoided any cryptographical calculations so that performance related impact is avoided.

- Simplicity is also another feature of our proposed mechanism. This protocol should be easy to implement and was designed in such a way that both end users and administrators face no difficulties while implementing it.

- This mechanism is also applicable for the public network like the free WiFi we find in the public areas. As other suggested protocols require some sort of authentication in order to connect to the network, it becomes almost impossible for such protocols to be implemented in a public networks. However, as PSK based model does not require any authentication with any one else, a new node can easily join the network.

- Even though at initial stage, installation based modules might be necessary in order to use PSK based model but originally it is designed primarily considering that the proposed mechanism will be baked inside IPv6 protocol. Thus, the proposed scheme is independent from other system. As of now, all other suggested systems require separate installation and configurations for them to work.

- Undoubtedly, cost is a serious concern when deploying security mechanism. Other available systems will incur cost in either certificate purchasing or developing infrastructure whereas our proposed system requires none.

7.2 Quantitative Analysis

In this section, we will compare packet size to show that the proposed scheme requires either less handshaking packet or reduces size for every packet.

7.2.1 Comparison With IPSec

IPSec has several phases which need to be established prior to any communication can take place. It uses two encrypted tunnels between nodes and each tunnel establishment requires additional communication. The first tunnel requires six packets and second tunnel requires three packets in order to establish initial communication. However, in case of our proposed mechanism, only two packets are required.

Additionally, the size of initial handshake packets are much larger in IPSec which requires approximately 1000 bits of packet size. The packet size for our proposed scheme is only 40 bits.

7.2.2 Comparison With SEND

SEND can be configured with minimum configuration for initial handshake. If trust anchor is used which is the most practical solution, it requires additional three packets in order to verify that the certificate chain is valid. On the other hand, our proposed scheme requires only two packets.

Additionally, in case of SEND, we see a significant increase in every packet size where a simple ping can jump up to approximately 450 bytes as SEND need to add various segments like Timestamp, Nonce and most importantly RSA Signature. As our proposed mechanism does not include any per packet based data, the size will not change.

8 Discussion

8.1 Memory Requirement

Regarding storage requirement of storing the keys, the storage for link layer address is 6 byte (48 bit) and the proposed key is 16 byte (128 bit). Thus the overall storage required for this method will be 22 bytes.

It should also be noted that because the PSK key will not be required for every packet transmission, keys does not have to be directly stored in active memory segment. They can be pushed to swap area of the memory if it is available. When it is required, the data can be fetched from the slower area for further processing.

8.2 Shortcomings

We noted that the proposed scheme is designed for an environment where all of the Layer 2 networking components are switches. As switches operate in Layer 2 of OSI layer, it is aware that in which port a specific host is located; thus traffic from one host to another is considered safe. However, if we consider a networking environment where some of the devices are hubs, this proposed PSK mechanism will not work as all of the packets will be sent to all of the nodes connected to the hub. As this mechanism works by transmitting a key between two nodes, using hubs will certainly have adverse consequence as the key will not be able to move from one node to another without leaking it to other parties.

However, nowadays devices like hubs are now non-existent in networking environment as prices for devices like switches have reduced drastically. Hubs does not provide any security features as all the packets received by the Hub are forwarded to all the connected nodes. As none of of the packets are filtered in any way, malicious observer can gain full visibility on what is going on the network. Moreover, in addition to security related constraints, higher transfer rate is not possible in a hub based system. Because in hub based network only one node can send traffic at a time, all other nodes will remain inactive when one node is sending information to another. So, such obsolete technology is almost non-existent in recent times. So, our proposed scheme should work in most of the networking environment.

9 Conclusion

IPv6 is the future of network addressing; so, the security of this protocol should be embedded into it. Unfortunately, as of now, a simple forged NA packet can cause significant damage. The protection measures currently available are complex and also introduces other side effects. We propose to rectify this problem at the protocol level. In this paper, we proposed a scheme that can solve MAC address

override attack in IPv6 with minimal overhead. As the proposed scheme is baked in IPv6, it would work out-of-the-box without additional configurations. Experimental results from NS3 implementation show that the proposed scheme can effectively identify fake NA packet and thus, foil the attack.

The primary concern of this mechanism is key transportation and verification; we plan to design cryptographic procedure for secure sharing of PSK. This can be done by using various hashing methods where subsequent keys could be hashed. Thus, it will be quite difficult for the attacker to obtain information on the key even if the there is a verification request. Additionally, since it is extremely difficult to track all the changes in the protocol level, further studies should be conducted to ensure the all other functionalities in NDP can be made secure using a pre-shared key mechanism. As a key has been already shared in this scheme, further trust relationship can be built on top of that which may lead towards trusting gateway and verifying other nodes in the network as well.

References

[1] A. S. Tanenbaum, Computer networks, Prentice Hall, Reading, MA, 2003.

[2] A. N. A. Ali, "Comparison study between IPV4 & IPV6," International Journal of Computer Science Issues (IJCSI), **9**(3), 314, 2012.

[3] M. Anbar, R. Abdullah, R. M. A. Saad, E. Alomari, S. Alsaleem, "Review of Security Vulnerabilities in the IPv6 Neighbor Discovery Protocol," Lecture Notes in Electrical Engineering, 603–612, 2016, doi:10.1007/978-981-10-0557-2_59.

[4] S. Mathi, L. Srikanth, "A New Method for Preventing Man-in-the-Middle Attack in IPv6 Network Mobility," in Advances in Electrical and Computer Technologies, 211–220, Springer, 2020.

[5] D. Plummer, "An Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware," 1982.

[6] S. M. Bellovin, "Security Problems in the TCP/IP Protocol Suite," SIGCOMM Comput. Commun. Rev., **19**(2), 32–48, 1989, doi:10.1145/378444.378449.

[7] S. Mcclure, S. Shah, S. Shah, Web hacking : attacks and defense, Addison-Wesley, 2005.

[8] "Dsniff." www.monkey.org, www.monkey.org/ dugsong/dsniff/. Accessed 24 Feb. 2020.

[9] "Oxid.it." www.oxid.it, www.oxid.it/cain.html. Accessed 11 Feb. 2020.

[10] "Ettercap Home Page." Www.ettercap-Project.org, www.ettercap-project.org/. Accessed 15 Feb. 2020.

[11] T. Narten, W. A. Simpson, E. Nordmark, H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)," 2007.

[12] D. J. Tian, K. R. B. Butler, J. I. Choi, P. McDaniel, P. Krishnaswamy, "Securing ARP/NDP From the Ground Up," IEEE Transactions on Information Forensics and Security, **12**(9), 2131–2143, 2017, doi:10.1109/tifs.2017.2695983.

[13] W. Liu, P. Ren, D. Sun, Y. Zhao, K. Liu, "Study on attacking and defending techniques in IPv6 networks," 2015 IEEE International Conference on Digital Signal Processing (DSP), 2015, doi:10.1109/icdsp.2015.7251328.

[14] J. Doyle, J. Carroll, CCIE Professional Developemnt. Routing TCP/IP, volume 1, Cisco Press, 2nd edition, 2006.

[15] A. Conta, M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification," 2006.

[16] C. Castelluccia, "Cryptographically Generated Addresses for Constrained Devices*," Wireless Personal Communications, **29**(3/4), 221–232, 2004, doi:10.1023/b:wire.0000047065.81535.84.

[17] P. Sumathi, S. Patel, "Secure Neighbor Discovery (SEND) Protocol challenges and approaches," 2016 10th International Conference on Intelligent Systems and Control (ISCO), 2016, doi:10.1109/isco.2016.7726976.

[18] B. Stockebrand, IPv6 in Practice, Springer Berlin Heidelberg, 2007, doi:10.1007/978-3-540-48001-3.

[19] K. S. Dar, I. Javed, S. A. Ammar, S. K. Abbas, S. Asghar, M. A. Bakar, U. Shaukat, "A survey-data privacy through different methods," Journal of Network Communications and Emerging Technologies (JNCET) www. jncet. org, **5**(2), 2015.

[20] N. Ahmed, A. Sadiq, A. Farooq, R. Akram, "Securing the Neighbour Discovery Protocol in IPv6 State-ful Address Auto-Configuration," 2017 IEEE Trustcom/BigDataSE/ICESS, 2017, doi:10.1109/trustcom/bigdatase/icess.2017.225.

[21] "Online Password Calculator." Lastbit.com, lastbit.com/pswcalc.asp. Accessed 24 Feb. 2020.

[22] Nsnam. "Ns-3." Ns-3, 2019, www.nsnam.org/. Accessed 16 Dec. 2019.

[23] Wireshark Foundation. "Wireshark · Go Deep." Wireshark.org, 2016, www.wireshark.org/. Accessed 6 Feb. 2020. .

[24] T. Cheneau, A. Boudguiga, M. Laurent, "Significantly improved performances of the cryptographically generated addresses thanks to ECC and GPGPU," Computers & Security, **29**(4), 419–431, 2010, doi:10.1016/j.cose.2009.12.008.

[25] A. Ferrante, V. Piuri, J. Owen, "IPSec hardware resource requirements evaluation," in Next Generation Internet Networks, 2005, 240–246, 2005.