Evasion of High-End IPS Devices in the Age of IPv6

Antonios Atlasis
secfu.net
aatlasis@secfu.net

Enno Rey
ERNW GmbH
erey@ernw.de

Abstract

IPv6 era is here, either if you already use it or if you continue to ignore it. However, even in the last case, this does not mean that your "nodes" (end-hosts, networking devices, security devices) are not already pre-configured with IPv6 connectivity, at least to some extent. At the same time, ARIN states that they are currently in phase three of a 4-phased "IPv4 Countdown Plan", being already down to about 0.9/8s in aggregate. On the other hand, RIPE NCC has reached its last /8 IPv4 address space quite some time ago.

And what IPv6 does not forgive for sure is the lack of security awareness. Several times in the past it has been shown that this "new" layer-3 protocol, apart from the huge address space and other new functionalities, it also brings with it several security issues. In this paper it will be shown that significant security issues still remain unsolved. Specifically, three different but novel techniques will be presented that allow attackers to exploit even a really minor detail in the design of the IPv6 protocol to make security devices like high-end commercial IDPS devices completely blind. These techniques allow the attackers to launch any kind of attack against their targets, from port scanning to SQLi, while remaining undetected. Moreover, in this paper, after presenting detailed analysis of the attacks and the corresponding exploitation results against IDPS devices, potential security implications to other security devices, like firewalls will also be examined. Finally, specific mitigation techniques will be proposed, both short-term and long-term ones, in order to protect your network from them.

1 Introduction

IPv6 deployment is rising every single day; Specifically, according to the statistics and the trends of the Internet Society, "*2013 marked the third straight year IPv6 use on the global Internet has doubled. If current trends continue, more than half of Internet users around the world will be IPv6-connected in less than 6 years.*" (Internet Society, 2014). So, "this time it is for real".

At the time of writing approx. 7% of the Internet users in the United States have IPv6 at their disposal (http://6lab.cisco.com/stats/cible.php?country=US), compared with ~ 2.4% 12 months ago, which indicates a consistent growth by a factor of three within one year. At the same time the *American Registry for Internet Numbers* (ARIN) states that they are currently in phase three of a 4-phased "IPv4 Countdown Plan" (https://www.arin.net/resources/request/ipv4_countdown.html). In

14th of May, it was already down to 0.87 /8s in aggregate. It can reasonably be expected that by the end of 2014 ARIN won't be able to serve customer/member requests for IPv4 address space any longer. The ARIN's European counterpart, the *Réseaux IP Européens Network Coordination Centre* (RIPE NCC) has reached its last /8 IPv4 address space quite some time ago (September 2012, http://www.ripe.net/internet-coordination/ipv4-exhaustion). Taking into account that the default behaviour of most operating systems is to prefer IPv6 over IPv4 for connection establishment once both are available, this means that a significant amount (10–20% according to most estimations) of connections to an organization's public services will be performed over IPv6 by the end of 2014. This in turn means that the security controls applied to/used for IPv4 network traffic should be able to provide a similar protection level to IPv6 network traffic, which – given the vast complexity of IPv6 (Rey, 2014) – might not be an easy task, to say the least.

However, even if you still do not use it or you plan to continue ignoring it, this does not mean that your "nodes" (hosts, networking devices, security devices) are not already pre-configured with IPv6 connectivity, at least to some extent. Nevertheless, sooner or later you will have to adopt it. And what IPv6 does not forgive for sure is the lack of security awareness.

In this paper, it will be shown that significant security issues still remain unsolved. Specifically, three novel techniques will be presented that can exploit a really minor detail in the design of the IPv6 protocol, which, however, allow attackers to launch any kind of attacks without being detected by making completely "blind" security devices like well-known high-end commercial IPS devices.

2 Problem Setting: Yet Another Mess with the IPv6 Extension Headers

It is well known that IPv6 brings several changes with it. One of them is that the IPv6 main has been simplified (by either dropping or moving some header fields or by making its size constant, etc.) in order to reduce the common-case processing cost of packet handling and to limit the bandwidth cost of the IPv6 header. However, aiming at offering improved support for Extensions and Options, it also brings changes in the way IP header options are encoded to allow "for more efficient forwarding, less stringent limits on the length of options, and greater flexibility for introducing new options in the future." (Deering & Hinden, 1998) Specifically, as it is explained in RFC 2460, "*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. There are a small number of such extension headers, each identified by a distinct Next Header value.*" According to the protocol specification, "*an IPv6 packet may carry zero, one, or more extension headers, each identified by the Next Header field of the preceding header.*"

According to the above explanation, a typical IPv6 datagram looks like the one displayed in Figure 1.

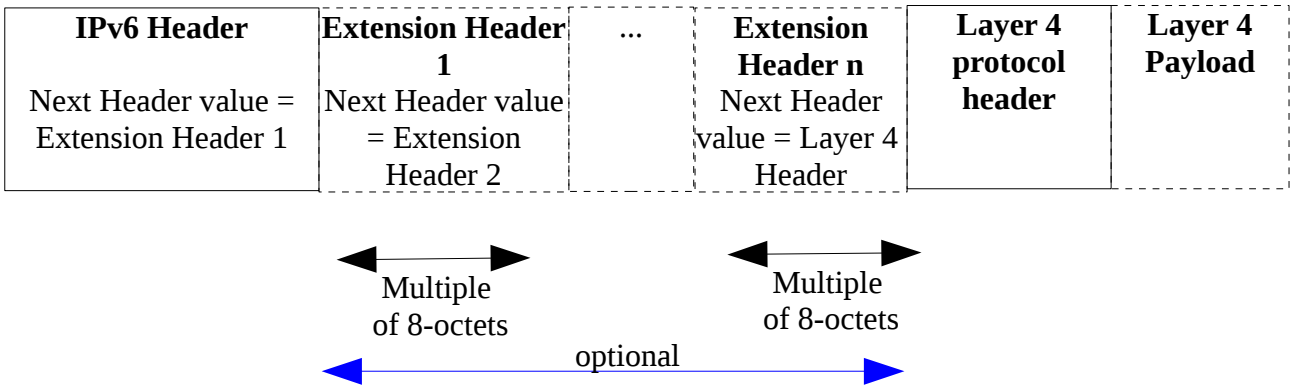


Figure 1*:* Structure of an IPv6 datagram

Several IPv6 Extension headers were introduced in (Deering & Hinden, 1998), while some more were added later. It is beyond the scope of this paper to mention or describe all of them, but the reader should have in mind that the security issues described later in this paper can actually affect most of them.

There have already been discovered several issues related with the handling of the IPv6 Extension headers (as for example, if we mix/change their order, or increase their number of occurrences arbitrarily, or both, etc); some of them were described in (Atlasis, 2012b) and in (Atlasis, 2013b) were the security impacts of abusing IPv6 Extension headers were demonstrated. In this paper, we will analyse a different issue which also has significant security implications: the improper handling of *Next Header* values during IPv6 Fragmentation. Moreover, we will also see that problems reported in (Atlasis, 2012b) still exist and that can be exploited in order to circumvent IDPS.

As already explained above, the type of each protocol header is identified by its preceding one by using the Next Header field. The Next Header field, as defined in (Deering & Hinden, 1998), is an 8-bit selector that identifies the type of header immediately following the IPv6 header, or the IPv6 Extension header that carries it. It uses the same values as the IPv4 Protocol field.

An an example, for the cases of figure 2, the Next Header values are defined as following:

IPv6 header	**TCP header + payload ...**
Next Header Value = 6 (TCP)	

Figure 2.a: Layer-4 header (TCP) immediately follows the IPv6 main header

IPv6 header	**IPv6 Routing Extension header**	**IPv6 Destination Options header**	**TCP header + payload ...**
Next Header Value = 43	Next Header Value = 60	Next Header Value = 6	

Figure 2.b: IPv6 main header is followed by an IPv6 Routing Extension Header and this one by the layer-4 header (TCP).

Figure 2: Typical examples of IPv6 datagrams

In the example of figure 2.a where the layer-4 header (TCP) immediately follows the IPv6 main header, the Next Header value of IPv6 header is set to 6, which is the protocol number of TCP.

In the example of figure 2.b, the IPv6 main header is followed by an IPv6 Routing Extension Header, this one by an IPv6 Destination Options header, and finally, by the layer-4 header (TCP). Thus, the next header value of IPv6 main header is set to 43, which is the protocol number of IPv6 Routing Extension headers; the Next Header value of the Routing header is set to 60, the protocol number of the Destination Options header, and this ones Next Header value is set to 6, the protocol number of TCP.

So far, so good. Everything seems simple and straightforward. If, for any reason, the Next Header value is not correct one (i.e. it does not identify correctly the header that follows), the recipient will simply be unable to recognise this next header and hence, it will discard the packet.

Now, things start to be complicated when fragmentation at the IP layer takes places. Generally speaking, fragmentation is an issue both in IPv4, as it was shown in (Ptacek & Newsham, 1998), as well as in IPv6 (Atlasis, 2012a), (Atlasis, 2013a). These cases were mainly related with layer-4 fragmentation overlapping though. However, as we shall see, this is not the only problem regarding fragmentation. In this paper, we shall examine how the combination of the incorrect usage of the Next Header values and legitimate fragmentation may be used to evade security devices like high-end Intrusion Prevention Systems (IPS).

In case of IPv6, fragmentation takes place as following (Deering & Hinden, 1998):

Let's assume that the initial IPv6 datagram that needs to be fragmented contains the IPv6 main header (obviously), a few IPv6 Extension headers and the layer-4 header (e.g. TCP) plus its payload. Some of the IPv6 Extension headers (like the Hop-by-Hop and the Routing IPv6 Extension headers) may need to be processed by intermediate nodes, while some other (e.g. Destination Options IPv6 Extension header) may not. A part of this initial IPv6 datagram will remain unfragmented, and the rest of it will be fragmented as following:

- The Unfragmentable Part will consist of the IPv6 main header plus any IPv6 Extension headers that must be processed by intermediate nodes en route to the destination (Hop-by-Hop and the Routing IPv6 Extension headers in our example).
- The Fragmentable Part will consist of the rest of the packet, that is, any IPv6 Extension headers that need be processed only by the final destination node(s), plus the upper-layer header and data (aka, Destination Options IPv6 Extension header, TCP header and its payload in our example).

Each fragment, as described in (Deering & Hinden, 1998), will be composed of:

1. "*The Unfragmentable Part of the original packet, with the Payload Length of the original IPv6 header changed to contain the length of this fragment packet only (excluding the length of the IPv6 header itself), and the Next Header field of the last header of the Unfragmentable Part changed to 44.*
2. *A Fragment header containing:*
    - *The Next Header value that identifies the first header of the Fragmentable Part of*

the original packet.

- *... <snipped for brevity>...*
- *The fragment itself."*

Based on the above, an example of fragmentation in IPv6 looks like in figure 3.

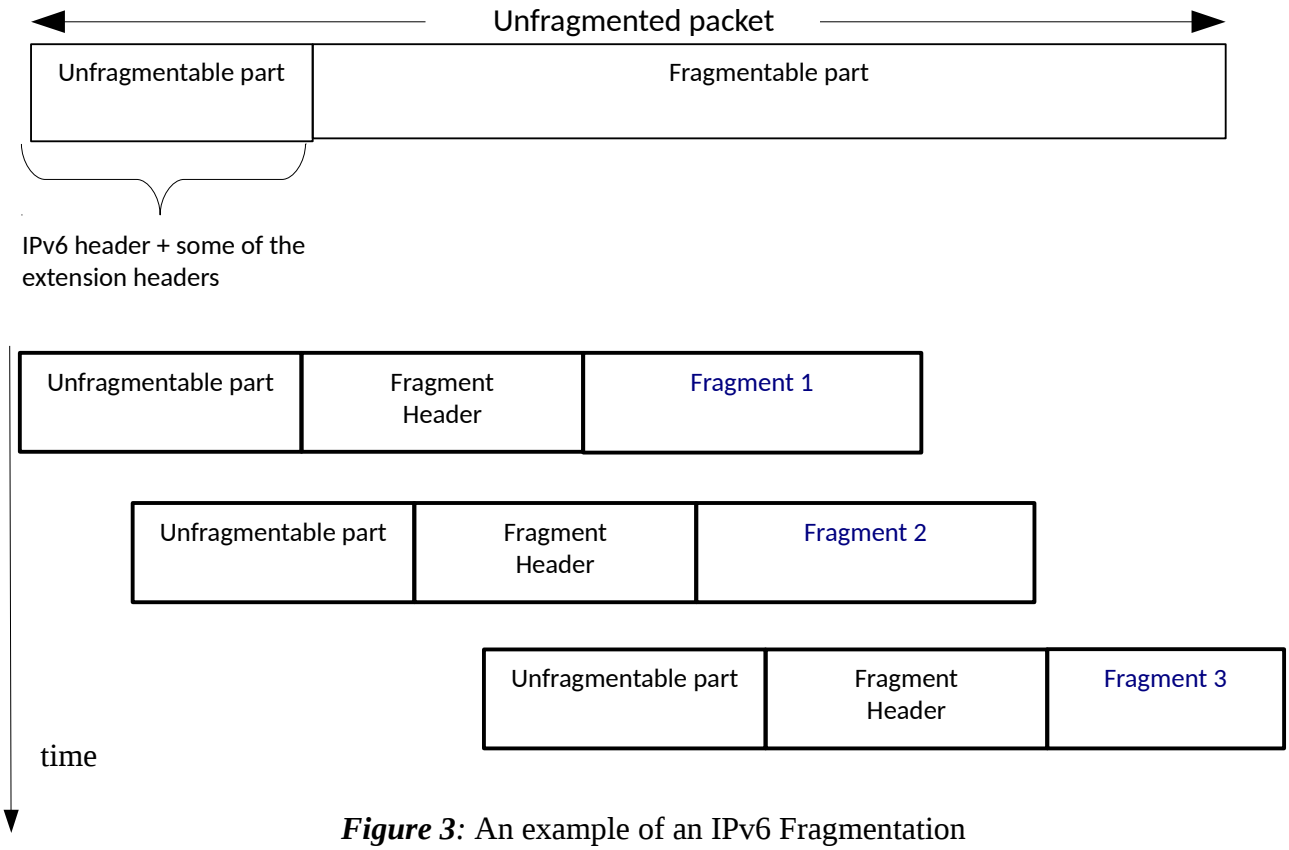


***Figure 3**:* An example of an IPv6 Fragmentation

As far as the reassembly of IPv6 fragmented datagrams is concerned, the following rules apply (Deering & Hinden, 1998):

"*The Unfragmentable Part of the reassembled packet consists of all headers up to, but not including, the Fragment header of the first fragment packet (that is, the packet whose Fragment Offset is zero), with the following change(s):*

- *The Next Header field of the last header of the Unfragmentable Part is obtained from the Next Header field of the **first** fragment's Fragment header. ..."*

And, to clarify things, according to this same RFC 2460:

"The following conditions are not expected to occur, but are not considered errors if they do:

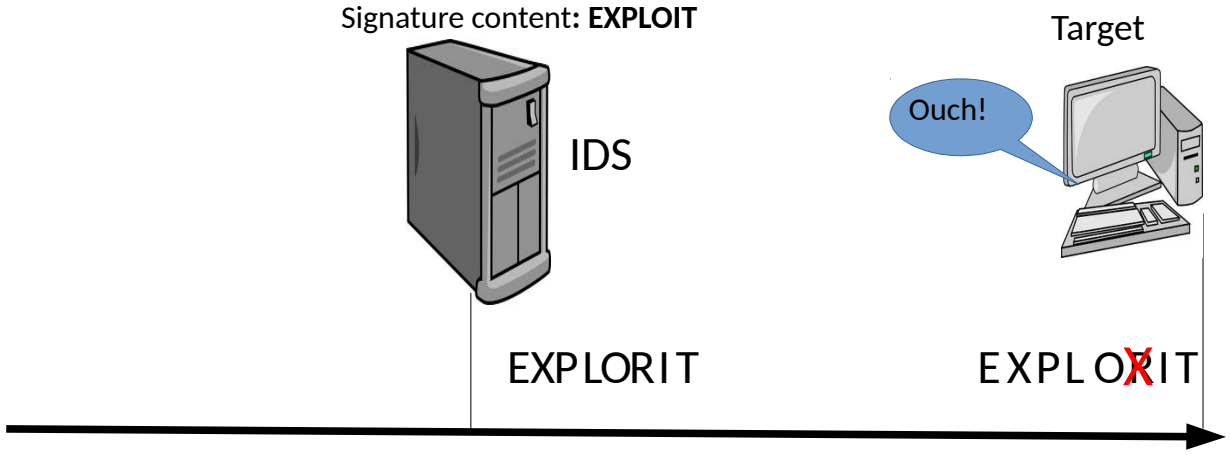
- *... <snipped for brevity>...*
- *The Next Header values in the Fragment headers of different fragments of the same original packet may differ. Only the value from the Offset zero fragment packet is used for reassembly. "*

Yet another potential ambiguity introduced by our beloved IPv6 RFCs. But, protocol ambiguities, always ring the same bell: "IDS/IPS Evasions". Let's see if we can exploit it. But before we

continue, for reasons of completeness of his paper, let's fresh our memory regarding IDS/IPS Insertion and Evasion attacks. For the rest of the paper, for reasons of simplicity, when we refer to Intrusion Detection Systems (IDS) or Intrusion Prevention Systems (IPS) we will actually refer to both (unless otherwise specified).

3 IDS/IPS Insertion/Evasion Attacks

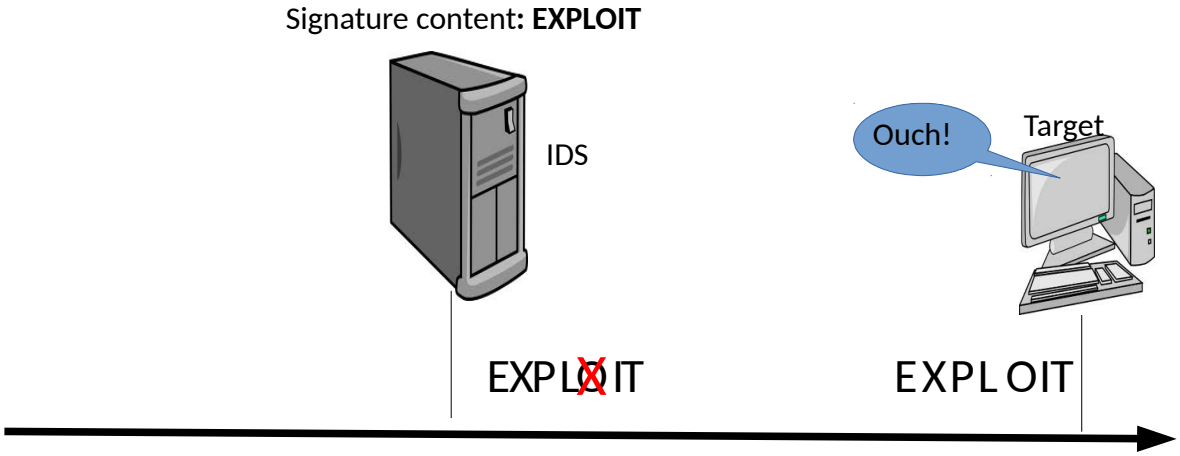
As defined in (Ptacek & Newsham, 1998), insertion attacks take place when an IDS accepts a packet that the end-system rejects (figure 4). An IDS that does this makes the mistake believes that the end-system has accepted and processed the packet when it actually hasn't. An attacker, by manipulating the sending packets properly, can use this type of attacks to defeat signature analysis and to pass undetected through an IDS.



The target rejects character "R", which IDS accepts; this breaks the IDS signature.

***Figure 4**: Example of an IDS insertion attack.*

On the other hand, an IDS evasion takes place when an end-system accepts a packet that an IDS rejects (figure 5). As it is also explained in (Ptacek & Newsham, 1998), an IDS that mistakenly rejects such a packet misses its content entirely, resulting in slipping through the IDS. Evasion attacks disrupt stream reassembly by causing the IDS to miss part of it. Such attacks are exploited even more easily that insertion attacks.



The target accepts character "O", which IDS rejects; this breaks the IDS signature.

***Figure 5**: Example of an IDS Evasion attack.*

Hence, different interpretation of some ambiguities like the ones introduced by protocol design or manipulations, if exploited properly, it can results and IDS Insertion/evasion attacks.

4 Analysis of the Problem

As RFC 4942 describes in (Davies, Krishnan & Savola, 2007), "*The IPv6 Specification contains a number of areas where choices are available to packet originators that will result in packets that conform to the specification but are unlikely to be the result of a rational packet generation policy for legitimate traffic*". Let's try to exploit one of them!

Using as an example the IPv6 datagram of figure 2.b, when this datagram has to be fragmented, the IPv6 Routing Extension header will be included in the unfragmentable part (since it needs to be processed by intermediate nodes), while the IPv6 Destination Options Extension header and layer 4, of course, will be included in the fragmentable part. Hence, the IPv6 datagram of figure 2.b will be fragmented as following (please, note the Next Header values in each header):

Fragmentable part:

IPv6 Destination Options header Next Header Value = 6	**TCP header + payload ...**

Fragment 1:

IPv6 header Next Header Value = 43	**IPv6 Routing Extension header** Next Header Value = 44	**IPv6 Fragment Extension header** Next Header Value = 60	(part 1 out of 2 of the fragmentable part)

Fragment 2:

IPv6 header Next Header Value = 43	**IPv6 Routing Extension header** Next Header Value = 44	**IPv6 Fragment Extension header** Next Header Value = 60	(part 2 out of 2 of the fragmentable part)

***Figure 6**:* Fragmentation of the IPv6 Datagram of figure 2.b

We should note two things here:

First, it is not necessarily defined where the fragmentable part should split. For instance, in the aforementioned example, this can be at the end of the IPv6 Destination Options header, somewhere in the middle of it, somewhere in the middle of the TCP header, at the end of the TCP header, or somewhere in the middle of the layer-4 payload. The actual point where this split will take place is actually a combination of the MTU size and the size of the fragmentable part. However, the size of each fragment (excluding the last one) has to be a multiple of 8-octets.

The second thing that we should note in the our example is that the Next Header value of the Fragment Extension header should be set to 60 (the protocol value of the first IPv6 Extension header of the fragmentable part of the original packet, as explained above).

What about if we have the following case?

Fragment 1:

IPv6 header Next Header Value = 43	**IPv6 Routing Extension header** Next Header Value = 44	**IPv6 Fragment Extension header** Next Header Value = 60	(part 1 out of 2 of the fragmentable part)

Fragment 2:

IPv6 header Next Header Value = 43	**IPv6 Routing Extension header** Next Header Value = 44	**IPv6 Fragment Extension header** Next Header Value = 6	(part 2 out of 2 of the fragmentable part)

Figure 7: A slightly different but still legitimate fragmentation of the IPv6 Datagram of figure 2.b

In the case of figure 7, the key difference here is that the Next Header value of the IPv6 Fragment Extension header is set to 6, and not to 60, as it was before. We make think that this make sense, at least in case the TCP header is the one that immediately follows it. However, as it was mentioned in Section 2, although this is not considered an error, still, even in this case, the Next Header value should be set to 60. But even if it is set e.g. to 6 (pointing for example to the TCP header), during the fragments reassembly only the Next Header value of the IPv6 Fragment Extension header whose offset is equal to 0 (aka the first fragment) should be used.

In the following tests, we created the officially “legitimate” but not actually expected aforementioned scenario of figure 7 in order to examine how various IPv6 nodes, hosts and security devices, react to it. As we have already said, they should accept it but they should use only the Next Header value of the IPv6 Fragment Extension header whose offset is equal to 0.

Of course, the pre-described method of figure 7 can be extended by using more than 2 fragments and messing with the Next Header values of some or even all the IPv6 Fragment Extension headers of the fragments.

5 Tested Devices – Results

Several security devices, commercial firewalls and high-end IPS, were tested using the technique described in Section 4. For reasons of simplicity of the tests, at first ICMPv6 Echo Requests were uses as “attacking” payloads. That is, because by using this, one the one hand you get a clear indication when you reach a target (by getting an ICMPv6 Echo Reply message), and on the other, you do not have for example to establish a 3-way handshaking as it is the case when using TCP as a layer-4 protocol. However, when issues were identified, other attacking payloads were also used to confirm that this technique can be used for other attacking purposes too.

5.1 Hosts – Operating Systems

The Operating Systems (OS) that were tested as potential targets using the technique of figure 7 are summarised below, all of them patced with their latest fixes as of 2nd June 2014:

- Fedora 20

- Centos 6.5
- OpenBSD 5.5
- FreeBSD 10
- Windows 8.1
- Windows 2012R2

From the aforementioned OS, only FreeBSD did not accept the packets of figure 7, and this actually because it does not accept IPv6 datagrams were the layer-4 header is at fragment other than the first. All the other, happily responded to them by sending an ICMPv6 Echo Reply message back. Definitely, according to the RFCs recommendations briefly described in Section 2, this is how they should respond, no matter if the Next Header value of the IPv6 Fragment Extension header of the second fragment should be 60 and not 6.

5.2 IDPS Devices

The authors and their group had the pleasure to test three high-end IPS up to know, updated at the latest software/firmware version, two open-source ones and a commercial one:

- HP Tipping Point IDPS (TOS Tipping Point, Package 3.6.1.4036 and digital vaccine 3.2.0.8530).
- Snort 2.9.6.1 GRE (build 56), Registered User's Release Rules, 01 May, 2014.
- Suricata 2.0.1, with Emerging Threats ETOpen Ruleset, 03 June, 2014

At these devices, before starting the attack, several other tests were also performed to observe their general behaviour. However, let's concentrate on our attacks.

5.2.1 Evading Tipping Point IDPS by Using Two Simple Fragments

By using the technique of figure 7 we found out that we could easily evade Tipping Point IDPS (send the ICMPv6 Echo Request without triggering any alert). However, since we “broke” layer 3 (IPv6 in our case), we can use this same technique for launching other attacks too.

Specifically, if there is a vulnerability of a layer-7 protocol, only this protocol is affected. For example, a discovered vulnerability in HTTP protocol will affect only applications that use this specific protocol. On the contrary, if a vulnerability in a layer-3 protocol is discovered, all the upper-layer protocols and the corresponding applications may be affected, which results in much more disastrous and extensive effects.

To confirm the above statement, we launched a “trivial” XSS attacks otherwise detected by Tipping Point IDPS. The XSS string was the following:

"GET /index.php?asd=\"><script>alert(1)</script>"

Again, our packets evaded Tipping Point IDPS and reached the target. Before we continue, let's examine how these packets looks like by using Wireshark.

First, at figure 8.a, the first of the two fragments is displayed:

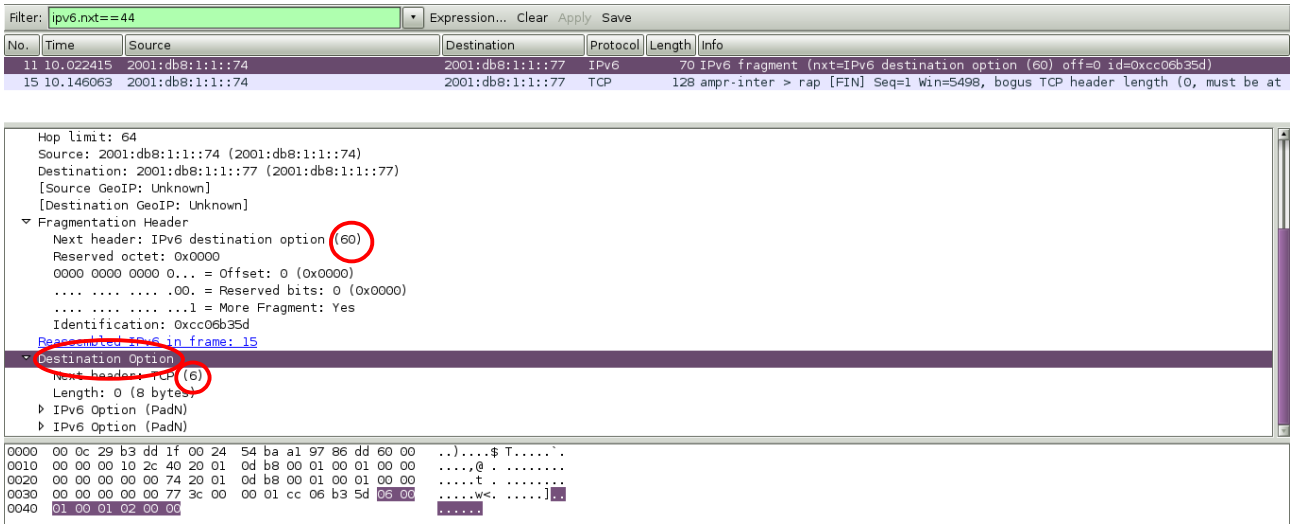


***Figure 8a*:** The first fragment of the attack

Here, we can see that:

- The Next Header value of the IPv6 Fragment Extension header is 60 (corresponding to an IPv6 Destination Options Extension header).
- The Destination Options Extension header follows the Fragment Extension header.
- The Next Header value of the IPv6 Destination Options header is 6 (pointing to TCP, our layer-4 header).
- Please, observe that the bytes of the IPv6 Destination Options header are as following (highlighted): *06 00 01 00 01 02 00 00*

So far, so good. Nothing peculiar yet. Now, let's examine the second fragment (figure 8.b), which, as we can see, contains our XSS payload. In this case, the reassembled datagram is actually presented.

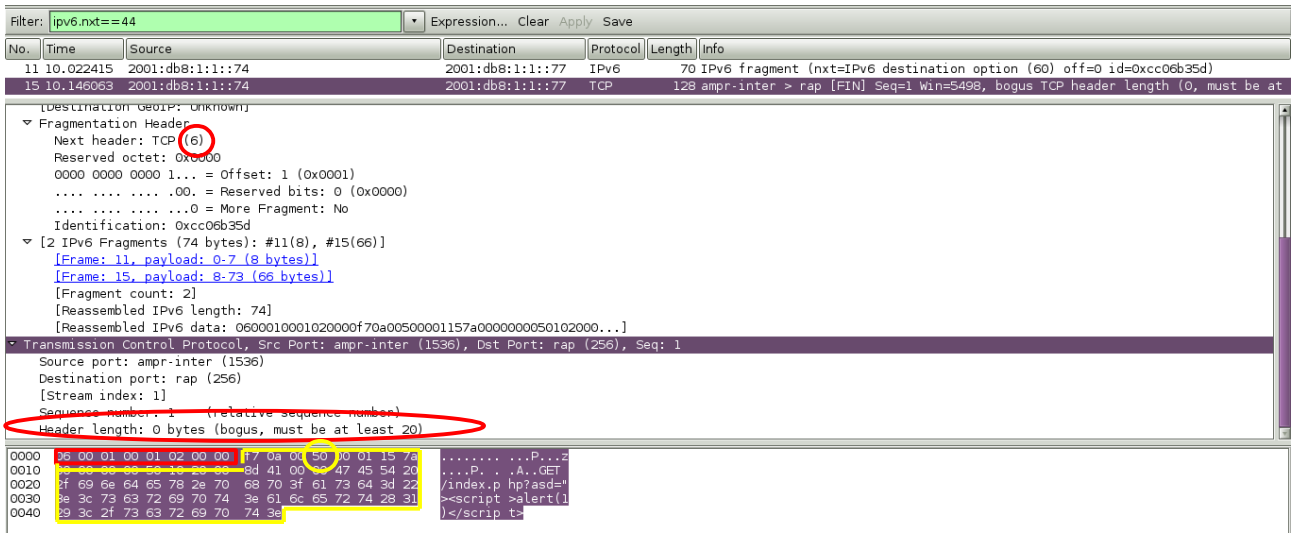


Figure 8b: The reassembled IPv6 datagram of the attack.

In the second fragment, the Next Header value of the IPv6 Fragment Extension header is set to 6

(and not to 60)! This makes the difference. Due to this, the TCP layer-4 header and its payload appears to be the highlighted one, in purple. Based on this, the destination port is 256.

Let's look a little bit closer: The Header length appears to be 0 (and it is noted as “bogus” by Wireshark, since it should be at least 20). What is this, now?

Let's look even closer: The first of the eight bytes of the – considered to be – TCP header are the following: *06 00 01 00 01 02 00 00*

But this is the IPv6 Destination Options Extension header carried by the first fragment. Correct. Obviously, due to the fact that the Next Header value of the IPv6 Fragment Extension header of the second fragment was set to 6, Wireshark, when reassembling the IPv6 datagram, misinterprets the fragmentable part and considers the IPv6 Destination Options Extension header (red rectangle) as part of the TCP header (purple, highlighted text). This is why the destination port appears to be 256 (which is also wrong).

And, where is the real destination port? First, let's find the real TCP header; this immediately follows the IPv6 Destination Options Extension header and it is marked by the yellow lines. The third and the fourth bytes of it are 00 50 respectively (in hexadecimal notation), which corresponds to 80 in decimal notation, the HTTP port. Here we are.

Our Tipping Point IDPS misinterprets the fragmented packets in the same way as Wireshark does, and so, it does not even it considers as HTTP traffic. This is how we can make it completely blind and fly under its radars no matter what kind of attack we launch.

After the XSS attack, we enabled the so called “Aggressive mode” of Tipping Point IDPS. Still, the attack remained undetected.

Then, we also enabled a feature of Tipping Point IDPS that blocks every single HTTP traffic. And the traffic still bypassed the device. Now, we shouldn't be surprised since, according to the aforementioned analysis, it “believes” that the port is completely different (256) and hence, it does not recognise it as HTTP traffic. Once again, if you break layer-3, you can break everything above it! The good thing (from an attacker's perspective) though is that potential targets (OS) interprets such traffic “normally” (subsection 5.1) and interprets it in a different way than the IPS and hence, IPS is evaded (section 3).

Finally, this technique will still be efective even if RFC 7112 (Gont, Manral, and Bonica, 2014) is implemented. That is because it is effective even if the first fragment (offset = 0) contains the entire IPv6 header chain, including the layer-4 one and even a part of the layer-4 payload. So, we are talking about a fully legitimate packet.

5.2.2 Patching the Tipping Point IDPS and Evading it Again by Repeating a Fragment

The aforementioned vulnerability is a typical example that even the minor detail can make the difference in the security field. After finding it, we decided to disclosure it responsibly and the vendor reacted rather fast and patched it. So, yet another challenge was ahead of us: Will we be able to evade it again? Can we find yet another way to abuse IPv6 Extension header to evade the device? Well, it took as a few hours (half a day actually), but we finally made it by introducing a small variation.

Specifically, we observed that even now the first fragment, which carried the IPv6 Extension header, the layer-4 payload and a part of the payload could pass through, and it was the second fragment, which carried just the second part of the payload was blocked. After that, we decided to simply ...resend the second fragment for a second time and voilà! This time came through.

The technique is displayed in figure 9.

Fragment 1:

IPv6 main header nh=44	**IPv6 Fragment Extension Header** nh=60	**IPv6 Destination Options Ext. Hdr** nh=6	**TCP Header**	**TCP payload** Part 1/2

Fragment 2:

IPv6 main header nh=44	**IPv6 Fragment Extension Header** nh=6/**60**	**TCP payload** Part 2/2

Fragment 2 (again):

IPv6 main header nh=44	**IPv6 Fragment Extension Header** nh=6	**TCP payload** Part 2/2

Figure 9: The second way used to evade the patched version of Tipping Point

That simple!

By observing the above figure, we can see that the Next Header value of the IPv6 Fragment Extension header can be either 60 (as it should be) or 6, but the one of the last fragment (the one that it is actually repeated) has to be 6 in order to evade the Tipping Point IDPS.

The packets used to evade Tipping Point IDPS against TCP port 22 (SSH) this time, are shown at the Wireshark screenshot of figure 10.

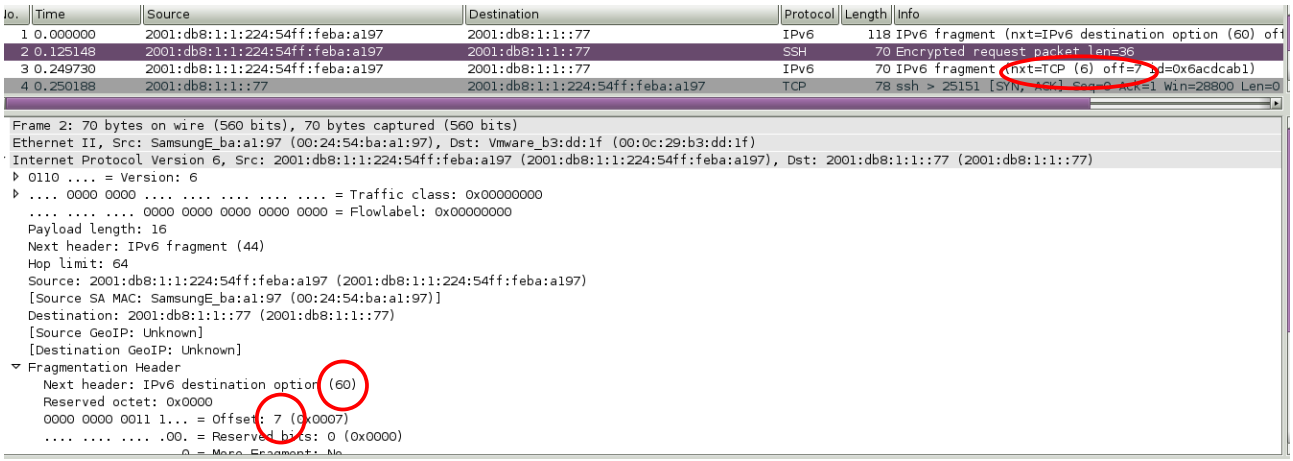


Figure 10: The packets used to evade the patched version of Tipping Point as captured by Wireshark running at the sender

In the above figure we can see that the second fragment (packet number 2) has an offset of 7 and a Next Header value of the IPv6 Extension header equal to 60, while when it is sent again (packet number 3), the offset is again 7 and the Next Header value of the IPv6 Extension header is equal to 6. However, while packet number 2 is dropped, packet number 3 passes through and a SYN-ACK is received (packet number 4).

This new technique makes Tipping Point completely blind too and it allows us, once more, to launch any kind of attacks while remaining undetected, unless this time the Aggressive mode of the device is used. However, in such a case we observed that functionality is broken and even packets which are fragmented legitimately are dropped.

5.2.3 Testing and Evading Snort

Before Enabling pre-processor decoder.rules

We performed similar tests against Snort. The technique described in subsection 5.2.1 (two fragments, wrong Next Header value at the IPv6 Fragment Extension header of the second) was found that it is not effective against Snort. However, when we increase the number of fragments from two to ten and still using a wrong Next Header value at the IPv6 Fragment Extension header of the last fragment, Snort is also evaded. In this time, the layer-4 fragment is moved to the the 10th fragment but it is the Next Header value of the IPv6 Fragment Extension header of the last fragment that makes the difference.

As an example, we repeated this test using a TCP SYN request to port 445 (SMB). To this end, we added a Snort local rule that detects any attempted TCP connection to port 445 (potential smb traffic). This rule is shown below:

alert tcp any any -> any 445 (msg: "Test SMB activity"; sid:1000001;)

Then, we used the same technique against Windows 7 (which accept SMB connections). A wireshark screenshot of the fragments sent to the target are shown at figure 11:

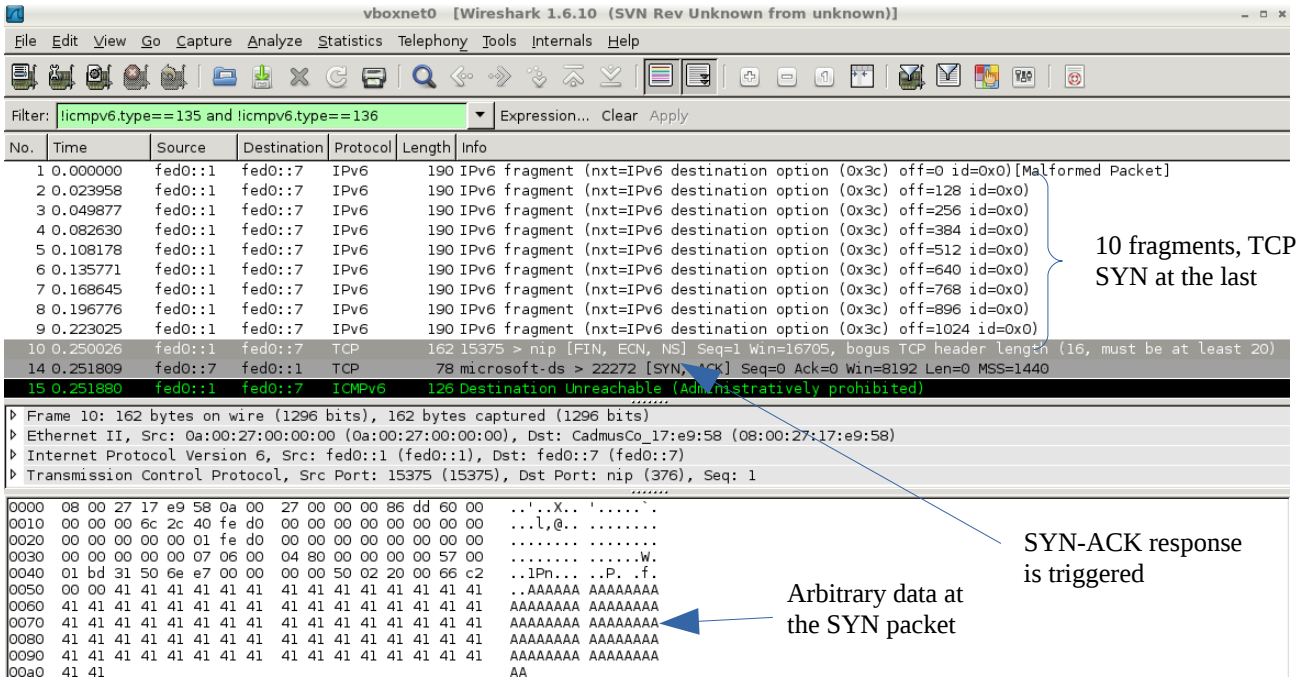


Figure 11: The packets used to evade Snort as captured by Wireshark running at the sender.

Once more, an attack technique that exploits a minor detail in the design of the IPv6 protocol is effective against a very popular IDPS device.

The aforementioned attack can actually be simplified by using just one fragment but several occurrences of IPv6 Extension headers. For instance, Topera (http://toperaproject.github.io/topera/) uses several instances of IPv6 Fragment Extension headers in an unfragmented datagram to evade Snort. However, if we have a closer look, e.g. by showing the alerts to the console, when using this attack the following output is produced:

```
Could not add event to decoderActionQ
Could not add drop event to decoderActionQ
Could not add event to decoderActionQ
Could not add drop event to decoderActionQ
Could not add event to decoderActionQ
Could not add drop event to decoderActionQ
Could not add event to decoderActionQ
Could not add drop event to decoderActionQ
Could not add event to decoderActionQ
Could not add drop event to decoderActionQ
Could not add event to decoderActionQ
Could not add drop event to decoderActionQ
Could not add event to decoderActionQ
Could not add drop event to decoderActionQ
Could not add event to decoderActionQ
Could not add drop event to decoderActionQ
Could not add event to decoderActionQ
Could not add drop event to decoderActionQ
Could not add event to decoderActionQ
Could not add drop event to decoderActionQ
Could not add event to decoderActionQ
Could not add drop event to decoderActionQ
Could not add event to decoderActionQ
Could not add drop event to decoderActionQ
```

Figure 12: Error messages when multiple IPv6 Fragment Extension headers are used (*Topera* attack).

This is also the case if, instead of the IPv6 Fragment Extension headers, IPv6 Routing Extension headers are used.

The aforementioned attack can be enhanced if, instead of using IPv6 Fragment Etension headers, IPv6 Destination Option headers are used. Specifically, in this case, when 9 or more IPv6 Destination Options headers are sent in a single, unfragmented datagram, not only the latest version of Snort is evaded, but, moreover, no error messages are triggered.

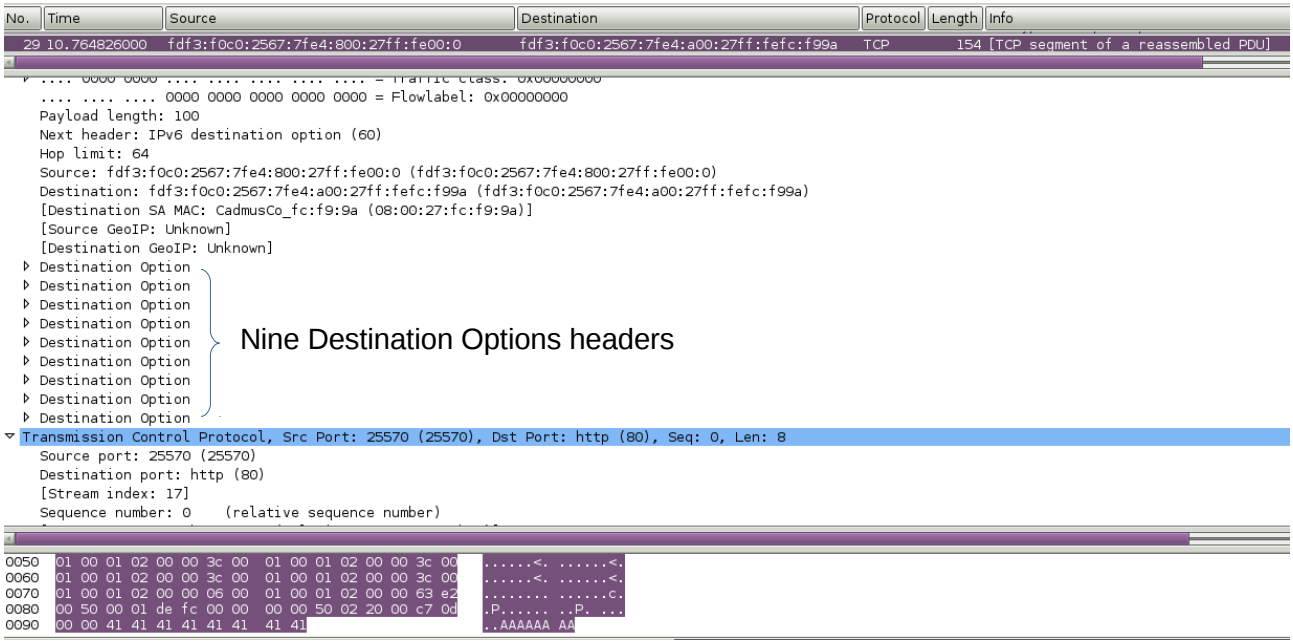


Figure 13: Evading Snort without alerts or any kind of errors.

Instead of using 9 Destination Options headers, you can use instead 8 Dest Opt and 1 Frag Ext Hdr, or, 1 Hop-by-Hop, 1 Routing Header, 1 Dest Opt Header, 1 Fragment Header, 5 Dest Opt headers, etc.

It should be noted that these packets are detected only when the packets are sent directly to the IDPS host (IPv6 destination address = IPv6 address of the IDPS) AND the corresponding port is open; but who runs its IDPS with http, smb, ftp, dns, etc ports open?

Finally, this technique will still be efective even if RFC 7112 (Gont, Manral, and Bonica, 2014) is implemented.

After Enabling pre-processor decoder.rules

Then, we tried to customise the pre-processor and decoder alerts, which, for some reason, are disabled by default , by making the following changes to the snort.conf file:

- Uncommenting line: include $PREPROC_RULE_PATH/decoder.rules
- Commenting out line: #config disable_decode_alerts

After these changes, the following warning is triggered:

06/17-20:42:48.432882 [**] [116:456:1] (snort_decoder) WARNING: too many IP6 extension headers [**] [Classification: Misc activity] [Priority: 3] {IPV6-OPTS} fdf3:f0c0:2567:7fe4:800:27ff:fe00:0 -> fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa

After checking out, we found out that this preproc rule triggers a warning when nine or more Extension headers are used. Although this rule seems to be effective against the tested case, we believe that this approach is not the best one, for the following reasons:

1. The "attack" itself (http traffic in our tests) is still NOT detected.

2. This warning could be ignored by an analysts, especially since quite a few false "alarms" (warnings) are generated by the preproc/decoder.rules. To name a few:

a) alert (msg:"DECODE_ICMP6_TYPE_OTHER"; sid:431; gid:116; rev:1; metadata:rule-type decode; classtype:misc-activity;) - when simple TCP over IPv6 traffic is sent.

b) alert (msg:"DECODE_IPV6_BAD_FRAG_PKT"; sid:458; gid:116; rev:1; metadata:rule-type decode; classtype:protocol-command-decode;) - when atomic fragments are sent. Atomic fragments are fully legitimate packets (RFC 6946, & Section 5 of RFC 2460).

c) alert (msg:"DECODE_IPV6_DSTOPTS_WITH_ROUTING"; sid:292; gid:116; rev:1; metadata:rule-type decode; classtype:protocol-command-decode;) - according to RFC 2460 this is a legitimate usage of IPv6 Destination Options header in combination with Routing header (section 4.1, note 1).

d) alert (msg:"DECODE_IPV6_UNORDERED_EXTENSIONS"; sid:296; gid:116; rev:1; metadata:rule-type decode; classtype:protocol-command-decode;) - when an IPv6 Destination Options header follows an IPv6 Fragment Extension header, which is also considered to be a legitimate usage by RFC 2460 (section 4.1, note 3).

3. Most important of all, from an RFCs perspective, there can be fully legitimate packets that include nine or more IPv6 Extension Headers, because:

a) There have been defined already nine IPv6 Extension headers, the one of which (Destination Options header) can be used twice, while may be more will come in the future:

0	IPv6 Hop-by-Hop Option	[RFC2460]
43	Routing Header for IPv6	[RFC2460][RFC5095]
44	Fragment Header for IPv6	[RFC2460]
50	Encapsulating Security Payload	[RFC4303]
51	Authentication Header	[RFC4302]
60	Destination Options for IPv6	[RFC2460]
135	Mobility Header	[RFC6275]
139	Host Identity Protocol	[RFC5201]
140	Shim6 Protocol	[RFC5533]

b) To make matter worse, according to RFC 2460 the upper-layer can also be an IPv6 main header. In this case (section 4.1 RFC 2460):

"*If the upper-layer header is another IPv6 header (in the case of IPv6 being tunneled over or encapsulated in IPv6), it may be followed by its _own_ extension headers*". Thus, in such a case the number of IPv6 Extension headers will increase significantly.

If someone argues that in the real world it will be very rare, if not impossible, to see so many IPv6 Extension headers in an IPv6 datagram, we would definitely agree with him/her. So, yes, from a functionality perspective, preproc/decoder.rules 116:456 can be effective, at least for the time being.

However, we believe that for the aforementioned reasons, the usage of preproc decoder.rules and specifically the rule 116:456 is not the best approach to handle this attack. Snort, as well as any

other IDPS that supports IPv6 should correctly decode and identify “malicious” traffic and should not be based on simlistic rules the usage of which may be controversial.

5.2.4 Evading Suricata

This situation was rather more difficult. Suricata uses several rules that detect duplicated IPv6 Extension headers, as well as IPv6 Destination Options Only Paading. An example of these specific Suricata alerts is displayed below:

06/06/2014-14:21:23.156410 [**] [1:2200018:1] SURICATA IPv6 duplicated Destination Options extension header [**] [Classification: (null)] [Priority: 3] {TCP} fdf3:f0c0:2567:7fe4:0800:27ff:fe00:0000:15011 -> fdf3:f0c0:2567:7fe4:0a00:27ff:fe74:ddaa:80
06/06/2014-14:21:23.156410 [**] [1:2200089:1] SURICATA IPv6 DSTOPTS only padding [**] [Classification: (null)] [Priority: 3] {TCP} fdf3:f0c0:2567:7fe4:0800:27ff:fe00:0000:15011 -> fdf3:f0c0:2567:7fe4:0a00:27ff:fe74:ddaa:80

However, after several tests we found out that even Suricata can be evaded. This is the case when:

- An IPv6 Destination Option header is used a part of the fragmentaple piece of the IPv6 tatagram.
- The IPv6 Destinations Option header is padded with six (6) octets of bytes (at least).
- The IPv6 datagram is fragmented in at least 7 fragments, which are sent mis-ordered.

The corresponding traffic is displayed at figure 14.

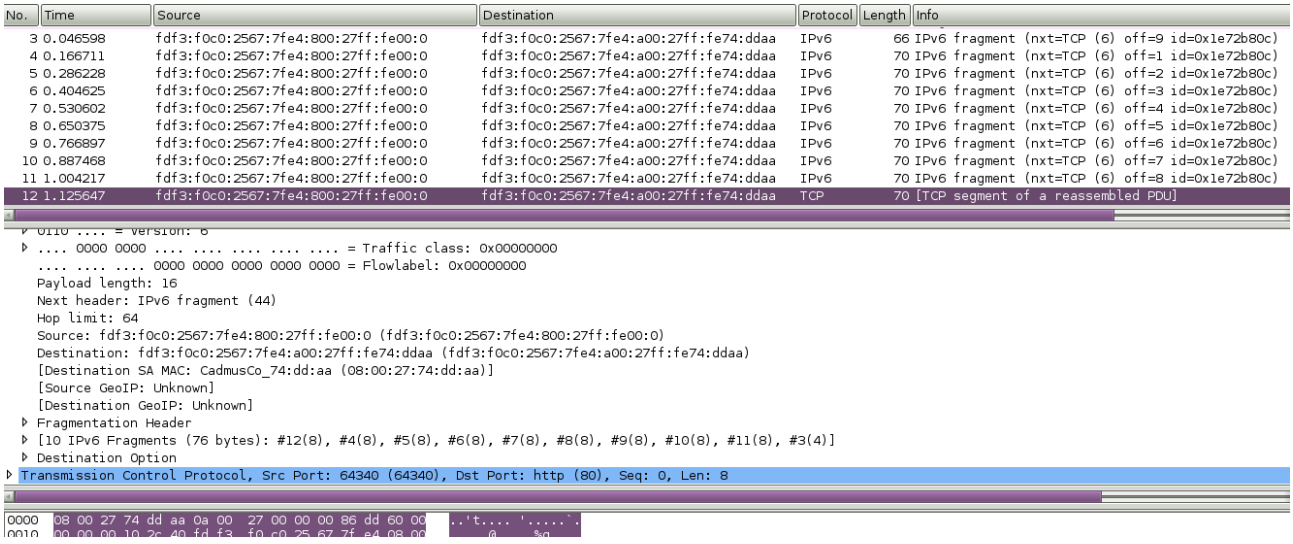


Figure 14: The packets used to evade Suricata as captured by Wireshark running at the sender

So, once more, by combining IPv6 Extension headers and fragmentation in an “unusual” (but not necessarily a non-legitimate) way, yet another popular IDPS can be evaded. In this case though, the layer-4 header has to be in a fragment other than the first in order the technique to be effective. Hence, when finally RFC 7112 (Gont, Manral, and Bonica, 2014) will be implemented, these fragments will not be considered to be legitimate and so, they should/may be dropped.

It should be noted that not all Operating Systems respond to such packets. However, the following one happily respond to them:

- Centos 6.5
- Windows 2012R2
- OpenBSD 5.5
- Windows 8.1

On the contrary, Fedora 20 and FreeBSD 10 do not respond to such packets.

5.3 Firewalls

The question that arises after the previous tests/results is whether this attack can also affect several other security devices, like IPv6 firewalls. To this end, several commercial firewalls were also checked regarding the aforementioned issue. These firewalls are the following:

- Cisco ASA 5505 running firmware 9.1(4)
- Checkpoint Gaia Release 77.10 running on commodity hardware
- Juniper SRX 100H2 running JunOS 12.1X46-DH.2
- Fortinet Fortigate 200B running v5.0,build0252 (GA Patch 5)

The key difference between the firewalls and IPS devices is the following:

Firewalls are usually used having a “Default Deny” rule and specifically whitelisting the permitted traffic. So, you have to “twist” malicious traffic to resemble a legitimate one in order to bypass them. Even if you manage to do so, this will probably result in the discard of the packet by the final target too. On the contrary, IPS devises permit the traffic by default (kind of “Default Allow”) and block the traffic that matches a specific signature (blacklisting malicious traffic). So, you just have to break a signature in order to evade them, which is definitely much easier to do, as it has already been show.

This, of course, does not mean that IPv6 firewalls, or at least some of them, do not suffer by the same issues like the ones described above. It “just” has to be combined with other, more sophisticated combinations in order to be effective. The authors, as part of their research at IPv6, are currently investigating several such cases trying to find out whether IPv6 firewalls can be affected too, a case which would be definitely much more disastrous.

6 Proposed Mitigation Techniques

After describing and analysing the attacks in detail, let's try to find a way to defend against it, starting from the most long-term ones.

First of all, RFCs should strictly define the exact legitimate usage and should explicitly recommend that the nodes should discard any non-legitimate packets. “Loose” specifications result in ambiguities and so, they introduce potential attack vectors. Functionality and flexibility is definitely a good thing, but security is non-negotiable.

Moreover, vendors should definitely make fully-compliant products and test them thoroughly before claiming IPv6-readiness. Offering IPv6 connectivity and some IPv6 features should not be the criterion to claim that they make IPv6 compliant product. Research has shown up to know that

several security issues still remain and a lot of work needs to be done towards this direction too.

Furthermore, up to the moment that the design of IPv6 will not suffer from any flaws, and the time that vendors will check their products more thoroughly, what can we do as "simple" users to protect ourselves in the IPv6 era? A "quick-and-dirty" approach: You should configure your devices to drop IPv6 Extension headers not used in your environment like Destination Options one, Hop-by-hop, etc. and/or fragmentation. This approach is definitely not the best. We should not drop functionality, this should not be the solution. This is why such a measure should be considered just as a temporary one.

However, still the most important required issue is the so called IPv6 Security awareness. Meet the protocol, play with it, test it in your lab and in your environment, study thoroughly potential configurations and finally, use it. You will have to to do it, sooner or later. The earlier you will be familiarised with it, the better.

7 Conclusions and Future Work

In this paper, three novel approaches were presented which exploit a minor issue in the design of the IPv6 protocol in order to circumvent IPS security devices. Real-world tests and results from high-end commercial devices were presented and the problem was analysed. Similar potential security implications to other devices, like firewalls, were also discussed. The results of this paper demonstrated, once more, that much more testing of the IPv6 implementation is required; but definitely, this does not mean that its deployment should be postponed. Instead, we should "dive" into it and try to increase our "IPv6 security awareness". By better knowing it, we will be able to configure our IPv6 networks and our hosts properly and finally, to protect them.

A lot of research is currently ongoing by the authors of this paper and their team. Still, several issues have to be investigated. More security high-end devices are being tested continuously (depending on their availability) and even more are going to follow. Finally, we are also in the process of expanding our research to evaluate the IPv6 Capabilities of Main IPAM Systems with a particular focus on auditing/forensics capabilities, a huge desideratum in the enterprise space.

8 References

Atlasis A. (2012a), Attacking IPv6 Implementation Using Fragmentation, *Black Hat Europe 2012*, March 14-16 2012, Amsterdam.

Atlasis A. (2012b), Security Impacts of Abusing IPv6 Extension Headers, *Black Hat Abu Dhabi 2012*, December 3-6 2012, Abu Dhabi.

Atlasis A. (2013a), Fragmentation Overlapping Attacks Against IPv6: One Year Later, Troopers13 – IPv6 Security Summit 2013, March 2013, Heidelberg.

Atlasis A. (2013b), IPv6 Extension Headers: New Features, and New Attack Vectors, Troopers13 – IPv6 Security Summit 2013, March 2013, Heidelberg.

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

Davies E., Krishnan S. and Savola P. (2007), IPv6 Transition/Coexistence Security Considerations,

RFC 4942, September 2007.

Gont, F., Manral, V., and R. Bonica, "Implications of Oversized IPv6 Header Chains", RFC 7112, January 2014.

IANA (2014), Protocol Numbers, retrieved March 26, 2014 from http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml#protocol-numbers-1

Internet Society (2014), The World is Different Now, retrieved March 25, 2014 from http://www.worldipv6launch.org/

Ptacek T., Newsham T. (1998). Insertion, Evasion and Denial of Service: Eluding Network Intrusion Detection, *Secure Networks, Inc*.

Rey E. (2014), Why IPv6 Security is so hard – Structural Deficits of IPv6 and their Implications, March 2014, Heidelberg.