IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 15, Issue 2 (Nov. - Dec. 2013), PP 66-75
www.iosrjournals.org

IPv6: Threats Posed By Multicast Packets, Extension Headers and Their Counter Measures

Santosh Naidu P[1], Amulya Patcha[2]
[1,2]Department of Computer Science and Engineering

ABSTRACT: *Security issues concerning the spreading Internet Protocol version 6 (IPv6) is one of the major issues in the world of networking today. Since it is not the default network protocol deployed nowadays (but systems are migrating slowly from ipv4 to ipv6) there are no best practices from the point of network administrators, nor are any guarantees that implemented IPv6 protocol stacks and security techniques without any bugs. This paper addresses some security concerns like extensive use of multicast packets and extension headers and its countermeasures.*

Keywords: *multicast, extension headers, reconnaissance, rogue dhcpv6 server spoofing, dual-stack, tunnels, Nat, ping of death*

I. Introduction

IPv6 is the next-generation internet protocol. The current version (IPv4) is running out of addresses and has become too complex too manage.IPv6 leaps from 32 to 128-bit addressing. It has insurmountable number of IP addresses that can be used to identify and connect possibly everything in the world over the Internet. Though, the IPv6 was developed and introduced in the 90s [1], it is still a new technology to most Internet users. As a new technology, the users need to understand, learn and adapt to it.

As the deployment of IPv6 proceeds, security issues appear simultaneously. That is, existing security attacks against IPv4 changed to attack IPv6 networks and clients, while new IPv6-only threats arise from the new protocol specification.

We show the so far known security issues concerning IPv6. We first describe each vulnerability in detail before we exploit it with the appropriate tools and scripts. After each issue we summarize all countermeasures for IPv6 nodes and firewalls. Here we deal with security issues that arise directly from the specification of IPv6 such as the numerous uses of multicast packets and the chaining of extension headers.

There are some security vulnerabilities that arise directly from the specification of IPv6. Some of these issues cannot be fixed without changing the protocol itself. Unlike some application layer attacks that can be closed by fixing the appropriate software, these IPv6 protocol specific vulnerabilities will remain the same. The IETF sometimes revives the protocols, but in some instances, the IETF leaves it up to the developers of IP systems to correct the specifications dependencies. Therefore it is crucial for a network administrator to understand the security issues that come with IPv6 and to know how to thwart them.

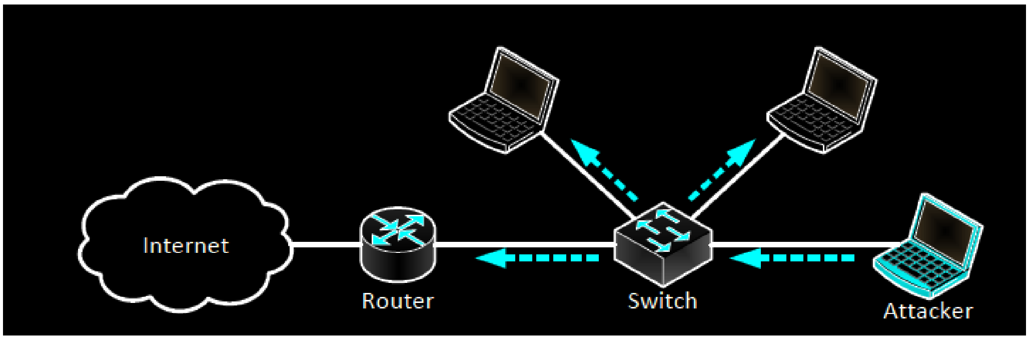
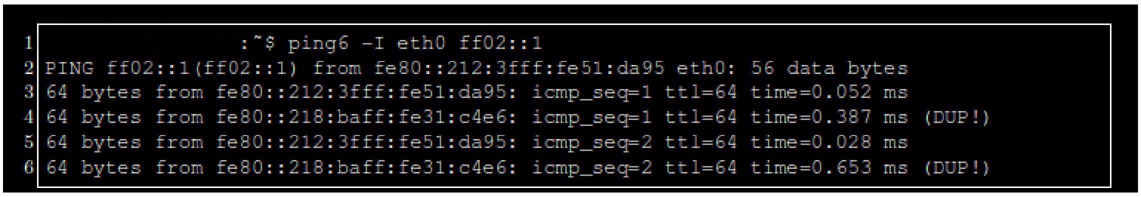


Figure 1: Pinging the all-nodes multicast address ff02::1: all IPv6 nodes on the same link receive the echo-request message since the switch forwards it on all ports.

II. Multicast Packets

IPv6 uses multicast packets for Neighbour Discovery such as resolving link-layer addresses or receiving Router Advertisements which are used during the process of auto configuration. Each IPv6 node joins at least all the nodes multicast address ff02::1 and its solicited-node multicast address. Despite its usefulness, an attacker can also send any packet to a multicast address. He can use this situation for speeding up the reconnaissance phase of a network or even run a denial of service (DoS) attack against all IPv6 nodes at once. This paper discusses only local multicast messages but not global multicast routing features; as they are used for multimedia streams on the Internet. Furthermore, many ICMPv6 messages are used in this for attacking hosts with multicast messages. However, these attacks merely rely on the concept of multicast but not on ICMPv6 vulnerabilities.

All attacks that send messages to multicast addresses within the link-local scope ff02: : are applicable, only if the hacker already resides on the local area network as they are not routed, as stated in RFC 4291: Routers must not forward any packets with Link-Local source or destination addresses to other links.



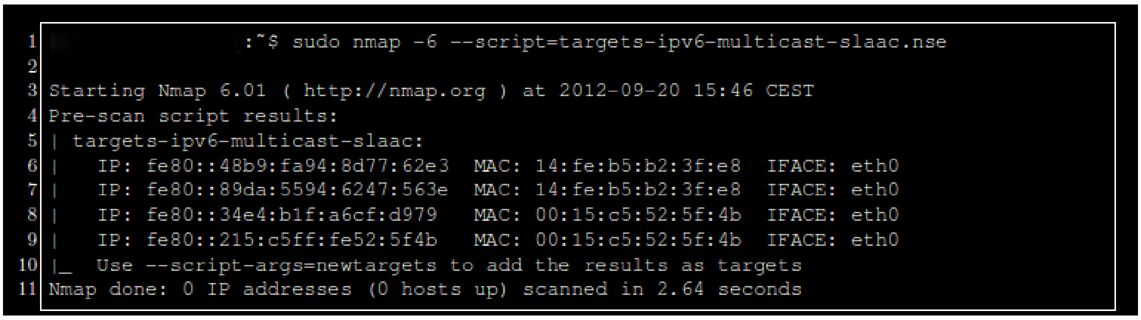
```
                    :~$ ping6 -I eth0 ff02::1
PING ff02::1(ff02::1) from fe80::212:3fff:fe51:da95 eth0: 56 data bytes
64 bytes from fe80::212:3fff:fe51:da95: icmp_seq=1 ttl=64 time=0.052 ms
64 bytes from fe80::218:baff:fe31:c4e6: icmp_seq=1 ttl=64 time=0.387 ms (DUP!)
64 bytes from fe80::212:3fff:fe51:da95: icmp_seq=2 ttl=64 time=0.028 ms
64 bytes from fe80::218:baff:fe31:c4e6: icmp_seq=2 ttl=64 time=0.653 ms (DUP!)
```

Figure 2: Reconnaissance: Pinging the All-Nodes Multicast Address

2.1. Reconnaissance Phase

Before the attacker can attack some hosts on the network, he first needs to find them. This is called the reconnaissance phase or reconnaissance attack. After the attacker has found some online hosts, he can scan specific layer 4 ports in order to detect vulnerabilities on certain applications or services. This deals with finding only active hosts on the network. The subsequent scanning of ports remains the same as with IPv4.

Since a complete ping sweep over all IPv6 addresses within an IPv6 subnet is not feasible due to the large address space, a basic reconnaissance phase can be done by sending echo-requests or some other (falsified) packets to the all-nodes multicast address. A straightforward approach is to ping the all-nodes multicast address as depicted in Figure 1. An example from a Linux machine is shown in Figure 2. Note that the source interface must be specified because the operating system needs to know from which interface it should source the link-local ping. Since the ping program expects only one answer at a time, multiple answers are marked with a duplicate statement (DUP!).There are no global unicast IPv6 addresses shown because the ping6 command is issued from the link-local address and therefore all nodes reply with a message sent from their link-local addresses too. Furthermore, not all IPv6 nodes reply to an echo-request message sent to a multicast address. For example, Windows 7 does not reply to a ping command by default and can therefore not be found using this reconnaissance method. This behaviour is correct due to the ICMPv6 specification in RFC 4443: An Echo Reply SHOULD be sent in response to an Echo Request message sent to an IPv6 multicast or any cast address. That is, an echo-reply to a multicast echo-request is not mandatory. Of course any other multicast addresses can be pinged, such as the all-routers address ff05: :2 or to the all-dhcp-server address ff05: :1:3 in order to reveal the appropriate nodes.



```
[redacted]:~$ sudo nmap -6 --script=targets-ipv6-multicast-slaac.nse

Starting Nmap 6.01 ( http://nmap.org ) at 2012-09-20 15:46 CEST
Pre-scan script results:
| targets-ipv6-multicast-slaac:
|   IP: fe80::48b9:fa94:8d77:62e3  MAC: 14:fe:b5:b2:3f:e8  IFACE: eth0
|   IP: fe80::89da:5594:6247:563e  MAC: 14:fe:b5:b2:3f:e8  IFACE: eth0
|   IP: fe80::34e4:b1f:a6cf:d979   MAC: 00:15:c5:52:5f:4b  IFACE: eth0
|   IP: fe80::215:c5ff:fe52:5f4b   MAC: 00:15:c5:52:5f:4b  IFACE: eth0
|_  Use --script-args=newtargets to add the results as targets
Nmap done: 0 IP addresses (0 hosts up) scanned in 2.64 seconds
```

Figure 3: Reconnaissance: Nmap Script for IPv6 Node Discovery

The network mapper Nmap [2] also includes some ipv6 related scripts that extend the reconnaissance phase from simply sending an echo-request by constructing some spoofed packets. A script can be executed by adding the option --script=scriptname.nse to the call of Nmap.

2.2. Amplification Attack (Smurf)

The reconnaissance phase described above, uses multicast messages to reveal potential targets on the network but does not actually attack them. For an attacker, the full potential of the multicast methodology, results in sending spoofed messages in order to attack all host at once or at least to use all hosts on a network to attack a single host. The former one could be used to run a denial of service attack (DoS) against the whole network, while the latter one is a kind of a distributed denial of service attack (DDoS); in which many hosts try to interrupt a single host. These types of attacks are called amplification attacks because they multiply the quantity of packets, i.e., the payload on the network. In the worst scenario, a single packet sent to a multicast group is multiplied by every node on the link.RFC 4443 states: An ICMPv6 error message MUST NOT be originated as a result of receiving a packet destined to an IPv6 multicast address. But this only reduces the possibility of some attacks or accidentally sent packets, which could result in amplification, but does not reduce the possibility for an attacker who is able to send any type of spoofed packets. In addition, there are two exceptions to that rule in which the node should send an ICMPv6 error message after receiving a multicast packet.

If an attacker sends packets with a spoofed source address to a multicast group and all nodes in that group respond to that message, the spoofed source address, i.e., the address of the victim, will be overwhelmed with traffic (refer to Figure 4). A simple tool that sends echo-requests to the all-nodes multicast address ff02::1 is smurf6 from the THC-IPv6 attacking toolkit [3]. Its syntax is. /smurf6 interface victim-ip [multicast-network-address] via specifying the multicast-network-address, the attacker can send spoofed echo-request packets to any other multicast group. Since Microsoft Windows does not answer to echo-request packets, this attack has more impact if it is used in environments with other operating systems such as Linux. The IPv6 address of the victim can also reside on a remote subnet. In that scenario, all local nodes send echo-replies via their default router to the remote host. That is, this attack would not only influence the remote victim but also the local network. However, if an attacker wants to disable Internet activity via a denial of service attack, he could run a few other DoS attacks against the router directly; which is called Router Advertisement spoofing. One further idea might be to set the victim-ip address to the all-nodes address, in order to have all nodes on the link to send back an echo-reply to all the nodes. In theory, this would interrupt the local network completely since it is a vast amplification. A correct implemented IPv6 stack will not handle such packets since the RFC 4291 specifies that multicast addresses must not be used as source addresses in IPv6 packets or appear in any Routing header. Therefore, IPv6 nodes should never answer to packets which have a multicast source address.

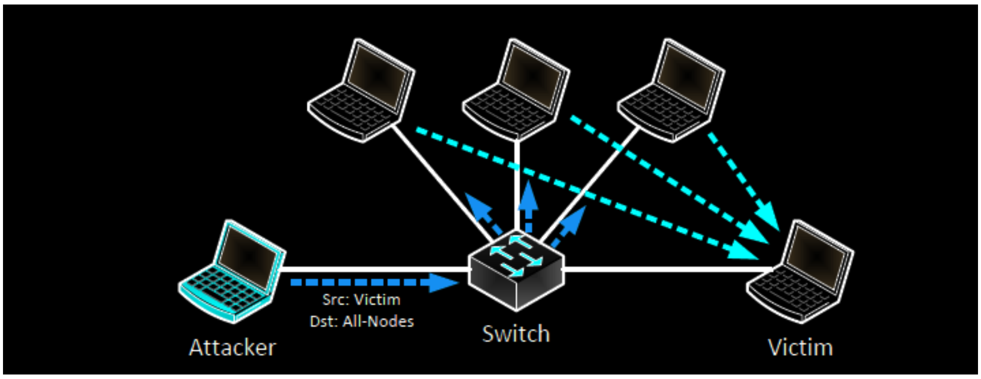
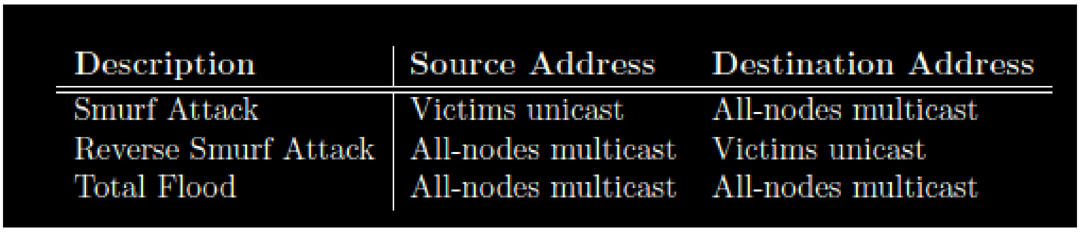


Figure 4: Smurf Attack: the attacker sends a single spoofed packet which is ampli_ed by all IPv6 nodes on the link and directed to the victim.

Another smurf tool which operates quite differently is rsmurf6 from the THC-IPv6 attacking toolkit [3]. It sends echo-requests from a source address of ff02::1 (all-nodes multicast address) to the destination victim-ip address. It uses the following syntax: ./rsmurf6 interface victim-ip. Theoretically, this would attack the victim's subnet and not the local subnet on which the attacker resides. If the IPv6 protocol stack of the victim is incorrectly implemented and answers with an echo-reply ICMPv6 message to the all-nodes multicast address, this results in an amplification attack on the remote network, since each echo-request packet sent by the attacker is answered by an echo-reply to all IPv6 nodes on that remote link. But as already mentioned, IPv6 nodes must not answer to packets with a multicast source address and therefore this attack should not work anymore as most IPv6 implementations should be without major bugs.

RFC 4443 (Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification) describes a possible denial of service attack, in which all IPv6 nodes would answer to a spoofed packet (operating systems that do not answer to normal echo requests packets). A malicious node can send a multicast packet with an unknown destination option marked as mandatory, with the IPv6 source address of a valid multicast source. A large number of destination nodes will send an ICMP Parameter Problem Message to the multicast source, causing a denial-of-service attack. On the other hand, the use of reverse path forwarding checks (to eliminate loops in multicast forwarding) automatically limits the range of addresses that can be spoofed, (RFC 4942). Since there is more efficient denial of service attacks within the IPv6/ICMPv6 protocols, the THC-IPv6 attacking toolkit does not implement this type of attack though it would not be that difficult. Figure 5 summarizes the mentioned smurf attacks and its addresses.



Description	Source Address	Destination Address
Smurf Attack	Victims unicast	All-nodes multicast
Reverse Smurf Attack	All-nodes multicast	Victims unicast
Total Flood	All-nodes multicast	All-nodes multicast

Figure 5: Source & Destination Addresses of Smurf Attacks

In summary, the concept of multicast in IPv6 has a few advantages such as eliminating broadcast packets which reduces network congestion, but it has also a few disadvantages such as the possibility for attackers to send spoofed multicast packets which will be delivered to any node participating in the appropriate multicast group. Since multicast packets are mandatory for a correct functioning of IPv6 nodes, it cannot be disabled entirely. Fortunately, the shown attacks are not that severe because they only reveal basic information about the IPv6 nodes such as their addresses, or they try to run DoS attacks. There are few more powerful DoS tools and even attacks that can gather confidential information via man-in-the-middle attacks, there is no reason for excessive fear about the concept of multicast.

III. Extension Headers

In IPv6, optional internet-layer information is encoded in separate headers that may be placed between the IPv6 header and the upper-layer header in a packet, (RFC 2460).Routers has a simple job since they only need to examine the IPv6 destination address and the Hop-by-Hop Options header. Firewalls that should enforce their security policy must recognize and parse through all existing extension headers since the upper-layer protocol information reside in the last header. An attacker is able to chain lots of extension headers in order to pass through firewalls and intrusion detections. He can also cause a denial of service attack, if an intermediary device or a host is not capable of processing lots of chained extension headers and might fail.

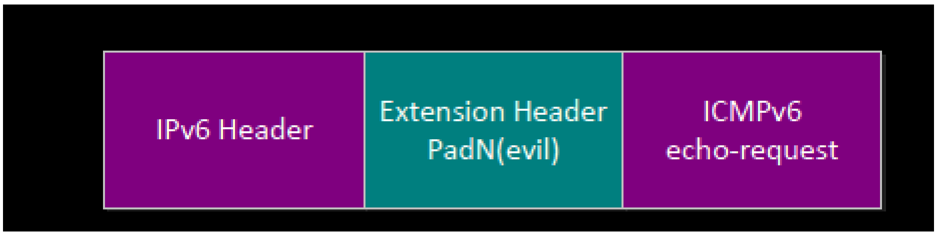
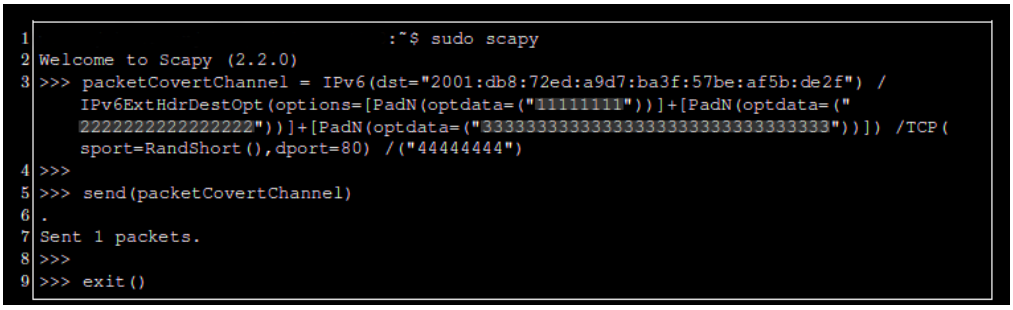


Figure 6: Covert Channel inside the Padding of an Extension Header.



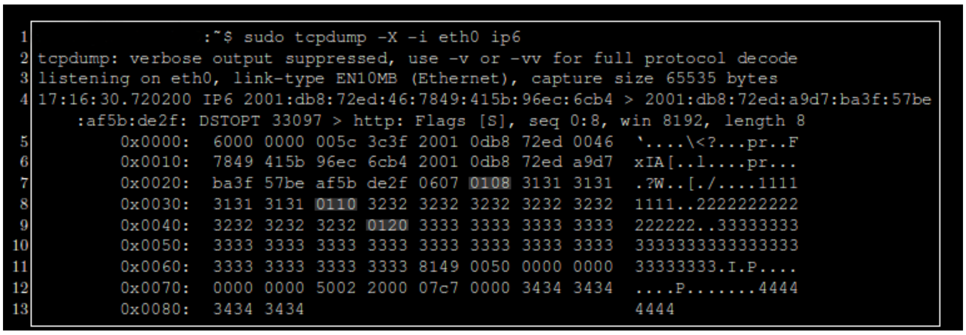
```
                    :~$ sudo scapy
Welcome to Scapy (2.2.0)
>>> packetCovertChannel = IPv6(dst="2001:db8:72ed:a9d7:ba3f:57be:af5b:de2f") /
    IPv6ExtHdrDestOpt(options=[PadN(optdata=("11111111"))]+[PadN(optdata=("
    2222222222222222"))]+[PadN(optdata=("33333333333333333333333333333333"))]) /TCP(
    sport=RandShort(),dport=80) /("44444444")
>>>
>>> send(packetCovertChannel)
.
Sent 1 packets.
>>>
>>> exit()
```

Figure 7: Scapy sends a message via a covert channel in the Destination Options header: three different PadN options are added, each with a different length.

3.1. Covert Channel in Hop-by-Hop and Destination Options Header

A covert channel is a path of communication that was not designed to be used for communication, [5, p. 440]. With a covert channel in a network, an attacker can deliver information by altering fields in a packet that are not intended to act as a storage for user-specific information, i.e., they are not the payload of the packet itself. As with many other TCP/IP protocols, some fields in IPv6 extension headers can be used for creating a covert channel.

RFC 2460 (Internet Protocol, Version 6 (IPv6) Specification) defines a PadN option, that is used to insert two or more octets of padding into the options area of a header. The option data field of this PadN is filled with zero-valued octets. If a firewall forwards the IPv6 packet, even if the PadN option does not contain only zero-valued octets, an attacker can use it as a covert channel (refer to Figure 6). Figure 7 shows an example in which we use Scapy [6] to send some messages embedded in PadN options. After the IPv6 header, a destination options header with three PadN options is constructed. (The attack behaves the same, if a Hop-by-Hop Options header is used instead.) Each PadN contains an option data with a different entry. The TCP header in the end, completes the IPv6 packet, but has no relevance for this attack. On the destination machine, we use tcpdump [7] in the hex and ASCII output mode (-X) in order to view all IPv6 packets. Figure 8 depicts the received message and reveals that all three PadN options arrive unbroken.



```
                :~$ sudo tcpdump -X -i eth0 ip6
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
17:16:30.720200 IP6 2001:db8:72ed:46:7849:415b:96ec:6cb4 > 2001:db8:72ed:a9d7:ba3f:57be
    :af5b:de2f: DSTOPT 33097 > http: Flags [S], seq 0:8, win 8192, length 8
        0x0000:  6000 0000 005c 3c3f 2001 0db8 72ed 0046  `....\<?...pr..F
        0x0010:  7849 415b 96ec 6cb4 2001 0db8 72ed a9d7  xIA[..l....pr...
        0x0020:  ba3f 57be af5b de2f 0607 0108 3131 3131  .?W..[./....1111
        0x0030:  3131 3131 0110 3232 3232 3232 3232 3232  1111..2222222222
        0x0040:  3232 3232 3232 0120 3333 3333 3333 3333  222222..33333333
        0x0050:  3333 3333 3333 3333 3333 3333 3333 3333  3333333333333333
        0x0060:  3333 3333 3333 3333 8149 0050 0000 0000  33333333.I.P....
        0x0070:  0000 0000 5002 2000 07c7 0000 3434 3434  ....P.......4444
        0x0080:  3434 3434                                4444
```

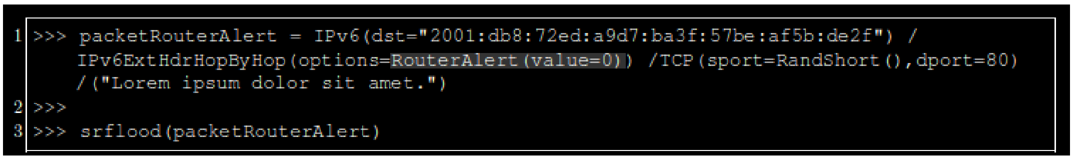
Figure 8: Covert channel message received: tcpdump indicates the three PadN options (first octet 0x01), each with a different length (second octet 0x08 to 0x20, i.e., 8 to 32 octets). On the right-hand side, the raw PadN values are shown.

Firewalls should drop packets that have multiple padding options, as well as packets that have more than 5 bytes of padding. Furthermore, firewalls should also drop padding that has anything other than 0's in the data field, [8, p. 32]. That is, a firewall that forwards this type of packet is inconsistent due to the RFCs and opens the possibility for a covert channel. This kind of attack is neither new to network administrators nor to attackers since there are many covert channels in the TCP/IP protocol family, which are investigated in other works such as [4] which implements a covert timing channel, [5, p. 446], by altering the timing of IP traffic or [14], which manipulates IPv4 headers as a type of a covert storage channel, [5, p. 446]. But it should be noted that, the IPv6 protocol provides even new possibilities for covert channels.

3.2. Router Alert DoS Attack in Hop-by-Hop Options Header

The Hop-by-Hop Options header is used to carry optional information that must be examined by every node along a packet's delivery path, (RFC 2460). One option is, the IPv6 Router Alert Option (RFC 2711) which tells routers to intercept the datagram and to look further into it. If this investigation is not done in efficient hardware, but if it is through slower software (slow path) the router might be vulnerable for a denial of service attack if it is flooded by many router alerts. This security issue is also described in the RFC: Gratuitous use of this option can cause performance problems in routers. A more severe attack is possible in which the router is flooded by bogus datagram containing router alert options. We therefore build such an IPv6 packet with a router alert and flood it to the destination.

Figure 9 shows the packet constructed with Scapy [6]. It contains a Hop-by-Hop Options header with a router alert option (value = 0x05) which tells the router that a Multicast Listener Discovery (MLD) message is in the IPv6 packet .The attached TCP datagram with a destination port of 80 (HTTP) is added in order to construct a normal looking TCP/IP packet. Since the srflood () does not flood the network with many thousands of packets per second, this constructed packet cannot overwhelm a router. The THC-IPv6 attacking toolkit [3] also provides a tool called denial6 which has a test-case (number 1) that sends exactly such messages:/denial6 interface destination test-case-number.



```
>>> packetRouterAlert = IPv6(dst="2001:db8:72ed:a9d7:ba3f:57be:af5b:de2f") /
    IPv6ExtHdrHopByHop(options=RouterAlert(value=0)) /TCP(sport=RandShort(),dport=80)
    /("Lorem ipsum dolor sit amet.")
>>>
>>> srflood(packetRouterAlert)
```

Figure 9: Router Alert flooding with Scapy: the Hop-by-Hop Options header contains a router alert option and is then flooded to the destination.

The best current practice RFC 6398 (IP Router Alert Considerations and Usage) also explains that there are currently no methods to distinguish between a spoofed router alert messages, i.e., an attack, and a legitimate router alert option, in a nutshell, the IP Router Alert Option does not provide a convenient universal mechanism to accurately and reliably distinguish between IP Router Alert packets of interest and unwanted IP Router Alert packets. This, in turn, creates a security concern when the IP Router Alert Option is used, because, short of appropriate router implementation- specific mechanisms, the router slow path is at risk of being flooded by unwanted traffic. One chance to defeat such attacks can be the implementation of rate limits for router alerts.

3.3. Routing Header 0 (deprecated)

With an inserted Routing header, an IPv6 packet visits all given IPv6 addresses via its traversal to the real destination node (refer to Figure 2.3). While a Routing header type 2 (RH2) is used for Mobile IPv6, a type 0 Routing header (RH0) can be used for any IPv6 packets. The usage of a Routing header type 0 has several security issues:

DoS between two nodes: If the Routing header stores the IPv6 addresses of two nodes and repeats them a few times, the IPv6 packet will bounce between these two nodes and will force congestion on the path. If the payload of the IPv6 packet and the bandwidth of the attacker are huge enough, this can lead to a denial of service attack.

Firewall bypassing: An IPv6 packet with the appropriate routing header may be able to bypass certain firewall configurations. Consider the following example (Figure 10): a firewall with three interfaces (outside, DMZ, inside) allows certain connections from the outside to the DMZ but not from outside to inside.

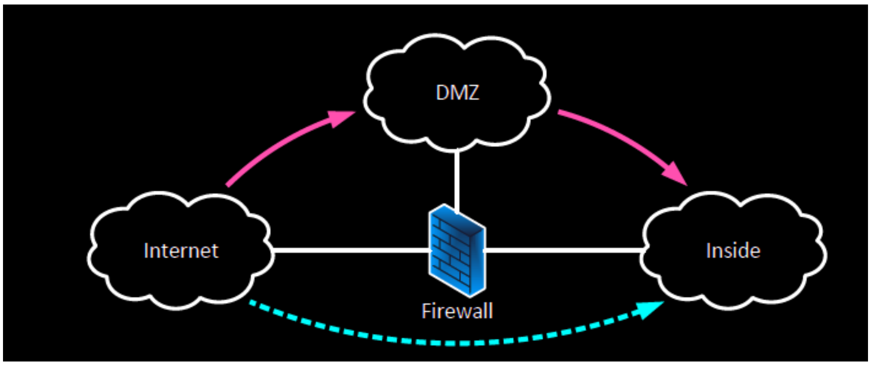


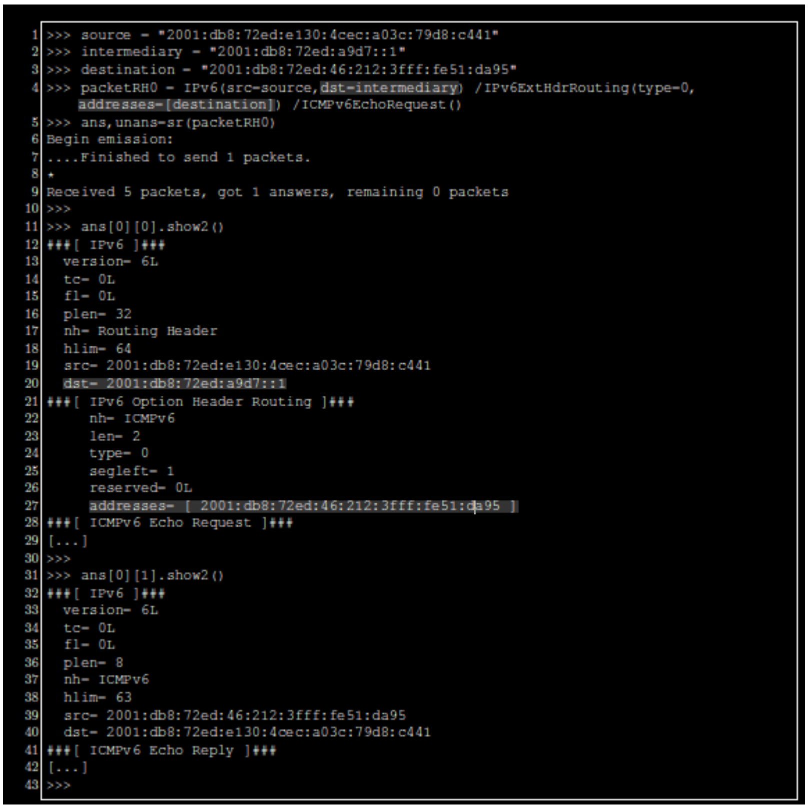
Figure 10. Firewall Bypassing with RH0: the firewall would block a direct connection from the Internet to the inside network, but allows it through the DMZ.

Furthermore, some connections from the DMZ to the inside are allowed. An attacker on the outside can now send a packet to the DMZ with a routing header type 0 which contains a final destination address of the inside network. With the usage of the packet manipulation program Scapy [6] we construct a simple ICMPv6 echo-request packet with a routing header type 0 (Figure 11). We declare three IPv6 addresses: the source of the packet, the intermediary node, and the destination node. Note that the IPv6 packet is sent to the intermediary IPv6 address first, while the real destination IPv6 address resides in the routing header2 as the intermediary node, we use an interface of a Cisco router with an older IOS version which does not block routing headers by default. Therefore, the intermediary node processes the routing header and then places the routing header address in the destination of the IPv6 packet and finally forwards it to the real destination. The figure also shows the echo-reply answer which is received directly from the destination, i.e., without a routing header.

Figure 12.The constructed packet is first sent to the intermediary node i1, then bounced between i2 and i1, and finally transmitted to the destination. This is similar to the IPv6 Routing Header Security presentation by Philippe Biondi and Arnaud Ebalard on the CanSecWest Conference 2007, [12], which shows a few real world examples of the RH0 DoS attack.

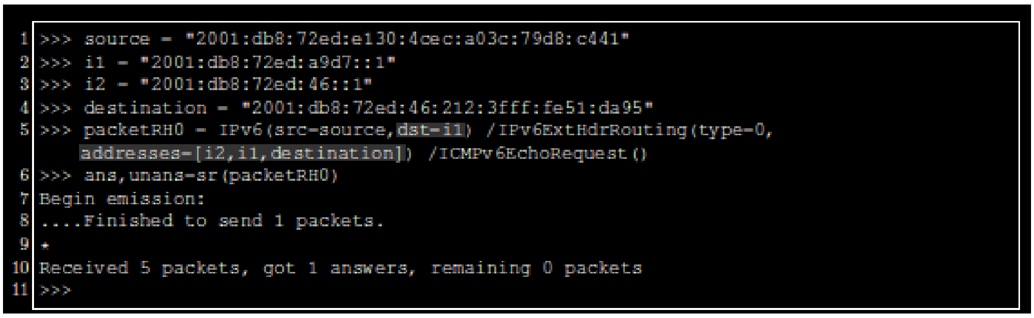
For example, they made some time measurements for the amplification effect and also computed the additional traffic that was placed between two nodes. Such information combined, results in a potential vast denial of service attack since an attacker can buffer traffic between two routers in the Internet.

The RH0 packets must be sent with appropriate RRT values, in order to hit a victim with all the stored traffic at the same time. They also showed that some trace route commands combined with Routing headers reveal several routers on the Internet that would not be revealed without the usage of RH0.



```
>>> source = "2001:db8:72ed:e130:4cec:a03c:79d8:c441"
>>> intermediary = "2001:db8:72ed:a9d7::1"
>>> destination = "2001:db8:72ed:46:212:3fff:fe51:da95"
>>> packetRH0 = IPv6(src=source,dst=intermediary) /IPv6ExtHdrRouting(type=0,
    addresses=[destination]) /ICMPv6EchoRequest()
>>> ans,unans=sr(packetRH0)
Begin emission:
....Finished to send 1 packets.
*
Received 5 packets, got 1 answers, remaining 0 packets
>>>
>>> ans[0][0].show2()
###[ IPv6 ]###
  version= 6L
  tc= 0L
  fl= 0L
  plen= 32
  nh= Routing Header
  hlim= 64
  src= 2001:db8:72ed:e130:4cec:a03c:79d8:c441
  dst= 2001:db8:72ed:a9d7::1
###[ IPv6 Option Header Routing ]###
     nh= ICMPv6
     len= 2
     type= 0
     segleft= 1
     reserved= 0L
     addresses= [ 2001:db8:72ed:46:212:3fff:fe51:da95 ]
###[ ICMPv6 Echo Request ]###
[...]
>>>
>>> ans[0][1].show2()
###[ IPv6 ]###
  version= 6L
  tc= 0L
  fl= 0L
  plen= 8
  nh= ICMPv6
  hlim= 63
  src= 2001:db8:72ed:46:212:3fff:fe51:da95
  dst= 2001:db8:72ed:e130:4cec:a03c:79d8:c441
###[ ICMPv6 Echo Reply ]###
[...]
>>>
```

Figure 11. Routing Header RH0 Attack: the source node sends an echo-request message, but the first destination is the intermediary node. The inserted Routing header stores the real destination IPv6 address. The echo-reply is sent without a Routing header directly to the originating source node.



```
>>> source = "2001:db8:72ed:e130:4cec:a03c:79d8:c441"
>>> i1 = "2001:db8:72ed:a9d7::1"
>>> i2 = "2001:db8:72ed:46::1"
>>> destination = "2001:db8:72ed:46:212:3fff:fe51:da95"
>>> packetRH0 = IPv6(src=source,dst=i1) /IPv6ExtHdrRouting(type=0,
    addresses=[i2,i1,destination]) /ICMPv6EchoRequest()
>>> ans,unans=sr(packetRH0)
Begin emission:
....Finished to send 1 packets.
*
Received 5 packets, got 1 answers, remaining 0 packets
>>>
```

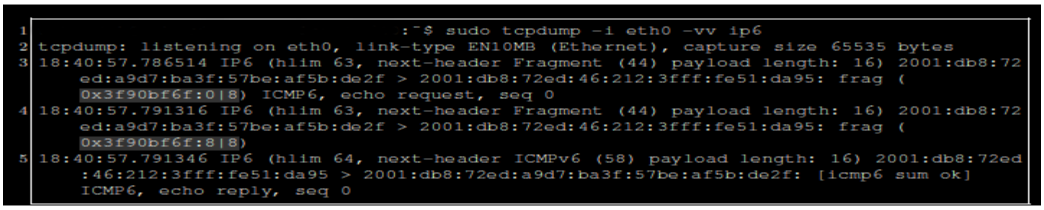
Figure 12: Routing Header RH0 Attack bouncing between two intermediary nodes before arriving at the real destination.

Due to the fact that the RH0 has these major security flaws, RFC 5095 deprecates the overall usage of routing header type 0. That is, an IPv6 node that receives a packet with a destination address assigned to it and that contains an RH0 extension header MUST NOT execute the algorithm. Furthermore, it MUST send an ICMPv6 parameter problem (code 0) message back to the source. The RFC further advises to implement ingress filtering on perimeter firewalls in order to block all IPv6 packets that contain a RH0 header. That is, not only the interfaces of a firewall must not process any RH0 packets, but all traversing IPv6 packets with a RH0 should also be blocked. One way to test whether an ISP has implemented ingress filtering is to send a boomerang packet to a destination which processes RH0 packets and check if this packet comes back. In [12], a suitable Scapy one-liner is presented: sr1 (IPv6 (src=us, dst=tgt) /IPv6ExtHdrRouting (addresses= [us]) /ICMPv6EchoRequest ()).

Note that the deprecation of RH0 for IPv6 was in 2007; many years after the security issues with source routing for IPv4 were known. For example, in 1589, S. M. Bellovin describes source routing as a way for attackers to hide their own IP address while still receiving answers from the attacked machines, [13].

3.4. Firewall Evasion with Fragment Header

Fragmentation in IPv6 is used by the originating node in order to have the packet size fitted into the path maximum transmission unit. Since IPv6 requires a MTU greater than 1280 bytes, (RFC 2460),



```
[redacted]:~$ sudo tcpdump -i eth0 -vv ip6
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
18:40:57.786514 IP6 (hlim 63, next-header Fragment (44) payload length: 16) 2001:db8:72
    ed:a9d7:ba3f:57be:af5b:de2f > 2001:db8:72ed:46:212:3fff:fe51:da95: frag (
    0x3f90bf6f:0|8) ICMP6, echo request, seq 0
18:40:57.791316 IP6 (hlim 63, next-header Fragment (44) payload length: 16) 2001:db8:72
    ed:a9d7:ba3f:57be:af5b:de2f > 2001:db8:72ed:46:212:3fff:fe51:da95: frag (
    0x3f90bf6f:8|8)
18:40:57.791346 IP6 (hlim 64, next-header ICMPv6 (58) payload length: 16) 2001:db8:72ed
    :46:212:3fff:fe51:da95 > 2001:db8:72ed:a9d7:ba3f:57be:af5b:de2f: [icmp6 sum ok]
    ICMP6, echo reply, seq 0
```

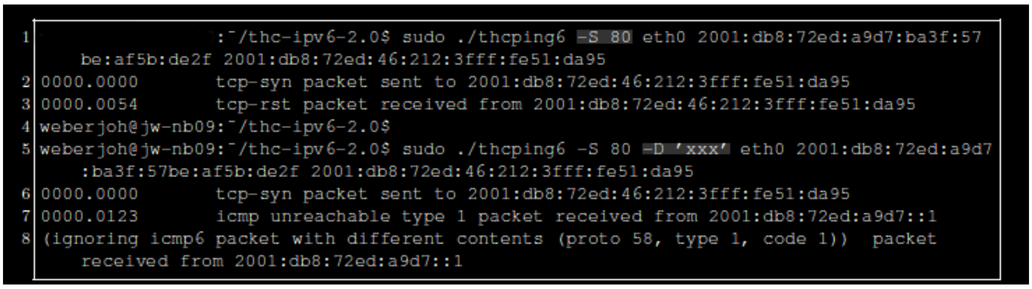
Figure 13: Fragment header: neither the router nor the Linux machine discard the tiny fragment. The computer sends an echo-reply.

Fragmented packets should not be smaller than 1280 bytes, except the last fragment with the more fragments M flag set to zero.

An attacker could try to bypass a firewall or IDS/IPS inspection by either fragmenting his packets too many small fragments or by chaining a few extension headers before his payload in order to place the upper-layer information in a rearmost packet. The combination of multiple extension headers and fragmentation in IPv6 creates the potential that the Layer 4 protocol is not included in the first packet of a fragment set, making it difficult to enforce Layer 4 policy on devices that do not do fragment reassembly, [16].

We execute two different firewall tests concerning the Fragment header. The first one tests whether the firewall allows tiny fragments to pass, i.e., fragments smaller than 1280 bytes. We use the same test method and the proposed Scapy script presented by Antonios Atlasis at the Black Hat Conference 2012, [15, p. 8]. The script is listed in Appendix A.4. It is called with four arguments, simple fragmentation source-address destination-address length-of-fragments offset-of-second-fragment. To test the smallest fragments, we call it with 1 1 to have two fragments, both with a length of 8 bytes. Figure 13 shows a capture from the destination node. Both fragments arrive and the node answers with an echo-reply. The highlighted regions show the fragment identification, the fragment offset, and the length of the Payload. Note that in our tests it is not interesting at all if the destination node replies, but only if the intermediary firewall forwards both tiny fragments. Furthermore, we do not test a real firewall evasion by spreading the upper-layer information over several fragments, but only whether the firewall lets tiny fragments pass in general, or not. In general, large amounts of fragmented traffic have been used as an early indicator of an intrusion attempt because most baselines of Internet traffic indicate that the percentage of fragmented traffic is low, [16, and 11].

The second test reveals whether the firewall allows a fragment to pass even though the upper-layer information is not present in the (first) fragment. We use the thcping6 tool from the THC-IPv6 attacking toolkit [3] which constructs special echo-requests or TCP SYN packets. Figure 14 shows two TCP SYN openings with the first one sent to port 80 (HTTP, Line 1). For the second packet we add a large Destination options header to have the packet fragmented in order to move the TCP header into the second fragment. As Line 7 reveals, the intermediary firewall sent an ICMPv6 unreachable error message back to the source because it did not forward the packet. This shows that they did not reassemble the fragments in order to decide whether the TCP port is allowed due to the policy, but did discard the fragment directly. In the case of a forwarded packet we can repeat the test with a TCP port that is not allowed due to the policy. This would reveal whether the firewall simply allows all fragments without checking the upper-layer information, or if it actually reassembles the complete packet.



```
1 [redacted]:~/thc-ipv6-2.0$ sudo ./thcping6 -S 80 eth0 2001:db8:72ed:a9d7:ba3f:57
      be:af5b:de2f 2001:db8:72ed:46:212:3fff:fe51:da95
2 0000.0000       tcp-syn packet sent to 2001:db8:72ed:46:212:3fff:fe51:da95
3 0000.0054       tcp-rst packet received from 2001:db8:72ed:46:212:3fff:fe51:da95
4 weberjoh@jw-nb09:~/thc-ipv6-2.0$
5 weberjoh@jw-nb09:~/thc-ipv6-2.0$ sudo ./thcping6 -S 80 -D 'xxx' eth0 2001:db8:72ed:a9d7
      :ba3f:57be:af5b:de2f 2001:db8:72ed:46:212:3fff:fe51:da95
6 0000.0000       tcp-syn packet sent to 2001:db8:72ed:46:212:3fff:fe51:da95
7 0000.0123       icmp unreachable type 1 packet received from 2001:db8:72ed:a9d7::1
8 (ignoring icmp6 packet with different contents (proto 58, type 1, code 1))  packet
      received from 2001:db8:72ed:a9d7::1
```

Figure 14: Fragment Header: firewall evasion with large fragment: the second TCP SYN request is fragmented and denied by the firewall.

If a firewall is able to parse through all extension headers without limiting the bandwidth of the link, it is still hard to decide which default behaviour a firewall should enforce if it recognizes some unknown extension headers. From a security perspective, a firewall should discard every IPv6 packet with an unknown extension header since it could be an attack. But this has the drawback that future extension headers will be blocked, too, if the list of known extension headers is not promptly updated.

A more detailed work with different fragmentation test cases against several operating systems was presented by Antonios Atlasis at the Black Hat Conference 1612, [15]. Furthermore, in the paper Target-Based Fragmentation Reassembly, Judy Novak from the Source fire Vulnerability Research Team shows different modes of fragment reassembly according to different operating systems, and further investigates how the open source network IPS Snort [10] can be used to defeat them, [12].

IV. Countermeasures and Firewall's Best Practices

4.1. Countermeasures for an IPv6 Node

A straightforward countermeasure for IPv6 nodes is to block any echo-request messages, i.e., not answering with echo-replies, or at least not answering if the requested IPv6 address is a multicast address. But since the more profound reconnaissance tools such as the Nmap SLAAC script use other techniques than echo-requests, IPv6 nodes do not have a chance to stay undetected on a local network if they communicate with each other. Also simply blocking all echo-request messages might not be the best choice for a network administrator since legitimate tasks could not use these features anymore, too. For example, to monitor intermediary devices like switches, routers or WLAN access points, a network management utility needs to ping these devices.

The standard RFCs should be implemented correctly in order to not answer to packets that are sourced from a multicast address. One countermeasure to not allow any node to reply to each message at very fast rates is to implement rate limiting for ICMP messages: They should be rare in every network so that a rate limit (10 messages/sec) can permit the correct use of those messages (path MTU discovery) while blocking the amplification attack, [8, p. 76]. It should be noted that these types of amplification attacks only work if the attacker already resides on the local subnet. In addition, he can only attack the local subnet, too. Hence, port statistics of the network infrastructure can reveal the attacking host quickly.

It is not easy to give recommendations concerning extension headers because it is a balance between security and usability. For example: even it is unusual to receive lots of tiny fragments what in fact could be an attack, it is not specified by the RFCs to block them. Therefore, a security administrator must decide whether to implement strict rules or to have the IPv6 stack work without any troubles.

4.2. Firewall's Best Practices

Block all site-local scope (ff05::) and variable scope (ff0x::) multicast addresses at the network perimeter in order to not reveal IPv6 addresses across the local network, (RFC 4942).For example, this would prevent an attacker who already sits in the Demilitarized Zone (DMZ) to ping the all-dhcp-servers multicast address ff05::1:3. It also blocks the forwarding of smurf attacks from the outside to an inside network.

IDS/IPS: Provide a simple Intrusion Prevention System (IPS) in order to detect and block massive echo-request floods from the Internet. At least the firewall should detect such ping storms and add an entry in its log _les or its network management software.

IDS: In order to recognize the SLAAC script from Nmap a firewall should produce an alert if unknown Router Advertisements are seen on the network.

Activate unicast reverse path forwarding (uRPF), on all inside interfaces, [8, p. 66]. RPF, described in RFC 3704, is a feature in which the router checks the source address of incoming packets against its routing table and only forwards the packets if the source address can be reached via the interface the packet was received, i.e., if the answer of the packet can be delivered correctly. (A router normally makes its forwarding decisions based on the destination address.) With unicast RPF enabled, an inside attacker cannot send spoofed packets like done with the smurf6 attack and a remote victim-ip anymore because the router notes that this packet could not be generated in a correct manner behind that interface, (RFC 4942).

Block all extension headers that are not used or unknown and review that list on a regular basis to not inadvertently disable new useful extension headers.

Verify that the RFCs are implemented as accurate as possible, e.g., the PadN option within extension headers should only have zero-valued octets and should be blocked otherwise. This would minimize the risk for covert channels, (RFC 4942).

IDS/IPS: Flooded router alerts in Hop-by-Hop Options headers can be mitigated by implementing rate limits for them. The thresholds for these rate limits must be chosen carefully.

All IPv6 packets that contain a Routing header type 0 should be blocked. (A blocking of all Routing headers at once would prohibit the use of Mobile IPv6 and its Routing headers type 2. Therefore, only RH0 packets should be blocked. If MIPv6 is not used, all Routing headers can be blocked.)

A firewall should be able to reassemble fragmented packets in order to investigate the upper layer information of the initial IPv6 packet. It should then enforce its security policy.

V. Conclusion

In this paper we explained the new IPv6 protocol and its security vulnerabilities. We showed the different scopes of security issues which we categorized in attacks against the protocol itself. A comparison of both internet protocols with respect to their level of security shows that they provide almost the same (in-) security, though some attack vectors have changed. There are not many attacks with a devastating effect that can be executed from a remote IPv6 network. But as in IPv4 networks, an IPv6 network loses its security completely if an attacker gains access to the local area network or is an inside attacker. We further showed that vulnerabilities can arise from insufficient implementations and covered the problems that are relevant during the transition from IPv4 to IPv6. For network administrators and security specialists it is quite important to know about IPv6 related security vulnerabilities, to test the used security equipment, and to update all running software's on the appropriate machines if firmware updates are provided by the vendors. As history has shown, new security vulnerabilities and attacks will arise every time and new products will still have implementation issues and have to work further to solve these issues.

References

[1] Request for Comments: 2460, Internet Protocol Version 6 (IPv6) Specification, December 1998: Internet Engineering Task Force.
[2] Gordon Fyodor Lyon. Nmap - Network Mapper, 2012.
[3] Marc Van Hauser Heuse. The IPv6 Attack Toolkit. 2012.
[4] Serdar Cabuk, Carla E. Brodley, and Clay Shields. IP Covert Channel Detection. ACM Trans. Inf. Syst. Secure 12(4):22:1{22:29, April 2009.
[5] Matt Bishop, Security: Art and Science. Addison Wesley Professional, 2003.
[6] Philippe Biondi. Scapy 2011.
[7] Van Jacobson, Craig Leres, and Steven McCanne. Tcpdump - command-line packet analyzer. 2012.
[8] Scott Hogg and Eric Vyncke. IPv6 Security. Cisco Press, 2009.
[9] Judy Novak. Target-Based Fragmentation Reassembly. 2005.
[10] Source fire. Snort - an open source network intrusion prevention and detection system (IDS/ IPS). 2012.
[11] Colleen Shannon, David Moore, and K Claffy. Characteristics of Fragmented IP Traffic on Internet Links. 2001.
[12] Philippe Biondi and Arnaud Ebalard. IPv6 Routing Header Security. 2007.
[13] S. Bellovin. Security Problems in the TCP/IP Protocol Suite. SIGCOMM Comput.Commun. Rev, 19(2):32-48, April 1989.
[14] Kamran Ahsan and Deepa Kundur. Practical Data Hiding in TCP/IP. In Proc. Workshop on Multimedia Security at ACM Multimedia '02, Juan Les Pins, France, 2002.
[15] Antonios Atlasis. Attacking IPv6 Implementation Using Fragmentation. 2012.
[16] Sean Convery and Darrin Miller. IPv6 and IPv4 Threat Comparison and Best-Practice Evaluation, (v1.0). 2004.