The Impact of Extension Headers on IPv6 Access Control Lists Real-Life Use Cases

Antonios Atlasis

aatlasis@secfu.net

Abstract

Traffic filtering using Access Control Lists (ACLs) is a fundamental functionality not only of security devices (like firewalls) but also of networking ones, like routers and layer-3 switches. Enterprise networks, from Internet Service Providers to medium business ones enhance their defence in-depth strategy by employing ACLs in such devices. Unwanted traffic is prevented from routing by using the appropriate ACLs, while the access of management or other interfaces and services are protected using ACLs too. Although this used to be an effective approach in the IPv4 era, this is not the case in IPv6 (by the way, did you know that more than 75% of the transit Autonomous Systems in Western Europe advertise IPv6 prefixes as of July 2015, with some countries reaching even 90%?).

While a decent amount of research have been performed the last few years concerning IPv6 security implications focusing mainly on local area networks, this is not the case regarding its impact on infrastructure (core) IP networks (for instance, if and how BGP is affected by IPv6 new functionalities?). This talk will fill this gap by demonstrating live how the ACLs of enterprise layer-3 devices of different major vendors (Cisco and Hewlett Packard to name a few) can be circumvented by exploiting IPv6 protocol functionalities using publicly available tools. It should be noted that this is not a vendors' issue, but rather a protocol design one. Due to this reason, even additional security measures taken by some vendors are finally proven to be inefficient.

Finally, the analysis of the root cause of the problem will allow us to reach our final goal: to propose very specific mitigation techniques, both in terms of device implementation (so as to protect our networks in short-term), but also regarding the philosophy of the Internet protocol itself and how this should be changed in the long run.

Keywords: Access Control Lists, Routers, IPv6 Extension Headers, infrastructure networks.

The Impact of Extension Headers on IPv6 Access Control Lists Real-Life Use Cases

Antonios Atlasis

aatlasis@secfu.net

1 Introduction

Routers are definitely the most significant internetworking devices in today's IP networks and Access Control Lists (ACLs) a core functionality of them. ACLs refer to a list of rules applied to IP addresses and networks, protocols (mainly layer-4 ones) and to some of their fields (e.g. port numbers). They can generally be configured to control either inbound or outbound traffic, or both, and in this context they are similar to firewalls. As such, they are subject to security regulations and standards.

Enterprise networks, from Internet Service Providers to medium business ones enhance their defence in-depth strategy by employing ACLs in such devices. Unwanted traffic is prevented from routing by using the appropriate ACLs, while the access of management or other interfaces and services are protected using ACLs too. Although this used to be an effective approach in the IPv4 era, this is not the case in IPv6

Layer-3 core networks have been changing the last few years due to the operational deployment of IPv6 from several Internet Service Providers (ISPs) worldwide. According to Cisco Labs measurements, as of July of 2015, the IPv6 transit Autonomous Systems[1] (AS) are more than 75% in Western Europe, with some countries reaching even 90%[2] (figure 1). These measurements are quite similar regarding other countries, especially North American ones.

1 IPv6 Transit AS: An Autonomous System (AS) that is Transit (gives Transit service to another AS) on both IPv6 and IPv4 networks.

2 http://6lab.cisco.com/stats/

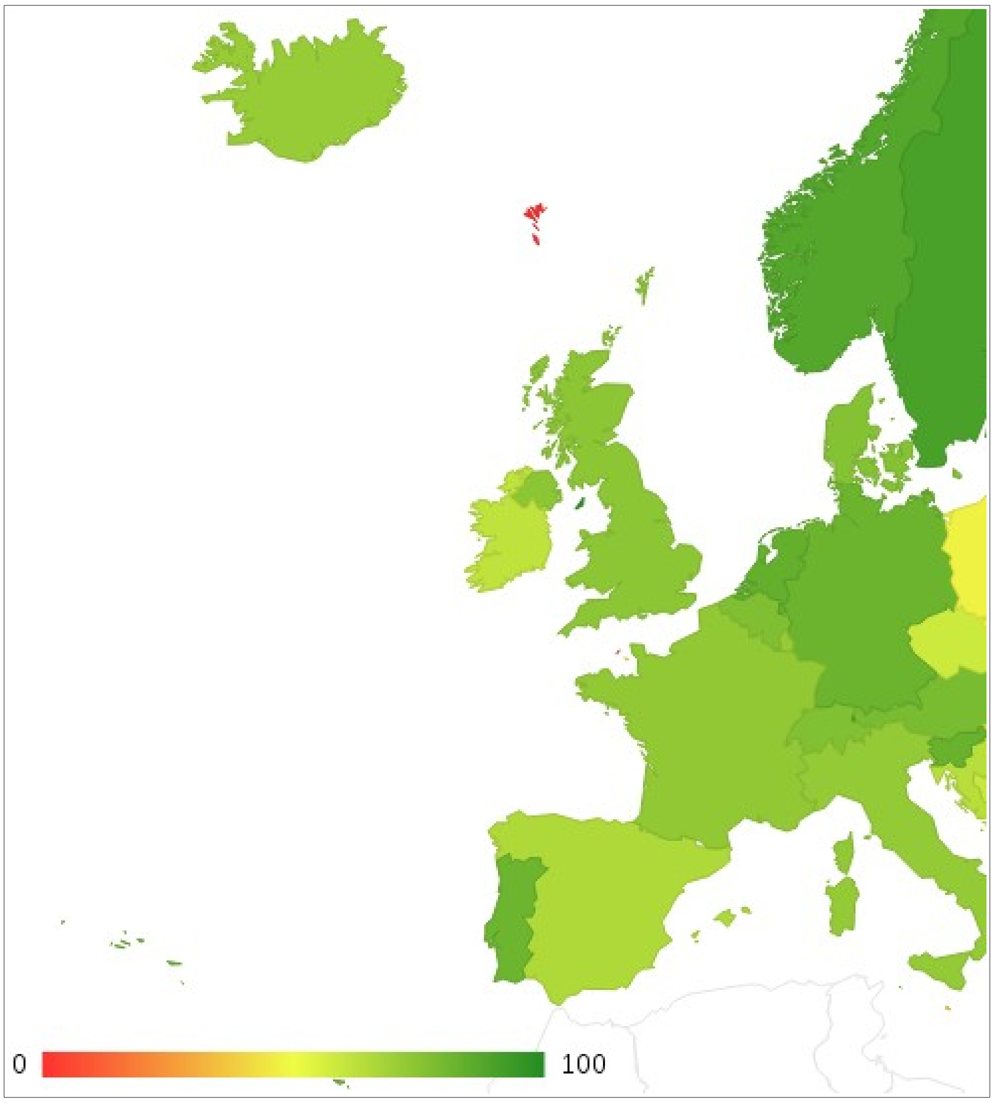


Figure 1: IPv6 Transit Autonomous Systems in Western Europe.

While a decent amount of research have been performed the last few years concerning IPv6 security implications, mainly on local area networks, this is not the case regarding its impact on backbone (core) IP networks (for instance, how BGP would be affected if ACLs become ineffective?). Hence, examining the effectiveness of the ACLs when IPv6 is used is crucial for today's core networks. The assumption that it should be the same as under IPv4 is rather naive given the new layer-3 functionalities that IPv6 introduces, as for example the so called IPv6 Extension headers.

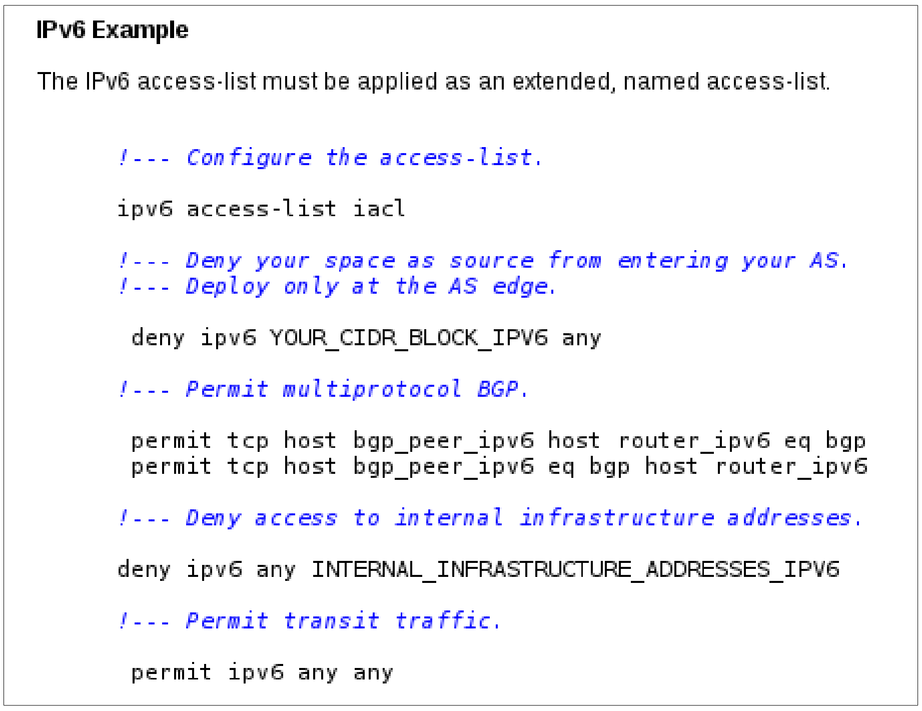
This study will fill this gap by demonstrating how the ACLs of enterprise layer-3 devices of different major vendors can be circumvented by exploiting IPv6 protocol functionalities using publicly available tools. It should be noted that this is not a vendors' issue, but rather a protocol design one. Due to this reason, even additional security measures taken by some vendors are finally proven to be inefficient.

By analysing the root cause of the problem we will be able to propose very specific mitigation techniques, both in terms of device implementation (so as to protect our networks in short-term), but also regarding the philosophy of the Internet protocol itself and how this should be changed in the long run.

2 The Significance of ACLs in Infrastructure Networks

To underline the importance of ACLs in the protection of infrastructure devices and networks, let's check the recommendations made publicly by one of the major vendors, Cisco. Specifically, Cisco suggest that "*in an effort to protect routers from various risks—both accidental and malicious—infrastructure protection ACLs should be deployed at network ingress points. These IPv4 and IPv6 ACLs deny access from external sources to all infrastructure addresses, such as router interfaces. At the same time, the ACLs permit routine transit traffic...*" [4].

But let's do it more specfic. At the same web site, specific ACL examples are suggested both for IPv4 and IPv6 traffic. Let's have a look at the IPv6 ACL example (figure 2).


IPv6 Example

The IPv6 access-list must be applied as an extended, named access-list.

```
      !--- Configure the access-list.

      ipv6 access-list iacl

      !--- Deny your space as source from entering your AS.
      !--- Deploy only at the AS edge.

       deny ipv6 YOUR_CIDR_BLOCK_IPV6 any

      !--- Permit multiprotocol BGP.

       permit tcp host bgp_peer_ipv6 host router_ipv6 eq bgp
       permit tcp host bgp_peer_ipv6 eq bgp host router_ipv6

      !--- Deny access to internal infrastructure addresses.

      deny ipv6 any INTERNAL_INFRASTRUCTURE_ADDRESSES_IPV6

      !--- Permit transit traffic.

       permit ipv6 any any
```

Figure 2: An ACL configuration example of an infrastructure router is recommended in [4]

The significant observation on the above list is that the final rule has to be a default allow one. As it is explained in [4], "*the final line in the infrastructure ACL explicitly permits transit traffic:* ***permit ip any any*** *for IPv4 and* ***permit ipv6 any any*** *for IPv6. This entry ensures that all IP protocols are permitted through the core and that customers can continue to run applications without issues*". This is the main difference of ACLs between routers (or layer-3 switches) and firewalls, where the final rule should be a "default deny).

It is this "default allow" rule in the ACLs of the routers which, allow us, in combination with other issues, to circumvent such ACLs.

3 IPv6 Extension Headers

It is beyond the scope of this paper to describe IPv6 Extension headers; for a very good summary of them the reader is advised to read ref. [1], and for more details, RFC 2460 [2]. For reasons of completeness of this paper, a very brief introduction of them is given below.

A generic structure of an IPv6 datagram is displayed in figure 3. Specifically, each IPv6 datagram can have zero, one or more IPv6 Extension headers.

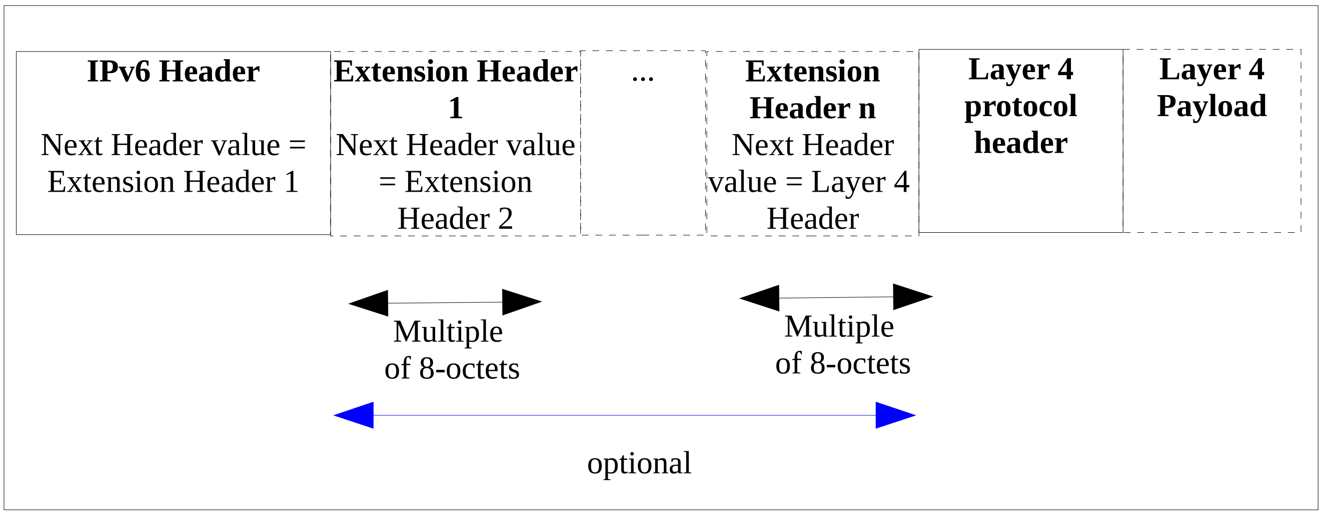


Figure 3: Structure of an IPv6 datagram

Some of the most common of the IPv6 Extension headers are the Hop-by-Hop Extension header (which MUST be processed by intermediate routers), the Destination Options header (which can occur twice), the Routing header, etc.

In case of fragmentation (which is also accomplished using an Extension header, the so called IPv6 Fragment Extension header), each fragment can carry zero, one or more IPv6 Extension headers in its unfragmentable part, in its fragmentable part or in both (figure 4).

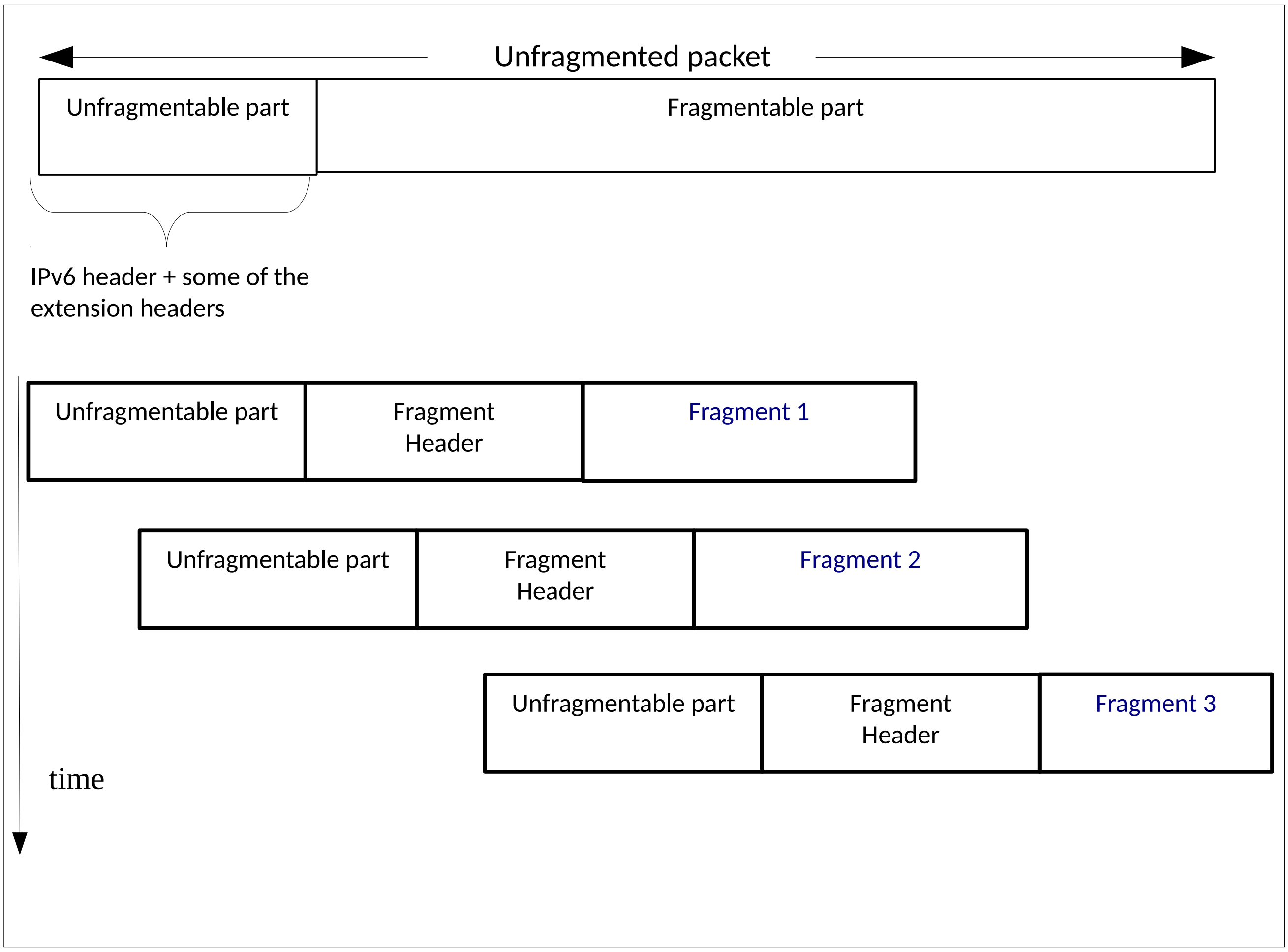


Figure 4: An example of an IPv6 Fragmentation

The combination of fragmentation and the usage of IPv6 Extension headers in the fragmentable part of the fragments can result in "moving" the layer-4 protocol (e.g. TCP) to a fragment other than the 1st one. This is one of the key issues which can be exploited to evade ACLs.

4 Performed Tests and Results

4.1 Lab Set-Up and Tools

We run several tests against different devices of several vendors, both high-end and low-end ones. To this end, several representative scenarios from enterprise environments or other potential ones were examined. We are going to present our results for three such vendors, because they are probably some of the most well-known ones of such devices:

a) Cisco.

b) Hewlett-Packard.

c) Alcatel.

What we want to underline through is the "nature" of the problem and not to make it to specific to a bunch of vendors. This will help the community (and hopefully, the IETF) to focus on the problem and try to solve it properly.

Without loss of generality, a set-up with one router is displayed. However, the results that will be demonstrated are valid in the common case where two or more results are present.

The lab set-up is displayed below:

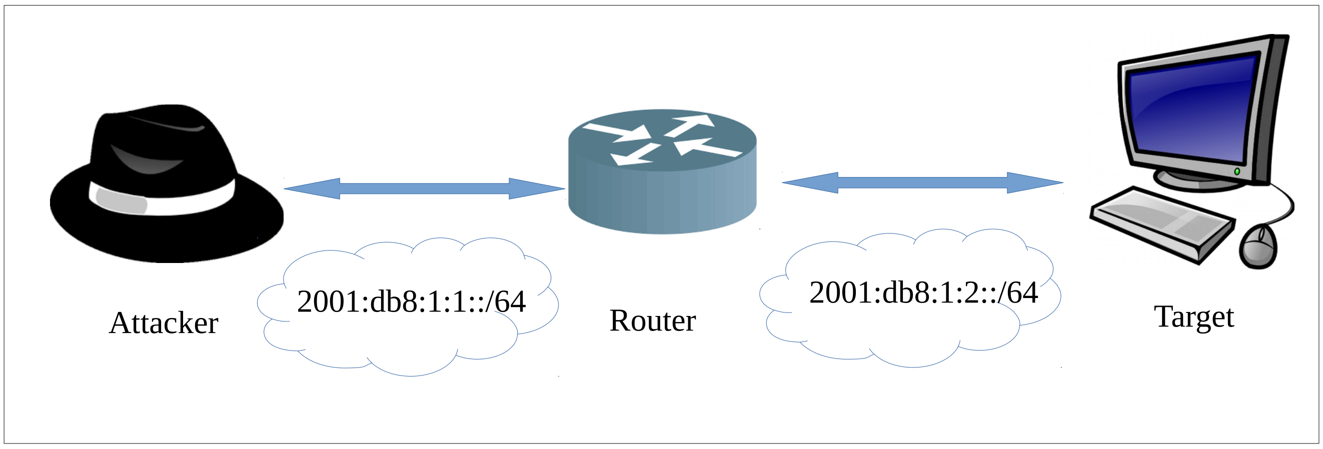


Figure 5: The lab diagram

The tool used to launch the attacks is *Chiron* [3], a public available open-source one based on Python and Scapy offering several advanced IPv6-related functionalities.

In the rest of this section, some specific test cases will be examined and discussed.

4.2 Case Study 1: Cisco

As far as Cisco router is concerned, its exact version used in our examples is displayed in the below “show version” output:

Router>show version

Cisco IOS Software, C1900 Software (C1900-UNIVERSALK9-M), Version 15.4(3)M, REL)
Cisco CISCO1921/K9 (revision 1.0) with 491520K/32768K bytes of memory.

The testing scenarios that were examined are the following:

4.2.1 Use-Case A: SSH is Blocked and Any Other IPv6 Connection is Allowed (Default Allow)

Based of the common case that the configuration of routers (either infrastructure or not) regarding ACLs, as explained in Section 2, is based on a default allow rule, in this case we block a specific service (as for example SSH) and we allow any other IPv6 connection. This is a common scenario in enterprise environments where we want to restrict access to sensitive services running to the device itself or behind a device, like SSH, while all the other traffic is allowed. An output of the

ACL configuration is given below.

Router#show ipv6 access-list

```
IPv6 access list protect_infrastructure
    deny tcp any any eq 22 sequence 10
    permit ipv6 any any sequence 20
```

In this case, the attacker's goal is to reach the target's SSH port as well as the SSH port of the router itself. This can be achieved by using the following generic way:

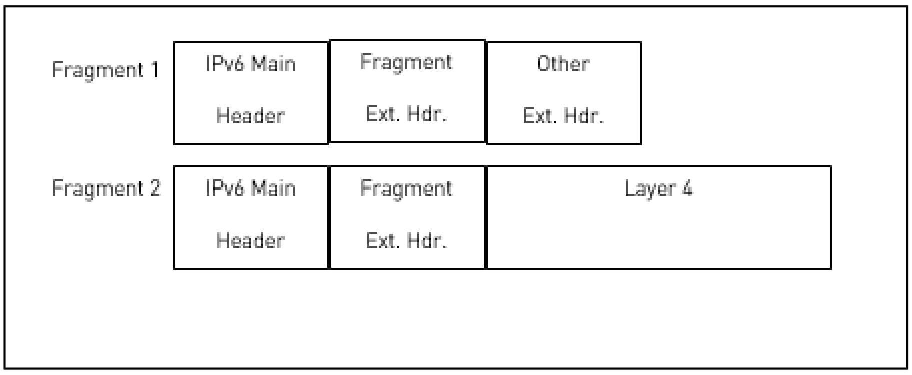


Figure 6*:* An example of an ACL Evasion technique

In the above figure, the "Other Extension Header" can be:

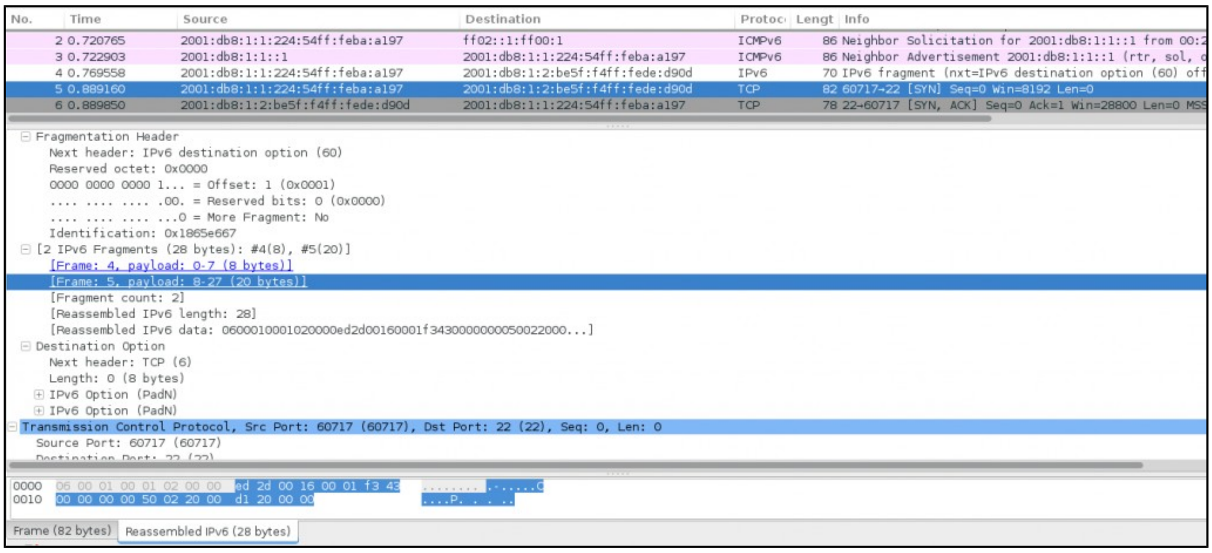
1. When the target is an Operating System like Linux, a Destination Options header, a Hop-by-Hop header, a Routing header (even a Type-0 one), or another Fragment Extension header (with offset=0 and the M-bit unset).
2. When the target is the router itself, all of the above but the Hop-by-Hop header. This is due to the fact that according to RFC 2460 the Hop-by-hop header MUST immediately follow the IPv6 main header, which is not the case here; it seems that Cisco IOS respects this recommendation and does not process packets when the Hop-by-hop header does not immediately follow the IPv6 main header. However, all the rest (Routing header, Destination Options header, or even Fragment header) can be used successfully.

One of the possible *Chiron* commands that can be used to replicate such an attack, is the following:

```
./chiron_scanner.py enp7s0[3] -d 2001:db8:1:2:be5f:f4ff:fede:d90d -gw 2001:db8:1:1::1 -sS -p 22
-lfE 60 -nf 2
```

A screenshot of a Wireshark capture that demonstrates the aforementioned attack is displayed below.

3 *enp7s0* is the network interface we use in our system.



```
No.   Time       Source                          Destination                     Protoc Lengt Info
    2 0.720765   2001:db8:1:1:224:54ff:feba:a197 ff02::1:ff00:1                  ICMPv6    86 Neighbor Solicitation for 2001:db8:1:1::1 from 00:2
    3 0.722903   2001:db8:1:1::1                 2001:db8:1:1:224:54ff:feba:a197 ICMPv6    86 Neighbor Advertisement 2001:db8:1:1::1 (rtr, sol, o
    4 0.769558   2001:db8:1:1:224:54ff:feba:a197 2001:db8:1:2:be5f:f4ff:fede:d90d IPv6     70 IPv6 fragment (nxt=IPv6 destination option (60) off
    5 0.889160   2001:db8:1:1:224:54ff:feba:a197 2001:db8:1:2:be5f:f4ff:fede:d90d TCP      82 60717→22 [SYN] Seq=0 Win=8192 Len=0
    6 0.889850   2001:db8:1:2:be5f:f4ff:fede:d90d 2001:db8:1:1:224:54ff:feba:a197 TCP      78 22→60717 [SYN, ACK] Seq=0 Ack=1 Win=28800 Len=0 MSS

⊟ Fragmentation Header
    Next header: IPv6 destination option (60)
    Reserved octet: 0x0000
    0000 0000 0000 1... = Offset: 1 (0x0001)
    .... .... .... .00. = Reserved bits: 0 (0x0000)
    .... .... .... ...0 = More Fragment: No
    Identification: 0x1865e667
⊟ [2 IPv6 Fragments (28 bytes): #4(8), #5(20)]
    [Frame: 4, payload: 0-7 (8 bytes)]
    [Frame: 5, payload: 8-27 (20 bytes)]
    [Fragment count: 2]
    [Reassembled IPv6 length: 28]
    [Reassembled IPv6 data: 0600010001020000ed2d00160001f343000000000050022000...]
⊟ Destination Option
    Next header: TCP (6)
    Length: 0 (8 bytes)
  ⊞ IPv6 Option (PadN)
  ⊞ IPv6 Option (PadN)
⊟ Transmission Control Protocol, Src Port: 60717 (60717), Dst Port: 22 (22), Seq: 0, Len: 0
    Source Port: 60717 (60717)
    Destination Port: 22 (22)

0000  06 00 01 00 01 02 00 00  ed 2d 00 16 00 01 f3 43   ........ .-.....C
0010  00 00 00 00 50 02 20 00  d1 20 00 00               ....P. . . ..

Frame (82 bytes)   Reassembled IPv6 (28 bytes)
```

Figure 7: A Wireshark screenshot of the aforementioned attack

4.2.2 Use-Case B: A Hop-by-Hop Header is Allowed and a Default Deny Rule is Used

In this case we assume that there is a default deny rule but packets with a Hop-by-Hop header are allowed. An example of such an ACL is displayed below.

IPv6 access list myrule2

permit hbh any any (1 match) sequence 10
deny tcp any any eq 22 (1 match) sequence 20

The goal of the attacker is to reach any service (like SSH) which is nevertheless blocked by the default deny rule. This can be used by:

1. Adding a Hop-by-Hop header and leaving the IPv6 datagram unfragmented.

The corresponding Chiron command is the following:

./chiron_scanner.py enp7s0 -d 2001:db8:1:2:5a55:caff:fe24:933d -gw 2001:db8:1:1::1 -sS -p 22 -lfE 0

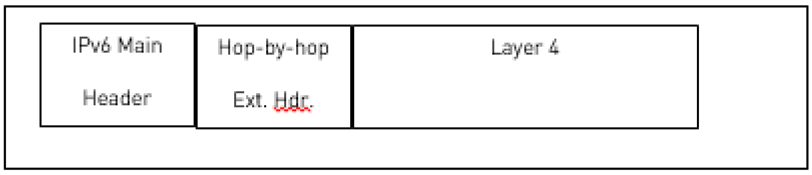


***Figure 8**:* Adding a Hop-by-Hop header and leaving the IPv6 datagram unfragmented.

2. Adding a Hop-by-Hop header and splitting the IPv6 datagram in two (2) fragments.

The corresponding Chiron command is the following:

./chiron_scanner.py enp7s0 -d 2001:db8:1:2:5a55:caff:fe24:933d -gw 2001:db8:1:1::1 -sS -p 22 -lfE 0 -nf 2

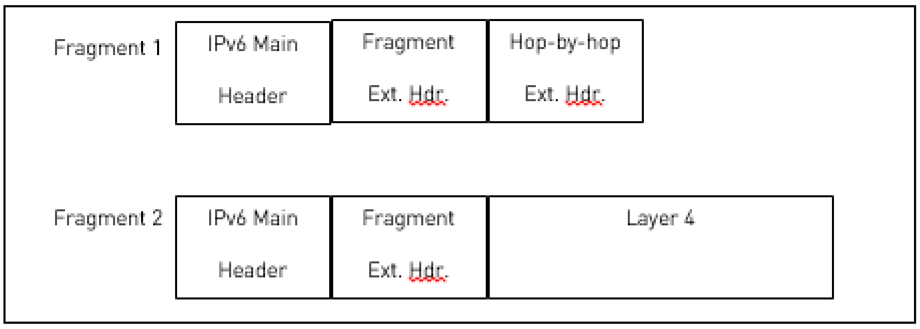


***Figure 9**:* Adding a Hop-by-Hop header and splitting the IPv6 datagram in two (2) fragments.

4.2.3 Use-Case C: Permit a Specific Service Explicitly Combined with an Extension Header and Use a Default Deny Rule

In this case, if we want to allow a specific service (as for example WWW) as well as the use of an Extension header like Hop-by-Hop, we define such a ruleset that tights the use of this extension header with the specific service. Example:

IPv6 access list myrule4

```
  permit tcp any any eq www sequence 10
  permit tcp any any eq www hbh sequence 20
```

In the above example, a) we allow the use of www and b), we also allow the use of Hop-by-Hop Extension header (hbh) with www, but without allowing hbh on each own. The goal of the attacker is again to reach a service, like ssh, which otherwise is forbidden.

The above approach seems to work quite well (it cannot be evaded), but it creates a few problems.

First, these combinations must be repeated for all the services that we want to allow, as well as for all the corresponding Extension headers. This makes the management of the ACL more difficult than usually.

Secondly, it creates some false alarms. For instance, if we add a Destination Options Header and fragment it in two fragments, these are blocked even when we try to reach the www service. The corresponding Chiron command for such a case is the following:

./chiron_scanner.py enp7s0 -d 2001:db8:1:2:5a55:caff:fe24:933d -gw 2001:db8:1:1::1 -sS -p 80 -luE 0 -lfE 60 -nf 2

The same is also true if we just fragment an IPv6 datagram which includes a Hop-by-Ho header:

./chiron_scanner.py enp7s0 -d 2001:db8:1:2:5a55:caff:fe24:933d -gw 2001:db8:1:1::1 -sS -p 80 -lfE 0 -nf 2

The packets created by the above command are fully legitimate and hence, they should not be dropped.

4.3 Case Study 2: Hewlett Packard

Similar tests were performed against Hewlett Packard layer-3 devices. As an example, an HP A5800 JC100A layer-3 switch is used.

4.3.1 Use-Case A: SSH is Blocked and Any Other IPv6 Connection is Allowed (Default Allow)

This is similar to use-case A of case study 1. Specifically, due to the nature of switches, the default allow policy should be expected. Hence, we will test the scenario were we want to block a specific service (e.g. ssh) from specific port(s). The tested configuration is the following:

```
[HP]display packet-filter all
  Interface: GigabitEthernet1/0/1
  In-bound Policy:
    acl6 3002, Successful
  Out-bound Policy:
  Interface: GigabitEthernet1/0/2
  In-bound Policy:
    acl6 3001, Successful
  Out-bound Policy:

[HP]display acl ipv6 all
 Advanced IPv6 ACL  3001, named -none-, 1 rule,
 ACL's step is 5
 rule 0 deny icmpv6 icmp6-type router-advertisement
 Advanced IPv6 ACL  3002, named -none-, 1 rule,
 ACL's step is 5
 rule 0 deny tcp destination-port eq 22
```

In our tests, after confirming that ssh is really blocked in normal IPv6 traffic, we used *Chiron* to add some IPv6 Extension Headers. It was found out that by adding at least three (3) Extension headers, the ACL is evaded (TCP SYN SSH packet reaches the target). In this scenario there was no need for fragmentation. Three IPv6 Extension headers were more than enough to circumvent the ACL.

4.3.2 Use-Case B: A Hop-by-Hop Header is Allowed and a Default Deny Rule is Used

This is similar to use-case B of case study 1. The configuration of the HP layer-3 switch used for these tests is as follows:

```
[HP]display packet-filter all
  Interface: GigabitEthernet1/0/1
  In-bound Policy:
    acl6 3003, Successful
  Out-bound Policy:
  Interface: GigabitEthernet1/0/2
  In-bound Policy:
    acl6 3001, Successful
  Out-bound Policy:
```

```
[HP]display acl ipv6 all
 Advanced IPv6 ACL  3001, named -none-, 1 rule,
 ACL's step is 5
 rule 0 deny icmpv6 icmp6-type router-advertisement
 Advanced IPv6 ACL  3002, named -none-, 5 rules,
 ACL's step is 5
 rule 0 deny tcp destination-port eq 22
 rule 1 deny 60
 rule 2 deny 43
 rule 3 deny 0
 rule 4 deny 44
 Advanced IPv6 ACL  3003, named -none-, 3 rules,
 ACL's step is 5
 rule 0 permit tcp destination-port eq 22
 rule 1 permit 0
 rule 10 deny ipv6
```

Using the above rules, ssh and Hop-by-Hop Extension Headers are allowed, whilst all the other traffic is blocked (default deny).

Once more, the results showed that this ACL can easily be evaded by adding just a Hop-by-Hop Extension header, as expected.

4.4 Case Study 3: Alcatel

Similar tests were performed against Alcatel routers. Let's examine some of the use-cases.

4.4.1 Use-Case A: SSH is Blocked and Any Other IPv6 Connection is Allowed (Default Allow)

Again, in this scenario the Access Control List (ACL) of therouter was configured to block some very specific traffic (ssh, for instance), while all the rest of the traffic was allowed. The goal was to pass the specifically forbidden traffic and reach the target. The ACL configuration is as follows:

```
A:Alcatel>config>filter# info
----------------------------------------------
        ipv6-filter 3 create
            default-action forward
            description "Block SSH via IPv6"
            scope exclusive
            entry 10 create
                match next-header tcp
                    dst-ip 2001:db8:1:1::/64
                    dst-port eq 22
                    tcp-syn true
                exit
                action drop
            exit
        exit
```

Against such a configuration, at least the following two different evasion techniques were found to be effective:

1. Adding six (6) IPv6 Extension Headers (e.g. Destination Option Headers) in an unfragmented IPv6 datagram. Note: Such packets are accepted by various OS, despite the fact that according to RFC 2460 [2] the recommended number of Destination Options headers is two (2). Nevertheless, we could also use different types of IPv6 Extension Headers to fully comply with RFC 2460.
2. Add only one (1) Extension Header (e.g. a Destination Option Header) and split the datagram in two fragments.

4.4.2 Use-Case B: A Hop-by-Hop Header is Allowed and a Default Deny Rule is Used

This is the same use use-cases B of the previous cases studies. In this scenario we assume that an ISP must support a Hop-by-Hop Extension Header, which nevertheless, when present, must be processed by all nodes along the route to the final destination. On the other hand, we block all the rest (default:deny).

The following ACL configuration was used:

```
A:Alcatel>config>filter# info
----------------------------------------------
    ipv6-filter 3 create
        description "Allow fragmentation and icmpv6 - Block the rest"
        scope exclusive
        entry 7 create
            match
                hop-by-hop-opt true
            exit
            action forward
        exit
        entry 9 create
            match next-header ipv6-icmp
            exit
            action forward
        exit
    exit
```

Notes:
1. When a default-action is not explicitly defined, this is a deny one.

2. The entry no 9 (which allows ICMPv6) was added to allow the Neigbor Discovery (ND) process to take place. This did not affect our testing because ICMPv6 was not employed by any means on them.

The following evasion technique was proven to be effective under this scenario.

A specific TCP port number at the target can be reached (e.g. TCP port 22 – ssh) if a Hop-by-Hop Extension Header is added to the datagram (without performing fragmentation).

This approach is also so generic that we do not actually need any additional technique.

Furthermore, this technique can also be expanded using any other IPv6 Extension header or a combination of them, added in the allowed list of the corresponding ACL.

4.4.3 Use-Case C: Allow Fragmentation and Block All the Rest

In this scenario we assumed that an ISP must support and provide fragmentation capabilities to its customers. So, we allow fragmentation and we block all the rest (default:deny).

The following ACL configuration is used:

```
A:Alcatel>config>filter# info
----------------------------------------------
    ipv6-filter 3 create
        description "Allow fragmentation and icmpv6 - Block the rest"
        scope exclusive
        entry 8 create
            match
                fragment true
            exit
            action forward
        exit
        entry 9 create
            match next-header ipv6-icmp
            exit
            action forward
        exit
    exit
```

Notes:

1. When a default-action is not explicitly defined, this is a deny one.
2. The entry no 9 (which allows ICMPv6) was added to allow the Neigbor Discovery (ND) process to take place. This did not affect our testing because ICMPv6 is not “exploited” by any means during our tests.

The following evasion technique were proven to be effective under this scenario.

Any TCP port number at the target can be reached (e.g. TCP port 22 – ssh) if the datagram is simply split in two fragments (without adding any Extension Header). *Of course, this technique can also be used against other ports or protocols which are explicitly blocked (e.g. TCP port 80, etc.). This approach is so generic that we do not actually need any additional technique.*

4.4.4 *Use-Case D:* Blocking No Next Headers

This scenario is actually the same with use-caseA with the exception that we also add the following

ACL rule:

```
match next-header ipv6-no-nxt
```

The rest is the same with the rules described in use-case A. Hence, in this case the used ACL is configured as following:

```
ipv6-filter 4 create
    default-action forward
    description "Block SSH - Allow the Rest"
    scope exclusive
    entry 10 create
        match next-header tcp
            dst-ip 2001:db8:1:1::/64
            dst-port eq 22
            tcp-syn true
        exit
        action drop
    exit
    entry 11 create
        match next-header ipv6-no-nxt
        exit
        action drop
    exit
exit
```

The results of this scenasrio are exactly the same with the ones obtained in use-case A. This means that we can easily evade the aforementioned ACL by using:

1. An unfragmented datagram with six (6) or more IPv6 Extension headers in each packet, or
2. Adding just one (1) IPv6 Extension header and splitting the datagram (including this Extension header) in two (2) or more fragments.

4.4.5 Use-Case E: Blocking No Next Headers and Using Cpm Hw Filters

In this scenario we <u>also</u> add the following ACL:

```
*A:Alcatel>config>sys>sec>cpm>ipv6-filter# info
----------------------------------------------
          entry 1 create
              action drop
              description "drop ssh"
              match next-header tcp
                  tcp-syn true
              exit
          exit
          entry 2 create
              action drop
              match next-header ipv6-no-nxt
              exit
          exit
```

```
        no shutdown
```

The results showed, once again, that the device could also be evaded by using exactly the same techniques.

5 Root Cause of the Problem and Consequences

The root cause of the problem is the combination of the facts that:

a) The insertion of IPv6 Extension headers, which takes place before the layer-4 one, moves backward the later. We have encountered cases where for instance three or more Extension headers in a row are more than enough to evade ACLs even without using fragmentation.

b) When one or more IPv6 Extension headers are fragmented, layer-4 header can be moved to a fragment other than the first.

c) Routers (and layer-3 switches) are devices which, due to their role, cannot perform deep packet inspection. They are meant not to store traffic but forward it as fast as possible; to make matter worse, speaking from the RFC point view, they are not supposed to examine any Extension header but the Hop-by-Hop one (if present) which, nevertheless, must not be fragmented.

The consequences of the above are rather obvious. ACL evasion is possible under some scenarios. From the ISP point of view is even more difficult because they are meant not to block any Extension headers (this is a decision that must be taken by the end users, usually they customers, either enterprise or home users). Hence, any protection mechanism based on ACLs is actually cancelled under IPv6. ACL circumvention may not lead to a device compromise itself, but it allows the launch of attacks against them which otherwise would be impossible (e.g. brute-forcing of an otherwise unreachable service like ssh). Similar security implications are also possible for core (infrastructure) protocols like BGP.

6 Mitigation Techniques

Although the configuration of Case C mentioned in Section 2 can be used to block effectively evasion attacks as long as it is done properly, its complexity, which increase significantly with the necessity of adding more rules, may make it not that a feasible solution. Hence, the best approach seems to be the blocking of IPv6 fragmentation.

6.1 RFC 7112 (Undetermined Transport) Implementation – Oh, really?

A way of mitigating the risk of fragmented Extension headers and moving the layer-4 header at the second fragment, officially suggested by Cisco themselves, is to use the “undetermined transport“ feature. This actually implements RFC 7112 [5] which actually recommends (but not strictly speaking) the blocking of fragments when layer-4 header is not in the first fragment. In this case, the corresponding ACL is modified as following:

```
IPv6 access list protect_infrastructure
deny ipv6 any any log undetermined-transport sequence 10
```

deny tcp any any eq 22 log sequence 20
permit ipv6 any any sequence 30

Such an ACL works effectively. It blocks access to ssh as well as any packets when layer-4 header is not included in the first fragment.

So, problem solved?

Well, not really. There's some additional remarks and lessons (to be learned) from our side here:

a) First of all it has to kept in mind that "undetermined-transport" has its own flaws too. In the Cisco "First Hop Security" Wiki[4] it's nicely stated: "*Some platforms may not support acl keyword "undetermined-transport". In that case they may either reject the command altogether, act erratically on such ACLs, or refuse to accept the ACL on the interface*" and we have seen this exact thing happening: We have experienced major problems in some high-end switches where even fully legitimate traffic is actually dropped after enabling undetermined-transport; this is what they mean by "act erratically"?

b) The potentially complex nature of IPv6 packets requires that stateless security controls have to be verified as for their actual security benefit and have to potentially "enhanced" by tweaks previously unknown in the IPv4 world (like "undetermined-transport").

c) Overall, this means that going for full IPv6 security is not about "feature parity" but about "identifying the right features/controls for the task", carefully taking IPv6 specifics into account.

Finally, there were cases, like the Hewlett-Packard case study, where fragmentation is not required to evade its ACLs. In such a case, even if RFC 7112 is successfully implemented, this is not enough to stop ACL circumvention of these devices.

6.2 Blocking Extension Headers Explicitly

One approach is to explicitly block all IPv6 Extension headers (or most of them, e.g. allowing just a Hop-by-hop header in the unfragmentable part may not hurt but this must be carefully tested depending on the environment and the specific implementation).

In the Cisco world, to block a specific Extension header, we can use a rule like the following:

deny 43 any any

This rule blocks packets that include a Routing header (next header value = 43). We should repeat similar rules for all known Extension headers, i.e. to block packets which include a Destination Options header, we should add:

deny 60 any any

and so on. However, the following rule is not enough to forbid fragmentation:

deny 44 any any

Although the next header value of the Fragment header is 44, this rule seems to be effective only when we encapsulate a Fragment Header in another fragment. To totally block IPv6 fragmentation

4 http://docwiki.cisco.com/wiki/FHS#.22undetermined-transport.22_keyword_support_on_various_platforms

in the Cisco world, we should add:

deny ipv6 any any fragments

This last rule blocks IPv6 fragmentation completely and even on each own, it should be more than enough to block most of the attacks against Cisco routers.

Similarly, in the Hewlett-Packard world, a configuration like the following can be used:

Given that NOT ALL IPv6 Extension Headers are available for blocking directly by HP devices, (just Hop-by-Hop, IPv6-Authentication Header, IPv6 Encapsulating Security Payload and fragmentation), which is NOT enough to mitigate the aforementioned risk, we will try to use protocol numbers to achieve it. For instance:

rule 1 deny 60	=> mitigates attack when 3 DestOpt are used, but not when one HbH and 2 DestOpt are used

Hence, to be effective, ALL IPv6 Extension Headers MUST be blocked explicitly by using the corresponding protocol numbers. An example is given below:

```
[HP]display acl ipv6 all
 Advanced IPv6 ACL  3002, named -none-, 5 rules,
 ACL's step is 5
 rule 0 deny tcp destination-port eq 22
 rule 1 deny 60
 rule 2 deny 43
 rule 3 deny 0
 rule 4 deny 44
...etc
```

The question is whether an ISP is allowed to do that and hence, remove all this IPv6 characteristics from its clients.

7 Conclusions

In this study it has been shown that IPv6 Extension headers and fragmentation, if abused “properly” by attackers, can be used to evade the Access Control Lists (ACLs) of networking devices, like routers and layer-3 switches, provided a certain configuration is present (which from our observations is a quite common case, if not unavoidable). ACLs are one of the measures taken in infrastructure routers for their protection as well as the protection of core networks and services and hence, their potential evasion can have a significant security impact on them. Mitigation techniques do exist but they are either not mature enough to offer the “silver bullet” against the problem, or they are just “quick and dirty” approaches. Thus, deep knowledge and careful testing and configuration is required per-case by the network administrators so as to enable IPv6 in their networks whilst protecting their assets from such attacks. Finally, it seems that some of the core characteristics of IPv6 must be reconsidered by IETF to make IPv6 more robust and less susceptible to related attacks. RFC 7112 is certainly a good step towards this direction but it was shown that it is not always enough, mainly due to implementation issues.

References

[1] Atlasis, A., Rey, E. and Schaefer, R., “Evasion of High-End IDPS Devices at the IPv6 Era”, BlackHat EU 2014, October 16-17, Amsterdam.

[2] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[3] A. Atlasis, “Chiron - An all-in-one IPv6 Pen Testing Framework”, downloaded from http://www.secfu.net/tools-scripts/

[4] Cisco, “Protecting Your Core: Infrastructure Protection Access Control Lists”, Document ID: 43920, retrieved from http://www.cisco.com/c/en/us/support/docs/ip/access-lists/43920-iacl.html in 12th July 2015.

[5] F. Gont, V. Manral, R. Bonica, “Implications of Oversized IPv6 Header Chains”, RFC 7112, January 2014.

[6], Cisco, “FHS: First Hope Security”, retrieved from http://docwiki.cisco.com/wiki/FHS in 12th July 2015.