Evading Intrusion Detection/Prevention Systems by Exploiting IPv6 Features

Antonios Atlasis

aatlasis@secfu.net

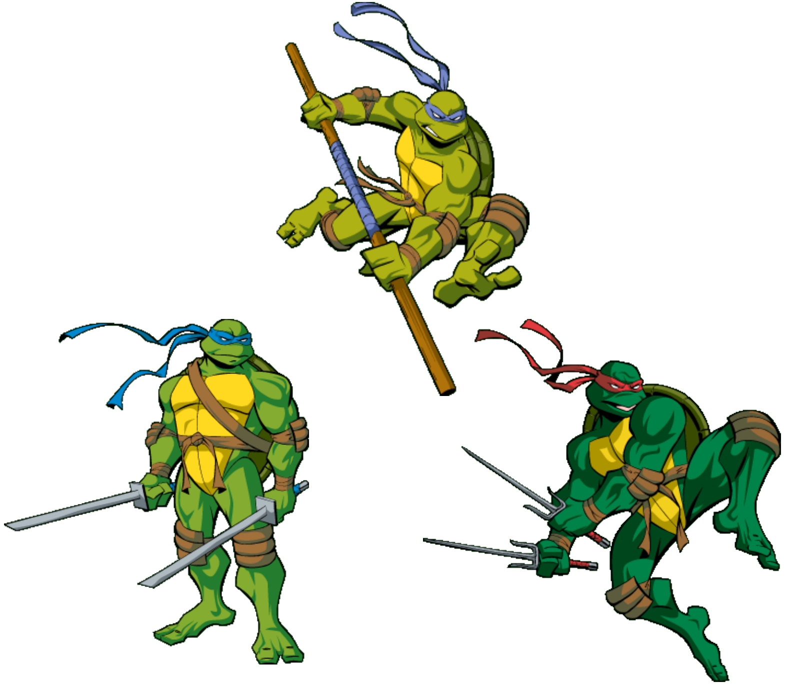
Enno Rey

ERNW GmbH
erey@ernw.de

Rafael Schaefer

ERNW GmbH
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

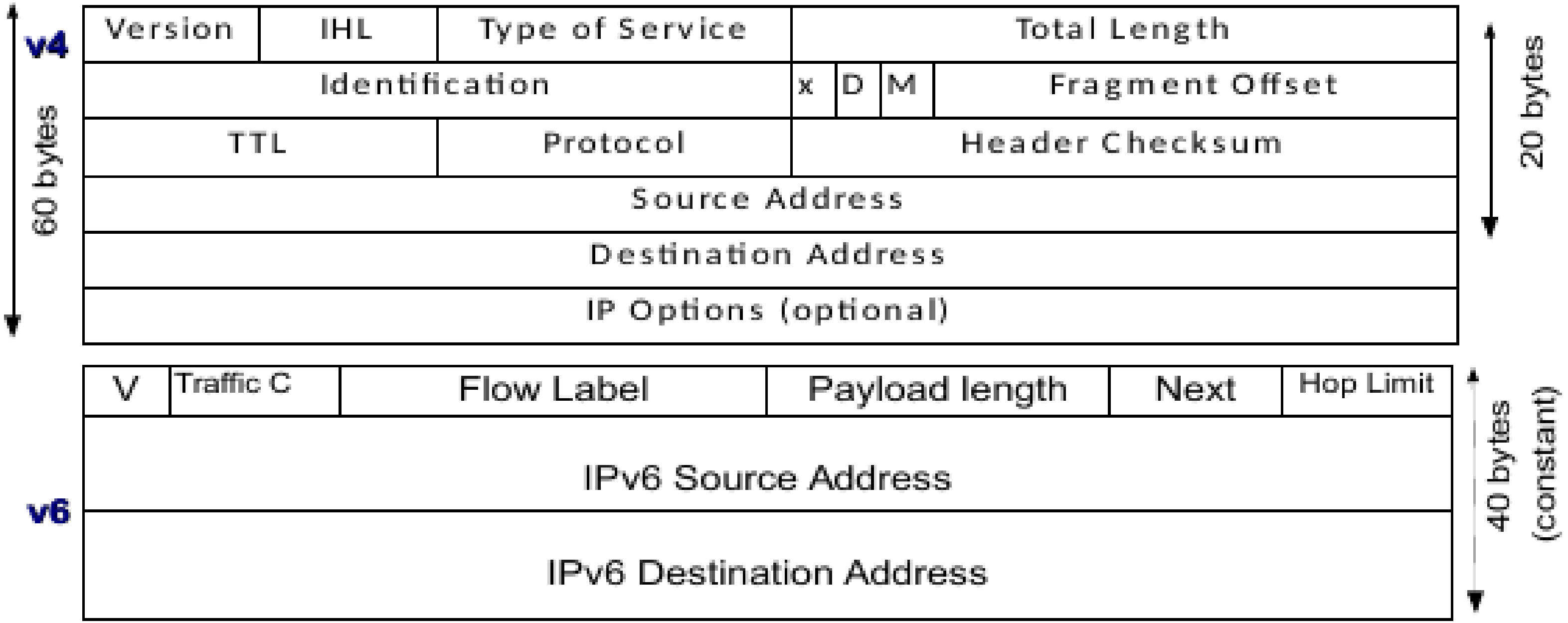
Outline of the Presentation

- 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-know one.
- But, we also have the IPv6 **Extension Headers** ☺

The IPv6 Main Header vs the IPv4 Header



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** ?

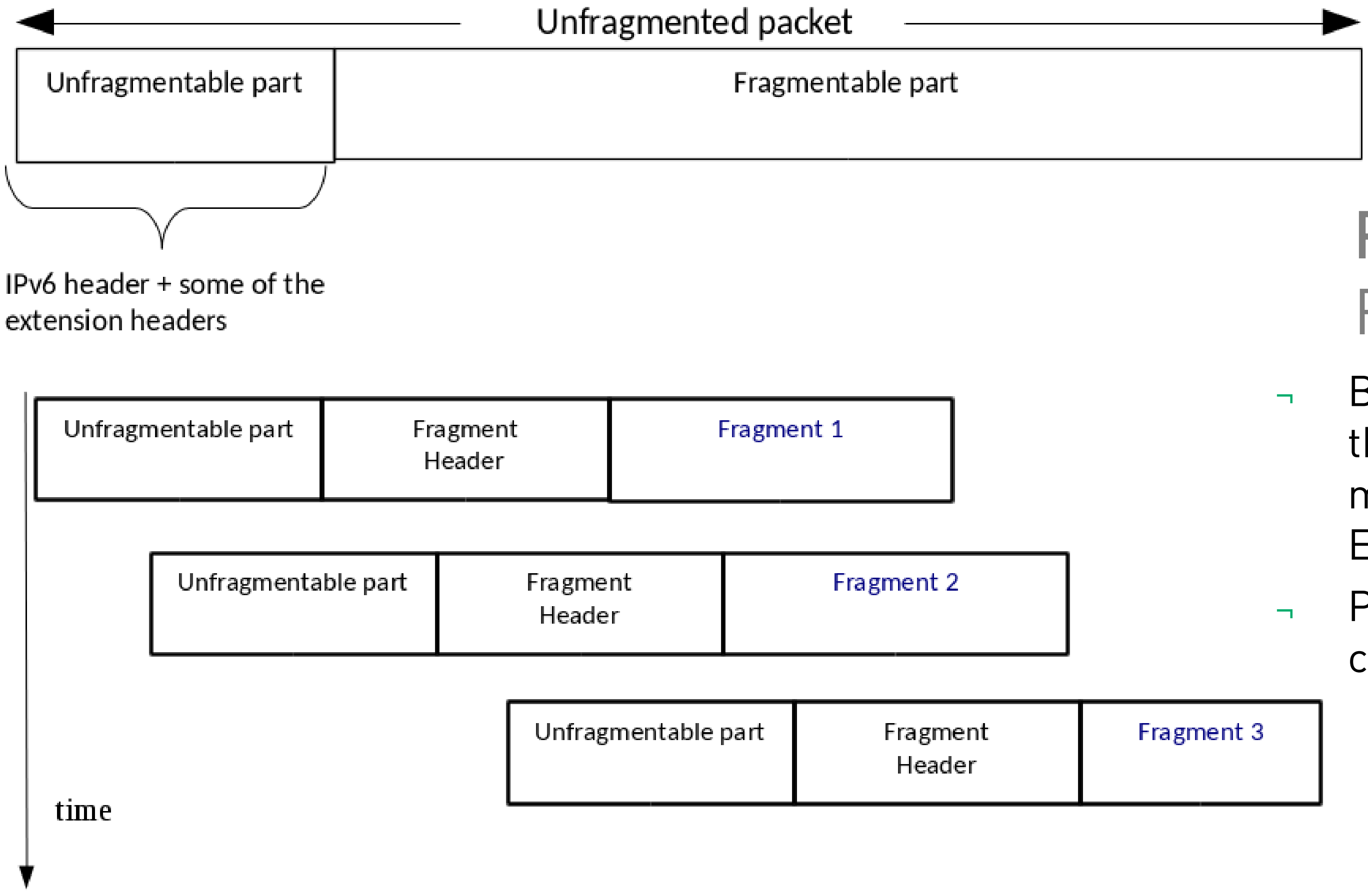
Transmission & Processing of IPv6 Ext. Hdrs

- 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,)



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 ...
Next Header Value = 43	Next Header Value = 60	Next Header Value = 6	

- **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.

Why IPv6 Header Chaining is a Problem?

IPv6 DestOpt Hdr Next header value = 6	TCP	TCP payload

Fragmentable part

1st fragment

IPv6 main header Next header value = 43	IPv6 Routing Hdr Next header value = 44	IPv6 Fragment Hdr Next header value = 60	(part 1 out of 2 of the fragmentable part)

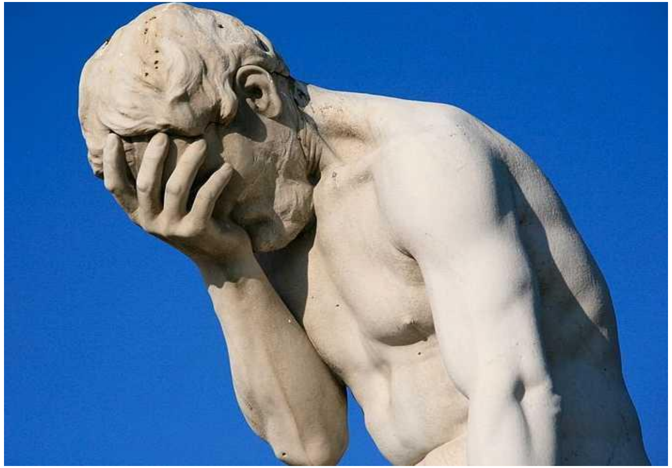
Unfragmentable part

Fragmentable part

2nd fragment

IPv6 main header Next header value = 43	IPv6 Routing Hdr Next header value = 44	IPv6 Fragment Hdr Next header value = 60	(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)

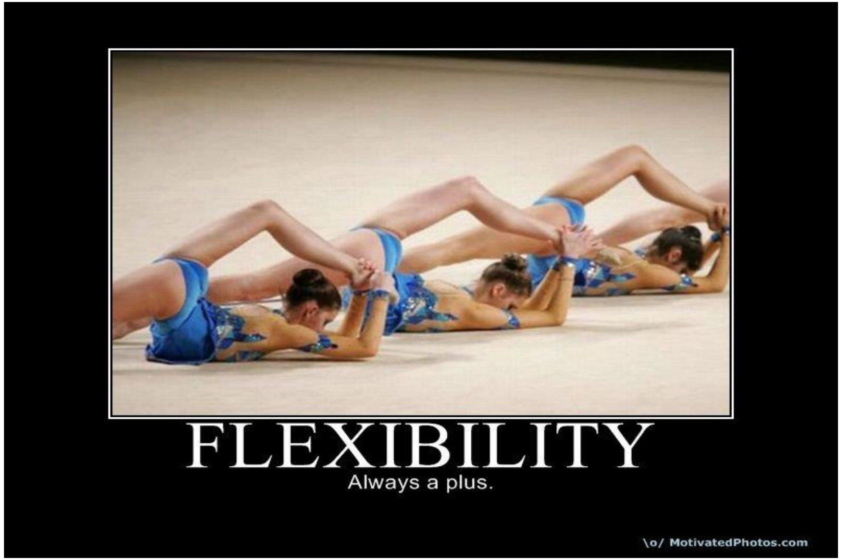
- And combine them.

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 *cha0s*…



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?

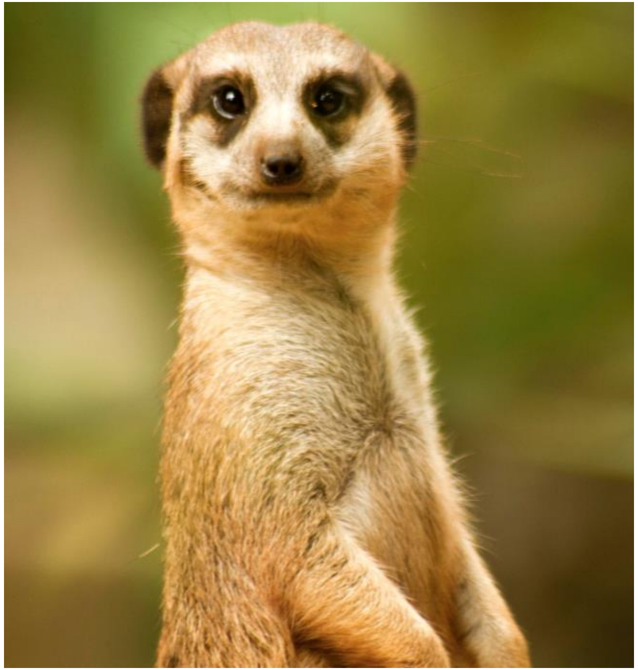
All this is not just a theory

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

Our Tests at a Glance

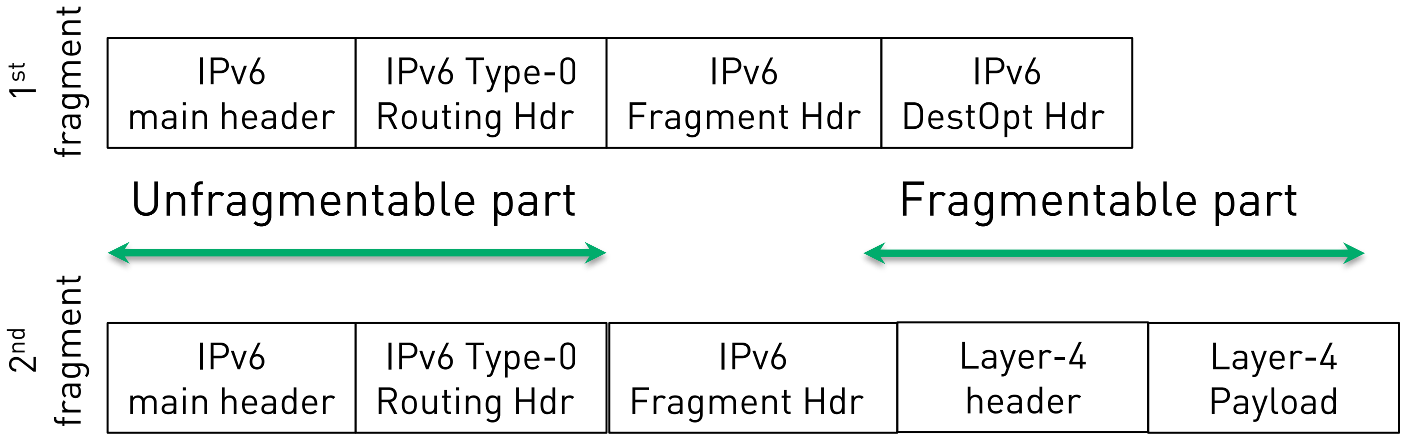
- Four (4) IDPS (two open-source, two high-end commercial ones).
- At least twelve (12) different evasion techniques, in total.
- All of them 0-days at the time of the finding.
- All of them were reported (disclosed responsibly).
- Most of them were patched, either promptly or not that promptly ☺.
- 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 #1276: 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)

IPv6 DestOpt Hdr Next header value = 6	TCP	TCP payload

Fragmentable part

1st fragment

IPv6 main header Next header value = 44	IPv6 Fragment Hdr Next header value = 60	(part 1 out of 2 of the fragmentable part)

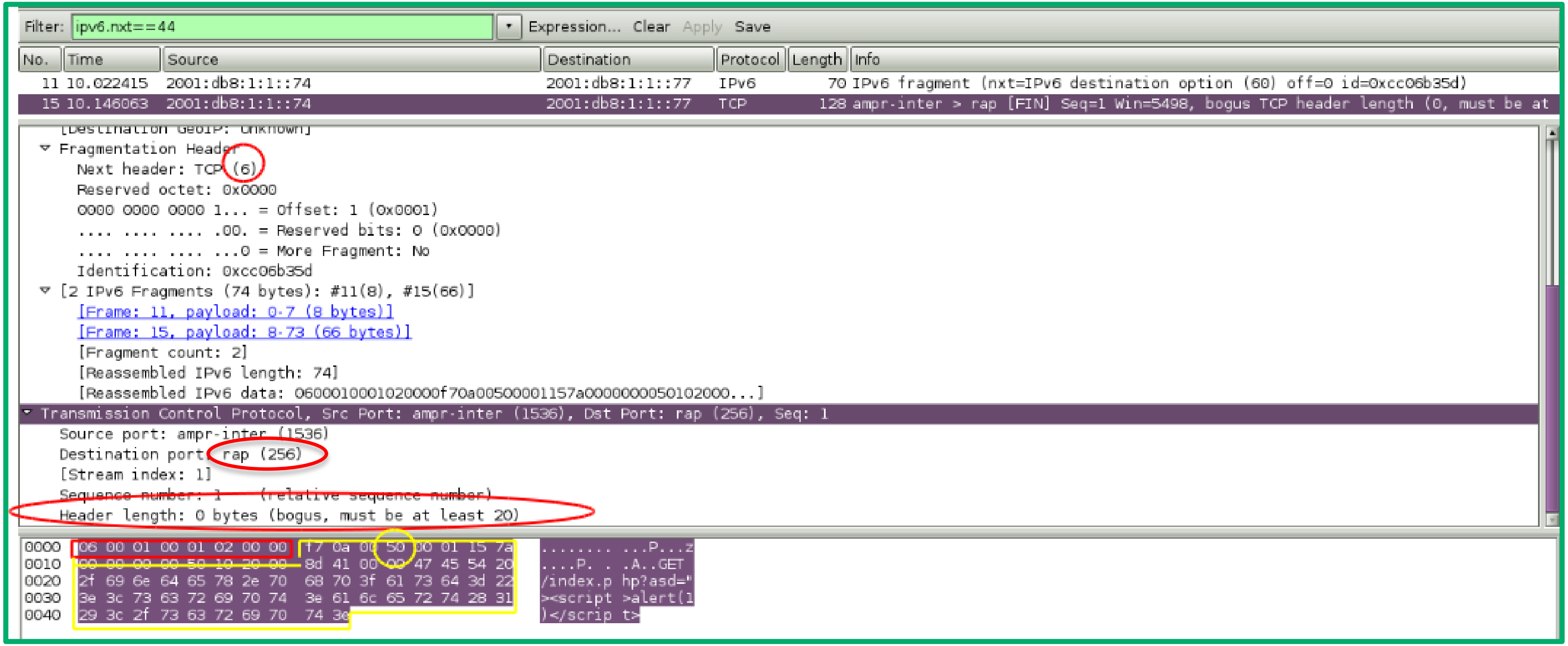
Unfragmentable part ⟷ Fragmentable part ⟷

2nd fragment

IPv6 main header Next header value = 44	IPv6 Fragment Hdr Next header value = 6	(part 2 out of 2 of the fragmentable part)

<u>Note</u>: Layer-4 header can be in the 1st fragment and the attack still works

Evading TippingPoint, "The Old Way"



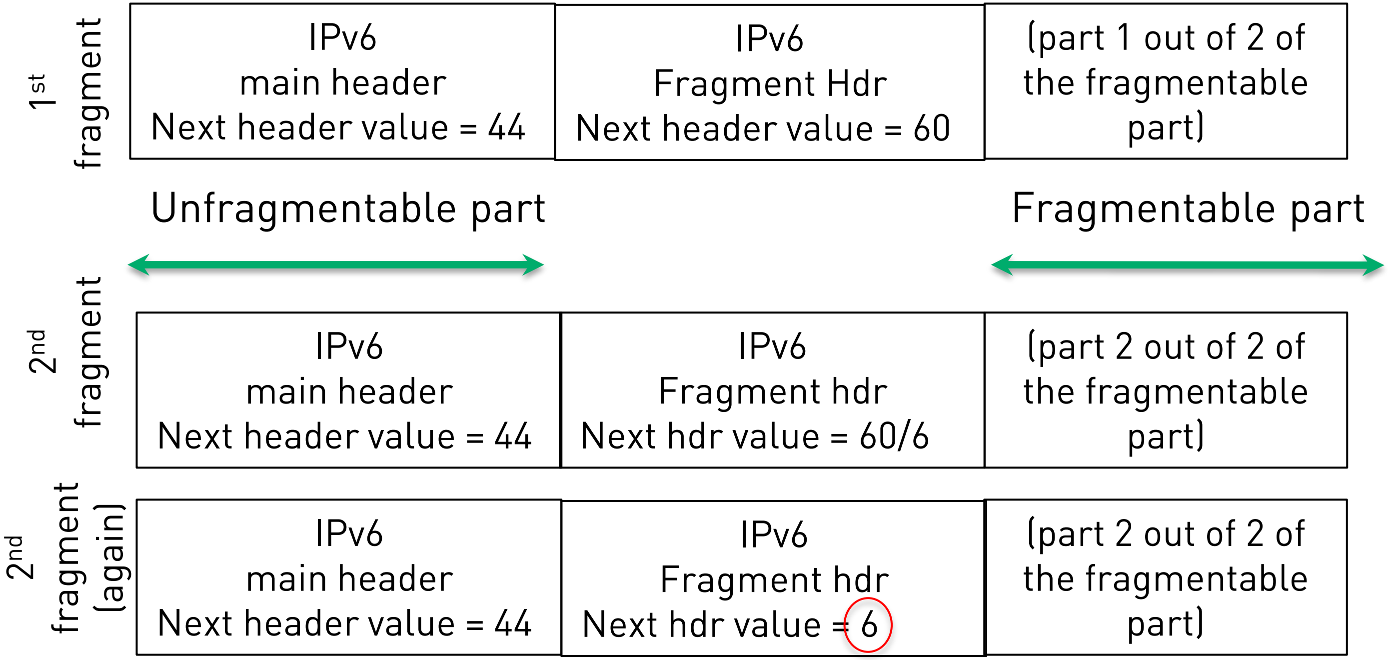
That First One Was Patched…

But Again We Had a New One ;-)

Model Number	110
Serial Number	U110C-50F
TOS Version	3.6.2.4109
Digital Vaccine	3.2.0.8565

- **Configured to:**
  - Operate inline at Layer 2.
  - Block <u>any</u> HTTP traffic.
  - Additional XSS rules (to test attacks at the payload too).

Evading TippingPoint, after First Patching



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

Snort / Sourcefire

- 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.

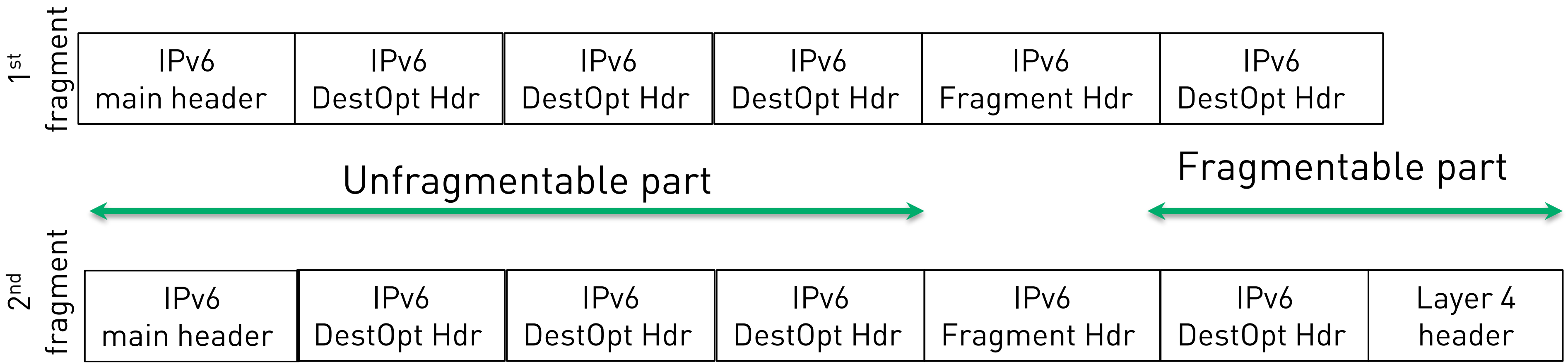
Fair enough!

- 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

No.	Time	Source	Destination	Protocol	Length	Info
5	0.064967	2001:db8:1:1::aa	2001:db8:1:1::cc	IPv6	94	IPv6 fragment (nxt=IPv6 destination option (60) off=0 id=0x56eecfa7)[Ma
6	0.190343	2001:db8:1:1::aa	2001:db8:1:1::cc	ICMPv6	102	Echo (ping) request id=0x129c, seq=0, hop limit=64 (reply in 7)
7	0.190407	2001:db8:1:1::cc	2001:db8:1:1::aa	ICMPv6	62	Echo (ping) reply id=0x129c, seq=0, hop limit=128 (request in 6)

```
    Source: 2001:db8:1:1::aa (2001:db8:1:1::aa)
    Destination: 2001:db8:1:1::cc (2001:db8:1:1::cc)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
  ▷ Destination Option
  ▷ Destination Option
  ▷ Destination Option
  ▷ Fragmentation Header
  ▽ [2 IPv6 Fragments (24 bytes): #5(8), #6(16)]
      [Frame: 5, payload: 0-7 (8 bytes)]
      [Frame: 6, payload: 8-23 (16 bytes)]
      [Fragment count: 2]
      [Reassembled IPv6 length: 24]
      [Reassembled IPv6 data: 3c000100010200003a000100010200008000103512 9c0000]
  ▷ 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]
```

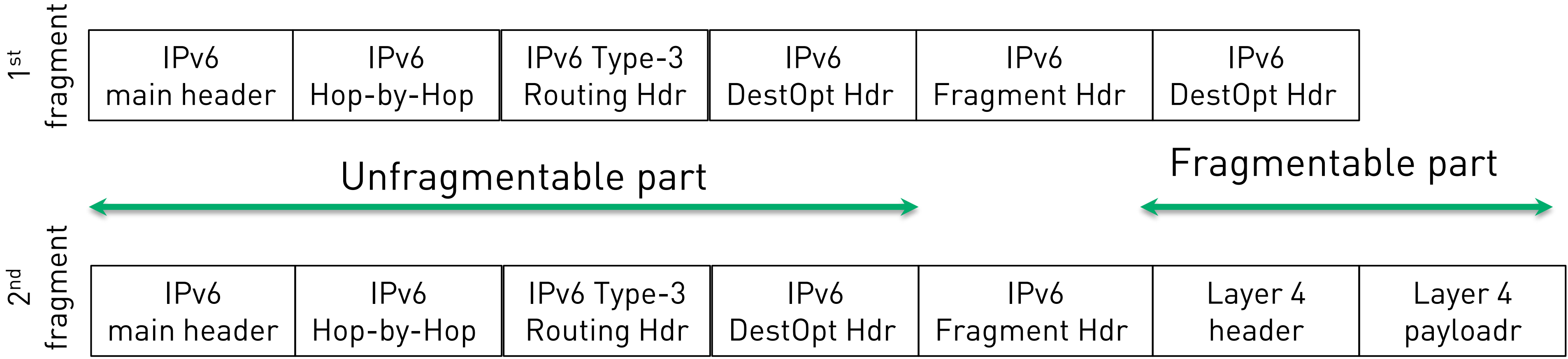
Evading Snort

- 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



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?

"Culture" of Mitigations

- 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

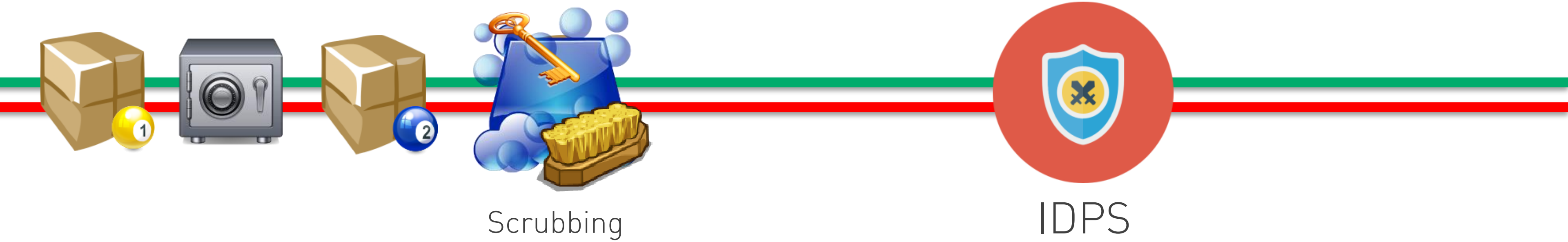
From sk39374

- 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 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/