Evading Intrusion Detection/Prevention Systems by Exploiting IPv6 Features

Antonios Atlasis
aatlasis@secfu.net

Enno Rey
erey@ernw.de

Rafael Schaefer
rschaefer@ernw.de
Who We Are

- Antonios Atlasis
  - IT Security enthusiast.
  - Researching security issues for fun.
- Enno Rey
  - Old school network security guy.
- Rafael Schaefer
  - ERNW young researcher
  - BSc Thesis in Evading IDPS by Abusing IPv6 Extension Headers
Outline of the Presentation

- Introduction
  - IPv6 is here
  - What IPv6 brings with it: The Extension Headers
- Problem Statement. Describe the Mess
- Tested IDPS devices:
  - Suricata
  - Tipping Point
  - Sourcefire
  - Snort
- Mitigation & Conclusions
What’s New in IPv6?

- Several things have changed.
- Yes, the HUGE address space is the most well-known one.
- But, we also have the IPv6 Extension Headers 😊
The IPv6 Main Header vs the IPv4 Header

<table>
<thead>
<tr>
<th>v4</th>
<th>Version</th>
<th>IHL</th>
<th>Type of Service</th>
<th>Total Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Identification</td>
<td>x</td>
<td>D</td>
<td>M</td>
</tr>
<tr>
<td>TTL</td>
<td>Protocol</td>
<td>Header Checksum</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Source Address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Destination Address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IP Options (optional)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>v6</th>
<th>V</th>
<th>Traffic C</th>
<th>Flow Label</th>
<th>Payload length</th>
<th>Next</th>
<th>Hop Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IPv6 Source Address</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IPv6 Destination Address</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What an IPv6 Datagrams Looks Like...
The IPv6 Extension Headers

- **Currently defined:**
  - Hop-by-Hop Options [RFC2460]
  - Routing [RFC2460]
  - Fragment [RFC2460]
  - Destination Options [RFC2460]
  - Authentication [RFC4302]
  - Encapsulating Security Payload [RFC4303]
  - MIPv6, [RFC6275] (Mobility Support in IPv6)
  - HIP, [RFC5201] (Host Identity Protocol)
  - shim6, [RFC5533] (Level 3 Multihoming Shim Protocol for IPv6)

- There is a **RECOMMENDED** order.
- All (but the Destination Options header) SHOULD occur at most once.
- How a device should react if **NOT**?
Transmission & Processing of IPv6 Ext. Hdrs

- RFC 7045. Any forwarding node along an IPv6 packet’s path:
  - should forward the packet **regardless** of any Extension Headers that are present.
  - MUST recognize and deal appropriately with all standard IPv6 Extension Header types.
  - SHOULD NOT discard packets containing **unrecognised** Extension Headers.
Problem 1: Too Many Things to Vary

- Variable types
- Variable sizes
- Variable order
- Variable number of occurrences of each one.
- Variable fields

IPv6 = f(v,w,x,y,z,)

3/30/2015
Problem 2: Fragmentation

Both the *Fragmentable* and the *Unfragmentable* parts may contain any IPv6 Extension headers.

Problem 1 becomes more complicated.
Problem 3: How IPv6 Extension Headers are Chained?

| IPv6 header | IPv6 Routing Extension header | IPv6 Destination Options header | TCP header + payload ...
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Next Header Value = 43</td>
<td>Next Header Value = 60</td>
<td>Next Header Value = 6</td>
<td></td>
</tr>
</tbody>
</table>

- **Next header fields:**
  - Contained in IPv6 headers, identify the type of header immediately following the current one.
  - They use the same values as the IPv4 Protocol field.

- Contents of Next header value:
  - 43: IPv6 Routing Extension header
  - 60: IPv6 Protocol Options header
  - 6: TCP header + payload
Why IPv6 Header Chaining is a Problem?

<table>
<thead>
<tr>
<th>Fragmentable part</th>
<th>Unfragmentable part</th>
<th>Fragmentable part</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPv6 DestOpt Hdr</td>
<td>IPv6 main header</td>
<td>IPv6 main header</td>
</tr>
<tr>
<td>Next header value = 6</td>
<td>Next header value = 43</td>
<td>Next header value = 43</td>
</tr>
<tr>
<td>TCP</td>
<td>IPv6 Routing Hdr</td>
<td>IPv6 Routing Hdr</td>
</tr>
<tr>
<td></td>
<td>Next header value = 44</td>
<td>Next header value = 44</td>
</tr>
<tr>
<td>TCP payload</td>
<td>IPv6 Fragment Hdr</td>
<td>IPv6 Fragment Hdr</td>
</tr>
<tr>
<td></td>
<td>Next header value = 60</td>
<td>Next header value = 60</td>
</tr>
</tbody>
</table>

(part 1 out of 2 of the fragmentable part)

(part 2 out of 2 of the fragmentable part)
To sum up the Mess in IPv6

- Vary:
  - The types of the IPv6 Extension headers
  - The order of the IPv6 Extension headers
  - The number of their occurrences.
  - Their size.
  - Their fields.
  - The Next Header values of the IPv6 Fragment Extension headers in each fragment.
  - Fragmentation (where to split the datagram)

- And combine them.
Did You Notice?

- When designing/writing IPv6 protocols & parsers they didn’t pay too much attention to #LANGSEC.
- Please visit www.langsec.org.
We May Have a Fundamental Problem Here...

- There is too much flexibility and freedom...
- Which is usually inverse proportional to security :-)
- And it can potentially lead to a complete chaos...
So, What Can Possibly Go Wrong?

- Detection Signatures, e.g. used by IDPS rules, etc. are based on blacklisting traffic.

- What if we confuse their parsers by abusing IPv6 Extension headers in an unusual / unexpected way?
All this is not just a theory

- You can reproduce all the results that we shall demonstrate using *Chiron*
- It can be downloaded from: [http://www.secfu.net/tools-scripts/](http://www.secfu.net/tools-scripts/)
- A dedicated hands-on workshop presenting all new features will be given tomorrow.
  - Including a CTF 😊
Our Tests at a Glance

- Four (4) IDPS (two open-source, two high-end commercial ones).
- At least twelve (12) different evasion techniques, in total.
- All of them 0-days at the time of the finding.
- All of them were reported (disclosed responsibly).
- Most of them were patched, either promptly or not that promptly 😊.
- One of them still suffers from a 0-day IPv6 evasion technique.
Evading Suricata

- Versions 2.0.1, 2.0.2 and 2.0.3 were evaded one by one by using various techniques.
- All of them can be reproduced using Chiron.
- We will demonstrate the latest one.
Evading Suricata 2.0.3

Note: Other combinations of Extension Headers can also work (your ...homework)
Time for Action

- Demo against Suricata 2.0.3
Suricata Developers in Each Reported Case Reacted really Fast

Suricata 2.0.4 Available!

The OISF development team is pleased to announce Suricata 2.0.4. This release fixes a number of important issues in the 2.0 series.

This update fixes a bug in the SSH parser, where a malformed banner could lead to evasion of SSH rules and missing log entries. In some cases it may also lead to a crash. Bug discovered and reported by Steffen Bauch.

Additionally, this release also addresses a new IPv6 issue that can lead to evasion. Bug discovered by Rafael Schaefer working with ERNW GmbH.

Download

Get the new release here: http://www.openinfosecfoundation.org/download/suricata_2.0.4.tar.gz

Changes

- Bug #1376: ipv6 defrag issue with routing headers
- Bug #1278: ssh banner parser issue
- Bug #1254: sig parsing crash on malformed rev keyword
- Bug #1267: issue with ipv6 logging
- Bug #1273: Lua – http request line not working
- Bug #1284: AF_PACKET IPS mode not logging drops and stream inline issue
Evading TippingPoint, “the Old Way” (March 2014)

<table>
<thead>
<tr>
<th>IPv6</th>
<th>TCP</th>
<th>TCP payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>DestOptHdr</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Next header value = 6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1st fragment

<table>
<thead>
<tr>
<th>IPv6</th>
<th>IPv6</th>
<th>TCP payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>main header</td>
<td>FragmentHdr</td>
<td></td>
</tr>
<tr>
<td>Next header value = 44</td>
<td>Next header value = 60</td>
<td></td>
</tr>
</tbody>
</table>

Unfragmentable part

2nd fragment

<table>
<thead>
<tr>
<th>IPv6</th>
<th>IPv6</th>
<th>TCP payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>main header</td>
<td>FragmentHdr</td>
<td></td>
</tr>
<tr>
<td>Next header value = 44</td>
<td>Next header value = 6</td>
<td></td>
</tr>
</tbody>
</table>

Fragmentable part

Note: Layer-4 header can be in the 1st fragment and the attack still works
Evading TippingPoint, “The Old Way”
That First One Was Patched...

But Again We Had a New One ;-)

<table>
<thead>
<tr>
<th>Model Number</th>
<th>110</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial Number</td>
<td>U110C-50F</td>
</tr>
<tr>
<td>TOS Version</td>
<td>3.6.2.4109</td>
</tr>
<tr>
<td>Digital Vaccine</td>
<td>3.2.0.8565</td>
</tr>
</tbody>
</table>

- Configured to:
  - Operate inline at Layer 2.
  - Block any HTTP traffic.
  - Additional XSS rules (to test attacks at the payload too).
Evading TippingPoint, after First Patching

<table>
<thead>
<tr>
<th>1st fragment</th>
<th>2nd fragment (again)</th>
<th>2nd fragment</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IPv6 main header</strong>&lt;br&gt;Next header value = 44</td>
<td><strong>IPv6 main header</strong>&lt;br&gt;Next header value = 44</td>
<td><strong>IPv6 main header</strong>&lt;br&gt;Next header value = 44</td>
</tr>
<tr>
<td><strong>IPv6 Fragment Hdr</strong>&lt;br&gt;Next header value = 60</td>
<td><strong>IPv6 Fragment hdr</strong>&lt;br&gt;Next header value = 60/6</td>
<td><strong>IPv6 Fragment hdr</strong>&lt;br&gt;Next header value = 6</td>
</tr>
</tbody>
</table>

**Unfragmentable part**

**Fragmentable part**

**Note:** Layer-4 header can be in the 1st fragment and the attack still works.
Time for some more ...Action

- Evading TippingPoint 3.6.2 demonstration
- Quite similar situations, as expected.
- Still, the latest open-source version suffer from a 0-day...
The Chronicle of the Communication

- We first contacted the Snort devs in 17th of June.
- We reported to Cisco/Sourcefire another issue in Sept 14.
- We disclosed publicly the issues in BlackHat Europe 2014.
- Latest Snort v. 2.9.7.0 provides a potential mitigation.
- In the meantime, Sourcefire was also “silently” patched.
Fair enough!

- Time for live demos for both.
Evading Sourcefire

- Sourcefire, Model 3D7020 (81) Version 5.2.0.3 (Build 48).
- Preproc decoder rules were enabled:
  - GID 116 family and specifically, SID 458 (IPV6_BAD_FRAG_PKT), 272 and 273 are enabled.
- This attack doesn’t work against latest Sourcefire.
Evading Sourcefire

Note: Next header values for Fragment Extension headers: The correct ones (60)
Evading Sourcefire


- Destination Option
- Destination Option
- Destination Option
- Fragmentation Header

- [IPv6 Fragments (24 bytes): #5(8), #6(16)]
  - [Frame: 5, payload: 0-7 (8 bytes)]
  - [Frame: 6, payload: 0-23 (16 bytes)]
  - [Reassembled IPv6 length: 24]
  - [Reassembled IPv6 data: 3c:00001000102000003a0001000010200000800001035129c0000]

- Destination Option
- Destination Option

Internet Control Message Protocol v6

Type: Echo (ping) request [128]
Code: 0
Checksum: 0x1035 [correct]
Identifier: 0x129c
Sequence: 0
[Response in: 7]
Evading Snort

- Latest Snort version, 2.9.7.0
- Preproc decoder rules are enabled:
  - GID 116 family and specifically, SID 458 (IPV6_BAD_FRAG_PKT), 272 and 273 are enabled.
- This attack is STILL effective against latest Snort.
Enabling Preproc Decoder rules

This can be tricky:

- Make the following changes to the snort.conf file:
  - Uncomment line: include $PREPROC_RULE_PATH/decoder.rules
  - Comment line: #config disable_decode_alerts
- Make sure that the following rules are enabled in /etc/snort/preproc_rules/decoder.rules
  - "DECODE_IP6_EXCESS_EXT_HDR"; sid:456; gid:116;
- Moreover, you can have:
  - "DECODE_IPV6_BAD_FRAG_PKT"; sid:458; gid:116; -> triggers a warning for Atomic Fragments
  - "DECODE_IPV6_UNORDERED_EXTENSIONS"; sid:296; gid:116; -> may trigger false alarms
### Evading Snort

#### IPv6 main header
- **Hop-by-Hop**
- **Type-3 Routing Hdr**
- **DestOpt Hdr**
- **Fragment Hdr**
- **DestOpt Hdr**

#### Unfragmentable part

#### Fragmentable part
- **Hop-by-Hop**
- **Type-3 Routing Hdr**
- **DestOpt Hdr**
- **Fragment Hdr**
- **Layer 4 header**
- **Layer 4 payload**

**Note:** Next header values for Fragment Extension headers: the correct ones (60)
Snort 2.9.7.0 Changelog

- Maximum number of Extension Headers can be configured manually.
- Eight (8) by default.

* doc/snort_manual.tex, src/dynamic-examples/dynamic-rule/detection_lib_meta.h, src/dynamic-plugins/sf_dynamic_engine.h, src/dynamic-plugins/sf_dynamic_meta.h, src/dynamic-plugins/sf_dynamic_preprocessor.h, src/dynamic-plugins/sf_engine/examples/detection_lib_meta.h, src/dynamic-plugins/sf_engine/sf_snort_packet.h, src/preprocessors/Stream6/snort_stream_tcp.c, src/decode.c, src/decode.h, src/encode.c, src/parser.c, src/parser.h, src/snort.c, src/snort.h:

  Added a new config option `max_ip6_extensions` to change the maximum number of IPv6 extension headers decoded. Thanks to Antonio Atlasis for providing data to the ChangeLog.
How to Harden Snort 2.9.7.0

- `/etc/snort.conf:

  ```
  config max_ip6_extensions: 1
  ```

- **01/11-16:40:33.391730 [**] [116:456:1] (snort_decoder)
  WARNING: too many IP6 extension headers [**]
  [Classification: Misc activity] [Priority: 3] {IPV6-OPTS}
  fe80::800:27ff:fe00:0 -> fe80::a00:27ff:fe74:ddaa

- Question: Is this the optimum way of handling the issue?
RFCs should strictly define the exact legitimate usage.

- “Loose” specifications result in ambiguities and so they introduce potential attack vectors.
- Functionality and flexibility are definitely good things, but security is non-negotiable.

Make fully-compliant IPv6 products and test them thoroughly.
Technical Mitigations

- Implementation of RFC 7112.
  - An intermediate system (e.g., router or firewall) that receives an IPv6 First Fragment that does not include the entire IPv6 Header Chain MAY discard that packet.
  - Still, not a panacea...

- For the time being:
  - Configure your devices to drop IPv6 Extension Headers not used in your environment. OR
  - At least sanitize traffic before the IDPS.
This Is how a Certain Vendors Interprets This

From sk39374

- **How to handle IPv6 Extension Headers**
  
  By default, Check Point Security Gateway drops all extension headers, except fragmentation. This can be adjusted by editing the `allowed_ipv6_extension_headers` section of `$FWDIR/lib/table.def` file on the Security Management Server.

  Furthermore, as of R75.40 there is an option to block type zero even if `Routing` header is allowed. It is configurable via a kernel parameter `fw6_allow_rh_type_zero`. The default of 0 means it is always blocked. If the value is set to 1, then the action is according to `allowed_ipv6_extension_headers`. 
In Case You still Want to Use an IDPS ...

- you MUST (header-wise) scrub the traffic before entering the IDPS.
The Most Important “Take Away”

- These are just some of the IPv6 “grey areas”. Other may also exist.
  - Hint: MLD comes to mind...

- IPv6 security awareness.
  - Test it and use it, in your lab or not.
  - You will have to do it, sooner or later, anyway...
Questions?

- You can reach us at: 
  - aatlasis@secfu.net, www.secfu.net
  - erey@ernw.de, www.insinuator.net
  - rschaefer@ernw.de

- Follow us at: 
  - @AntoniosAtlasis
  - @Enno_Insinuator
There’s never enough time...

THANK YOU... ...for yours!

Tool & Slides:
https://www.insinuator.net
http://www.secfu.net/tools-scripts/