March 14-16, 2012
NH Grand Krasnapolsky Hotel
Amsterdam, Netherlands

ATTACKING IPV6 IMPLEMENTATION USING FRAGMENTATION

Antonios Atlasis, antonios.atlasis@cscss.org
Centre for Strategic Cyberspace + Security Science

Bio

- Independent IT Security analyst and researcher.
- Over 20 years of diverse Information Technology experience.
- Instructor and software developer, etc.
- Hobbies: bug-finding.
- Recently joined the **Centre for Strategic Cyberspace + Security Science** non-profit organisation.
- E-mail: antonios.atlasis@cscss.org

CSCSS CENTRE FOR STRATEGIC CYBERSPACE + SECURITY SCIENCE

Presentation Outline

- Some background regarding fragmentation in IPv4 and its consequences.
- Fragmentation in IPv6.
- Examination of fragmentation issues in IPv6 implementation against some of the most popular OS – Examples.
- Conclusions

Some Background

IP Fragmentation

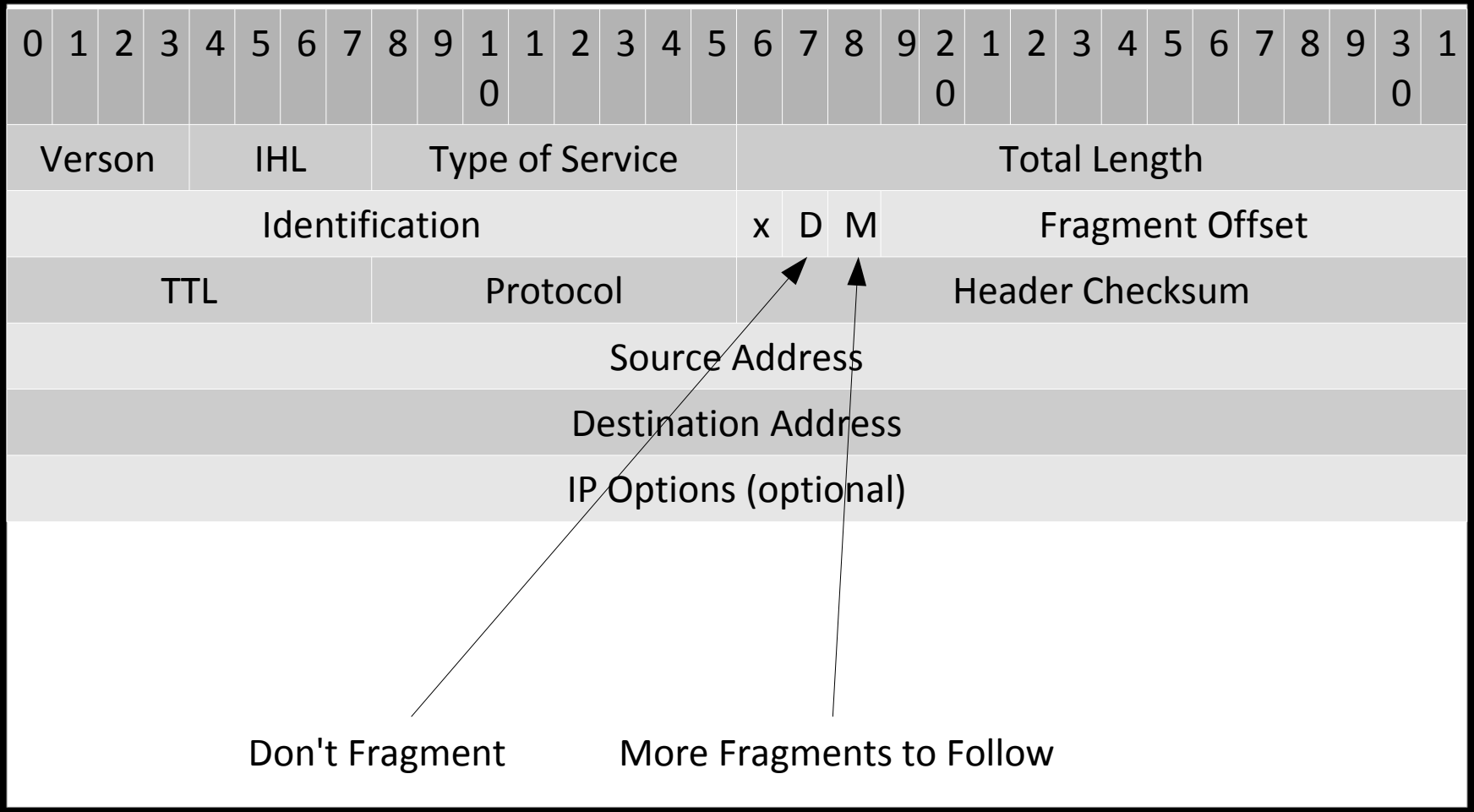
- Usually a normal event.
- Required when the size of the IP datagram is bigger than the Maximum Transmission Unit (MTU) of the route that the datagram has to traverse (e.g. Ethernet MTU=1500 bytes).
- Packets reassembled by the receiver.

Fragmentation in IPv4

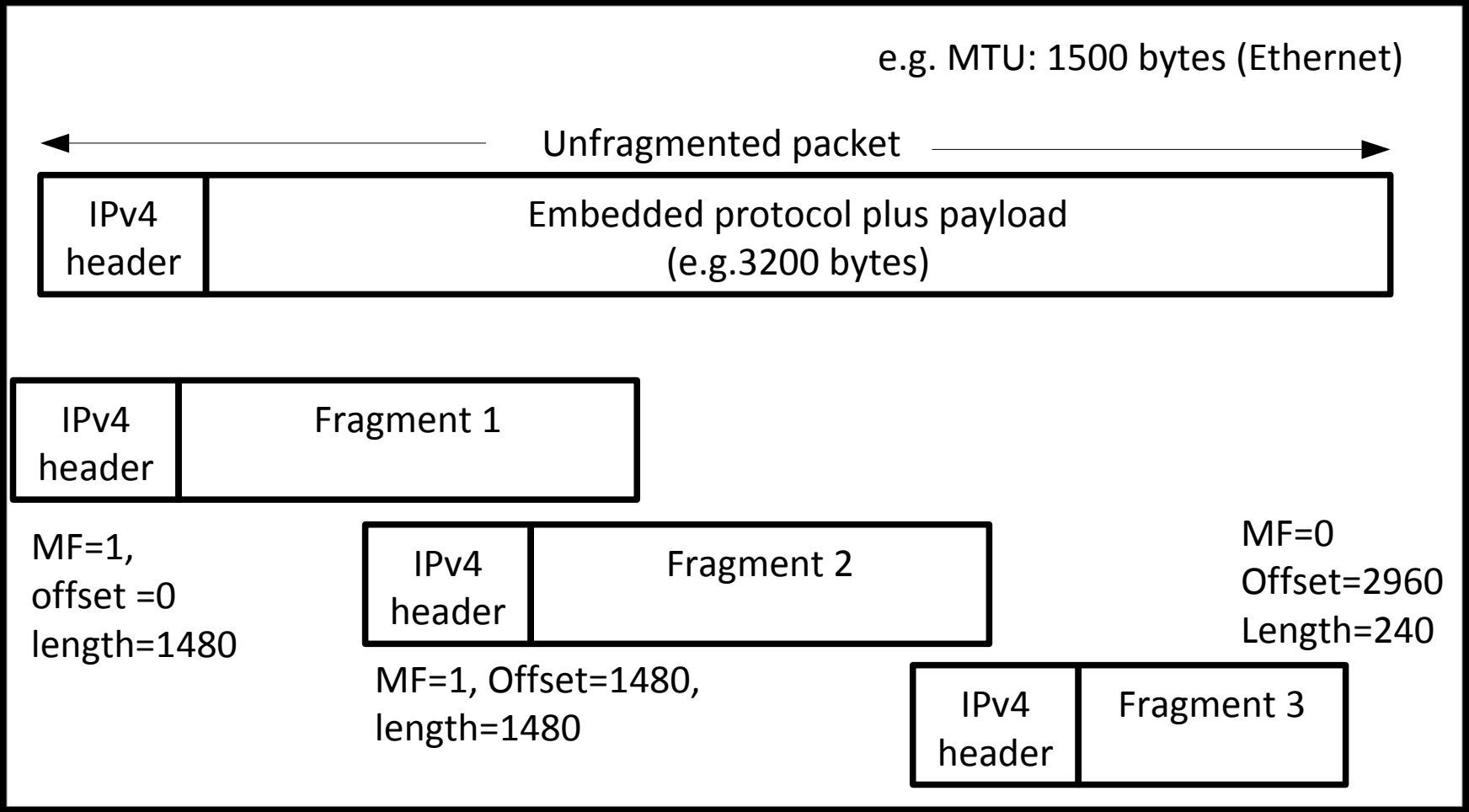
- Share a common fragment identification number (which is the IP identification number of the original datagram).
- Define its **offset** from the beginning of the corresponding unfragmented datagram, the **length** of its payload and a **flag** that specifies whether another fragment follows, or not.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

IPv4 Header

RFC 791



IPv4 Fragmentation



(some of the) Consequences of malformed fragmentation

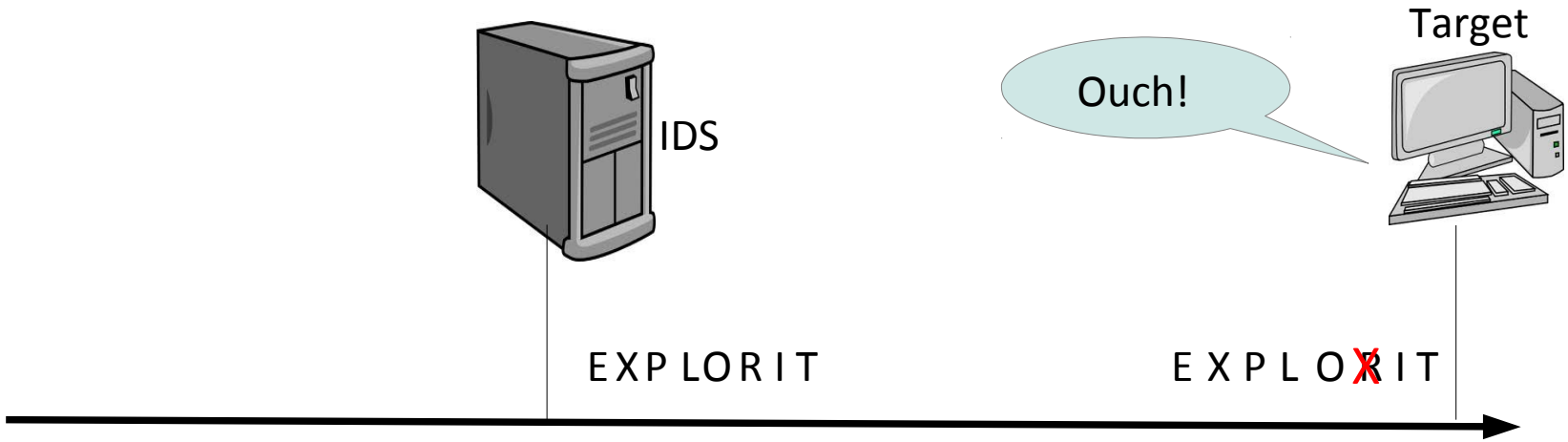
When it all started

- "***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



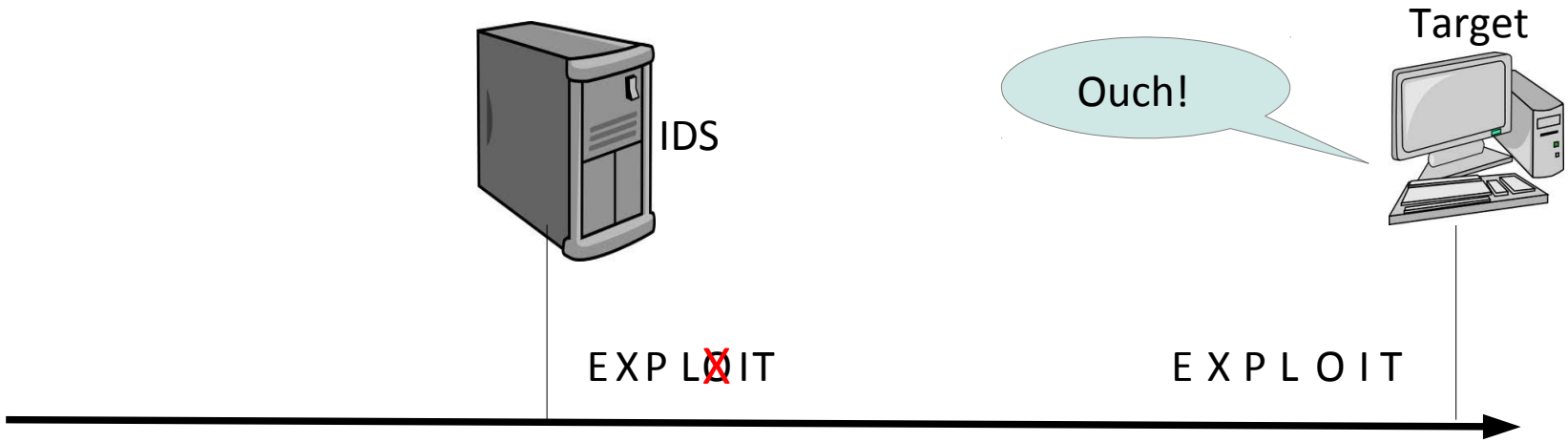
The target rejects character “R”, which IDS accepts; this breaks the IDS signature.

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**

Fragmentation Attacks

- Disordered arrival of fragments.
- IDS flooding by partial fragmented datagrams.
- Selective dropping of old and incomplete fragmented datagram.
- Overlapping fragments.
- IP Options in Fragment Streams.

What Changes in IPv6 (regarding fragmentation)

In IPv6

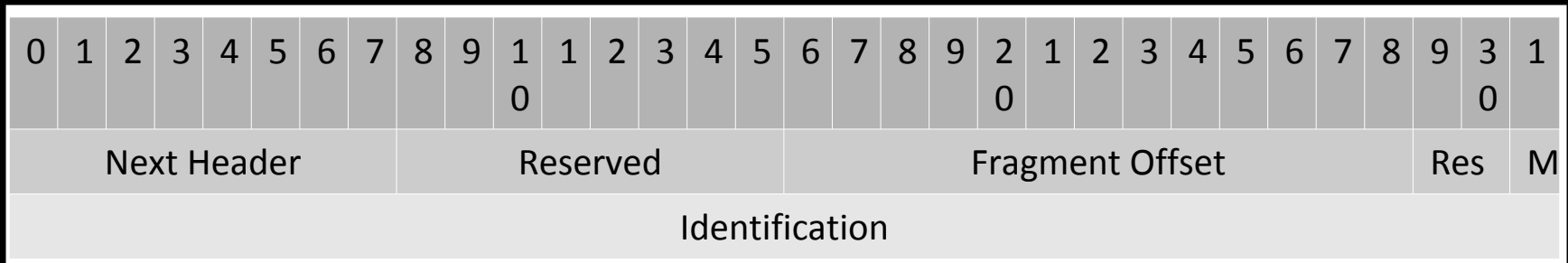
- Fragmentation fields (offset, D and M bits) have been totally removed.
- IPv6 header length is limited to 40 bytes, BUT the use of Extension Headers has been introduced.
- These IPv6 Extension Headers add additional functionality.

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

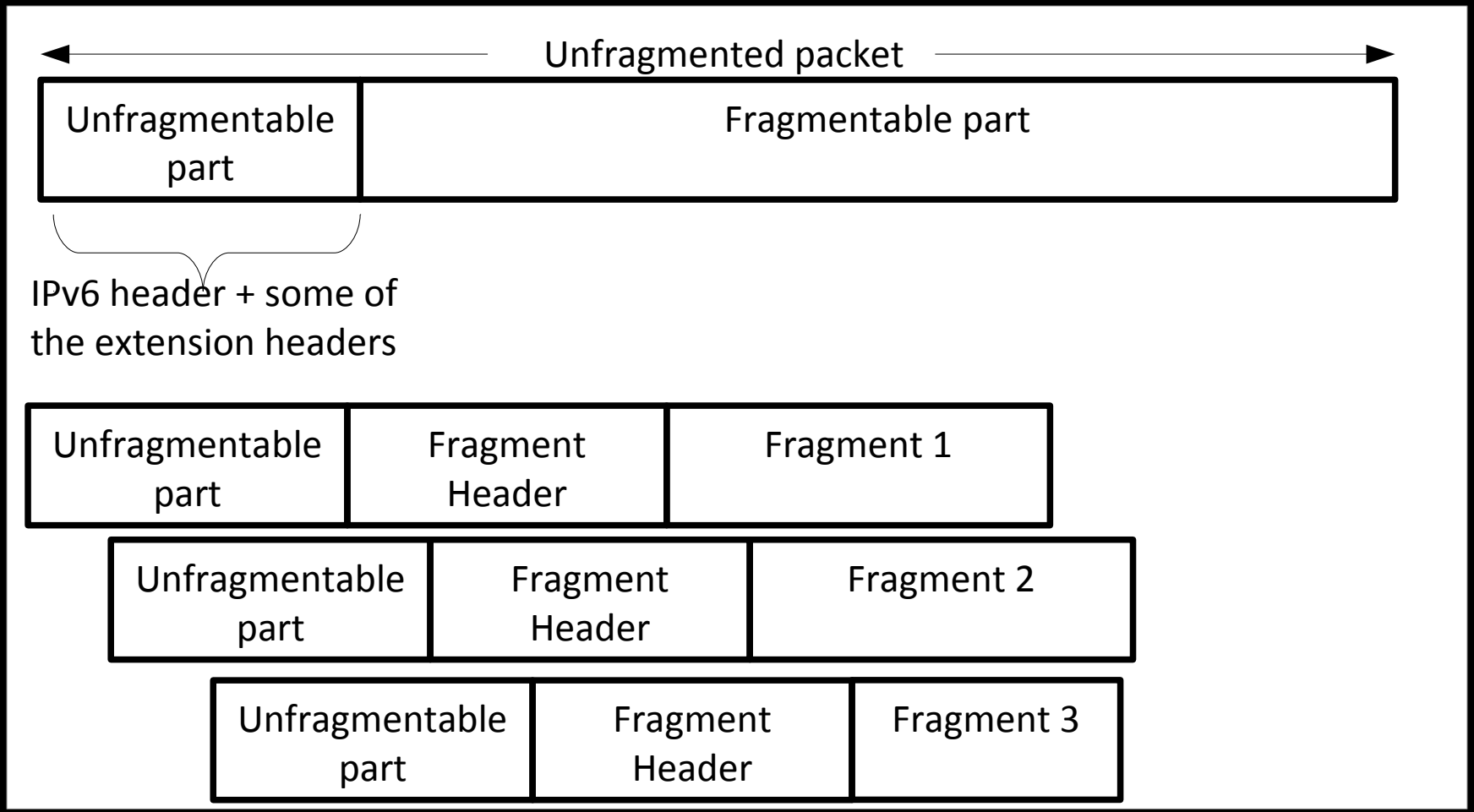
IPv6 Fragment Header



- **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.

Each fragment, except possibly the last one, is an integer multiple of 8 octets long.

IPv6 Fragmentation



Recommended Handling of IPv6 Fragmentation

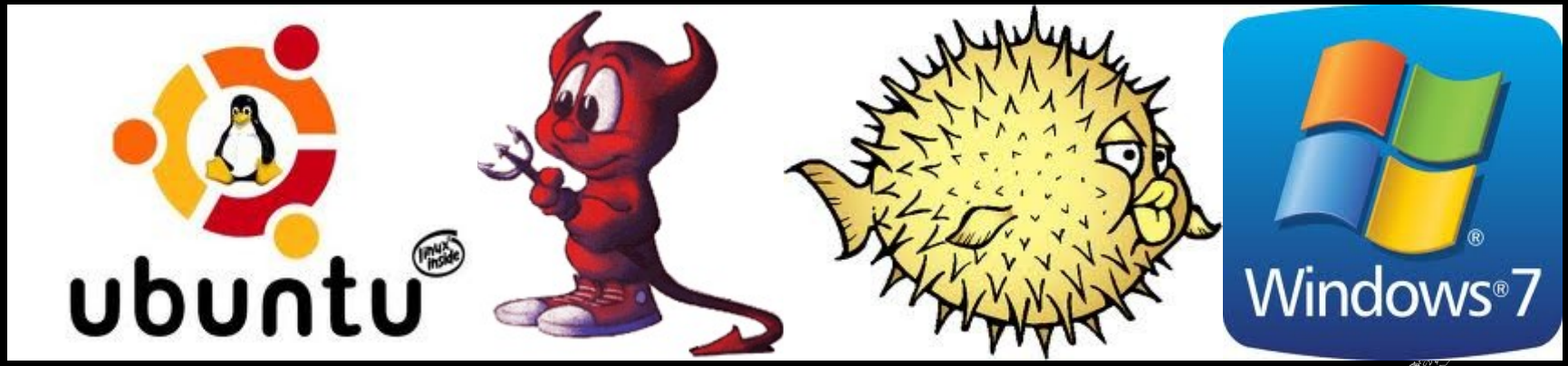
- IPv6 attempts to minimise the use of fragmentation by:
  - Minimising the supported MTU size to 1280 octets or greater. If required, link-specific fragmentation and reassembly must be provided at a layer below IPv6.
  - Allowing only the hosts to fragment datagrams.

Recommended Handling of IPv6 Fragmentation

- 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 fragment, the entire datagram (and any constituent fragments, including those not yet received) must be silently discarded.

Let's play a bit!

Our Targets



Ubuntu 10.04.3 LTS
2.6.32-38 i386
IPv6: fec0::2/64

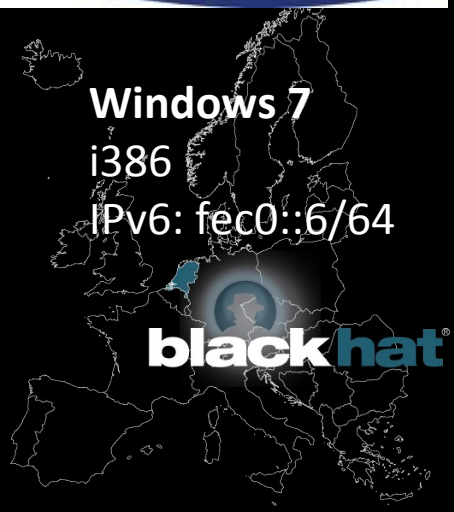
Ubuntu 11.10
3.0.0-15 i386
IPv6: fec0::3/64

FreeBSD 8.2-p3
i386
IPv6: fec0::4/64

FreeBSD 9
amd64
IPv6: fec0::7/64

OpenBSD 5.0
i386
IPv6: fec0::5/64

Windows 7
i386
IPv6: fec0::6/64



Our Attacking Tool

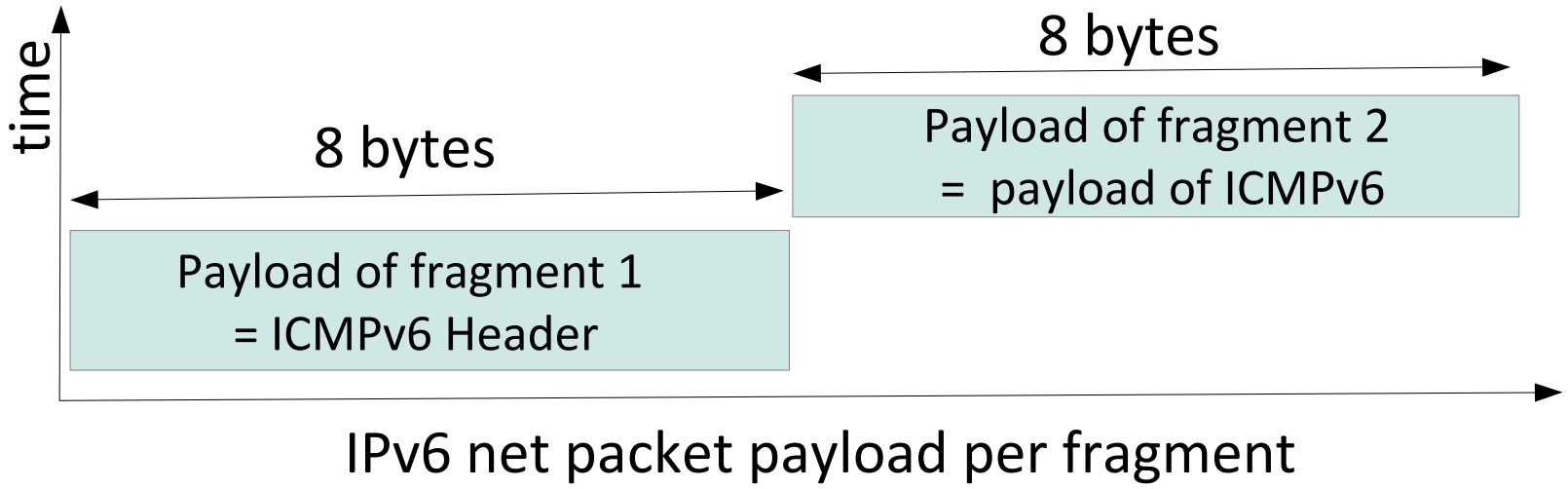
- **Scapy**
  - A powerful interactive packet manipulation program.
  - http://www.secdev.org/projects/scapy/

The Used Protocol for our Testing Purposes

- 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.

Using Tiny Fragmentation (without overlapping)

Using of Small Fragments



The Code

```
#!/usr/bin/python
from scapy.all import *
#IPv6 parameters
sip="fec0::1"
dip="fec0::2"
conf.route6.add("fec0::/64",gw="fec0::1")
payload1="AAAAAAAA"
ipv6_1=IPv6(src=sip, dst=dip, plen=16)
icmpv6=ICMPv6EchoRequest(cksum=0x7d2b)
#Fragment
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

Demo: tiny fragmentation

Results

- All of the tested OS sent an echo reply to the sender.
- Hence, all major OS accept fragments **as small as 56 bytes** (including IPv6 header = 40 bytes IPv6 Header + 8 bytes Fragment Header + 8 bytes of ICMPv6 Header).

So, what's the big deal?

Tiny Fragmentation Consequences

- In IPv4, the embedded protocol's header, e.g. TCP (or at least a part of it) has to be in the 1st fragment.
- Firewall evasions could occur if a subsequent fragment would overwrite the TCP header (e.g. the destination port, the SYN/ACK flags, etc.)
- To this end, RFC 1858 defined 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 264 bytes (RFC 2462).
- Hence, using 8-bytes fragments, we can split the Destination Option headers to 33 fragments!

What does this mean?

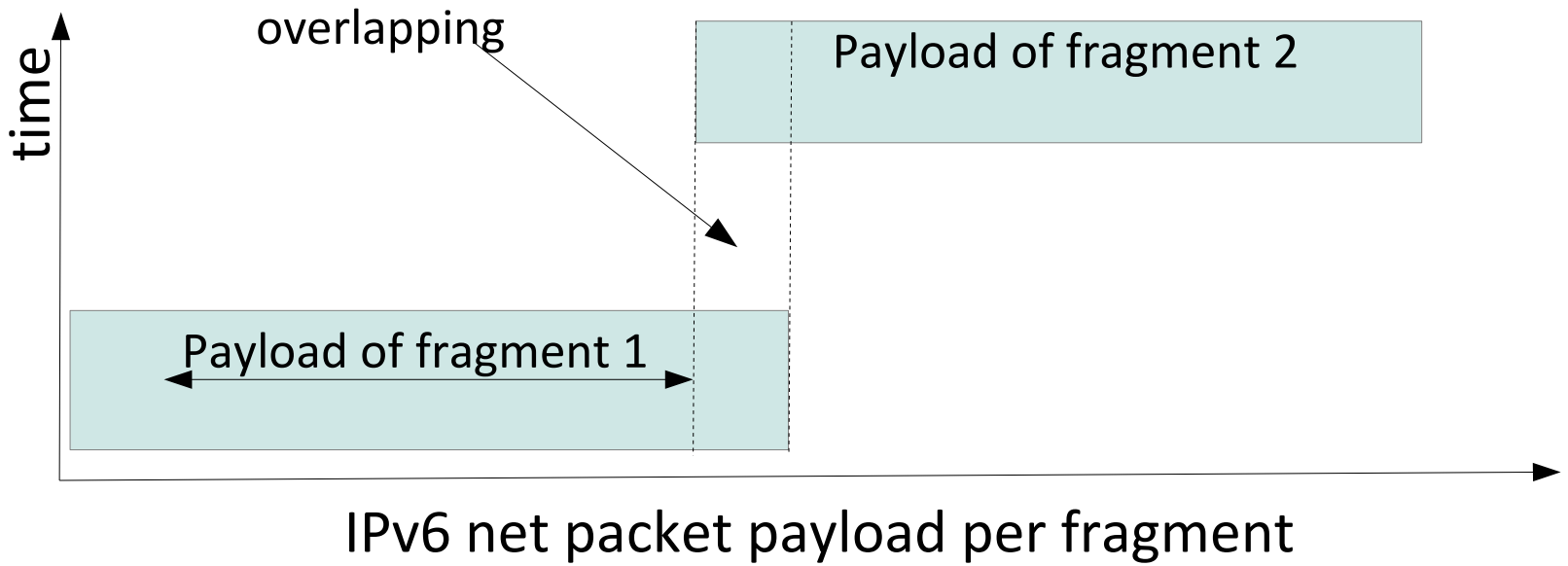
- The layer-4 protocol header will start at the 34th fragment!
- And unless Deep Packet Inspection (= complete IP datagram reassembly before forwarding it), this can lead to firewall evasion, without having to overlap any fragments (as it was the case in IPv4)!

What does this mean?

- This number can increase if we increase the number of the used extension headers that follow the fragment extension header (although not recommended by RFC 2460, but, who cares?).

Creating a very simple fragmentation overlapping

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

Demo: Simple fragmentation overlapping

Results

- FreeBSD, Ubuntu 11.10 and Windows 7 were immune to this attack.
- Ubuntu 10.04 and OpenBSD were susceptible to these attacks.
  - These two OS accept the fragmentation overlapping with the first fragment overwriting the second one.

and so?

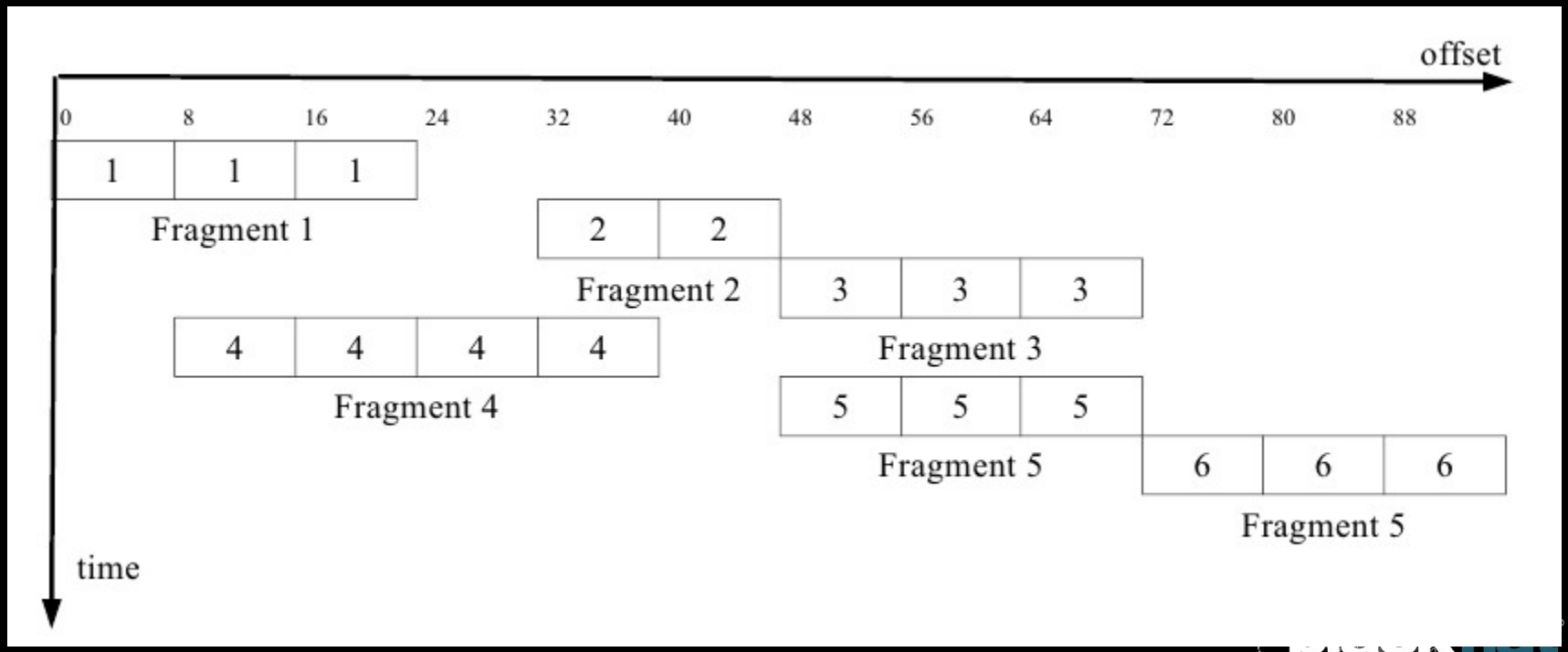
- Acceptance of fragmentation by two of our targets implies that this attack can be used:
  - For OS fingerprinting purposes
  - For IDS Insertion / Evasion purposes (depending for example on whether Ubuntu 10.04 is used as the host OS of the IDS or as a guest OS).
- The fact that the 1st fragment overlaps the second, seems that on its own cannot be exploited for firewall evasion purposes.

The Paxson/Shankar Model

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 this is partially overlapped by a subsequent fragment with an offset less than the original.

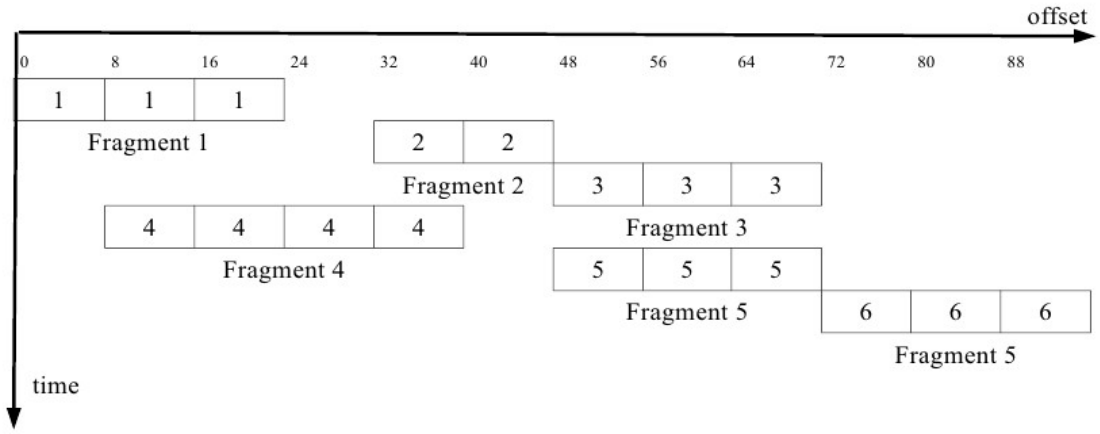
The Paxson/Shankar Model



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



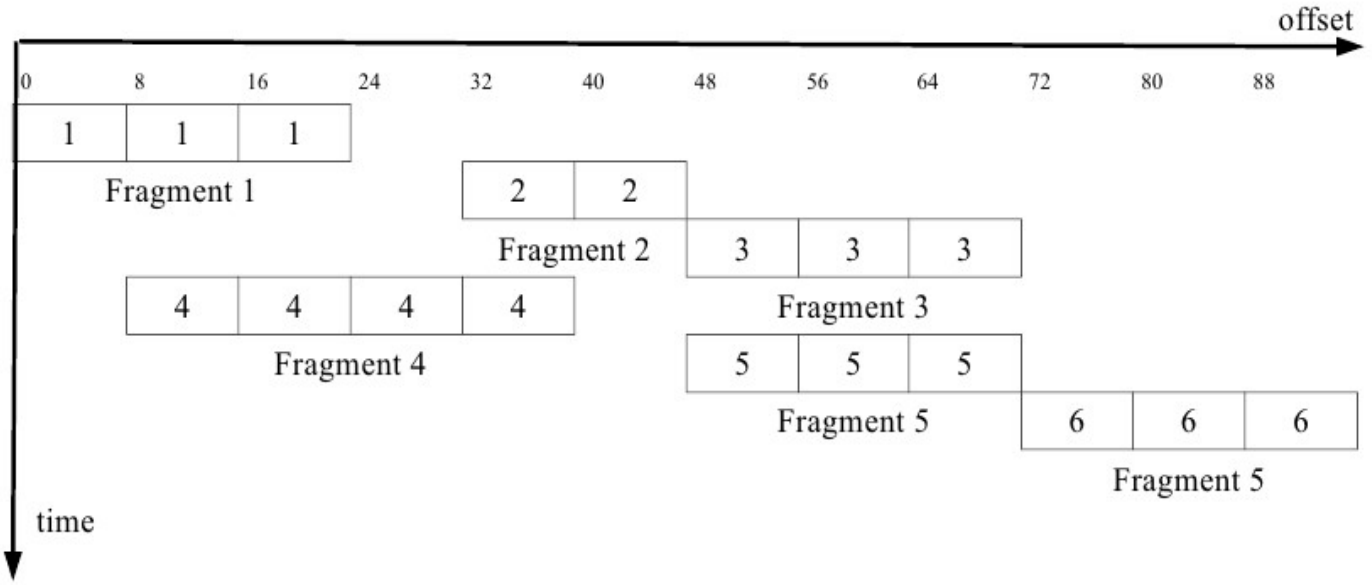
- BSD policy: 111442333666
- BSD-right policy: 144422555666
- Linux policy: 111442555666
- First policy: 111422333666
- Last policy: 144442555666

(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)
```

Demo: The Paxson/Shankar Model

Received ICMPv6 Responses



- Ubuntu 10.04

Frag1	Frag1	Frag1	Frag4	Frag4	Frag2	Frag5	Frag5	Frag5	Frag6	Frag6	Frag6

- OpenBSD 5

Frag1	Frag1	Frag1	Frag4	Frag4	Frag2	Frag3	Frag3	Frag3	Frag6	Frag6	Frag6

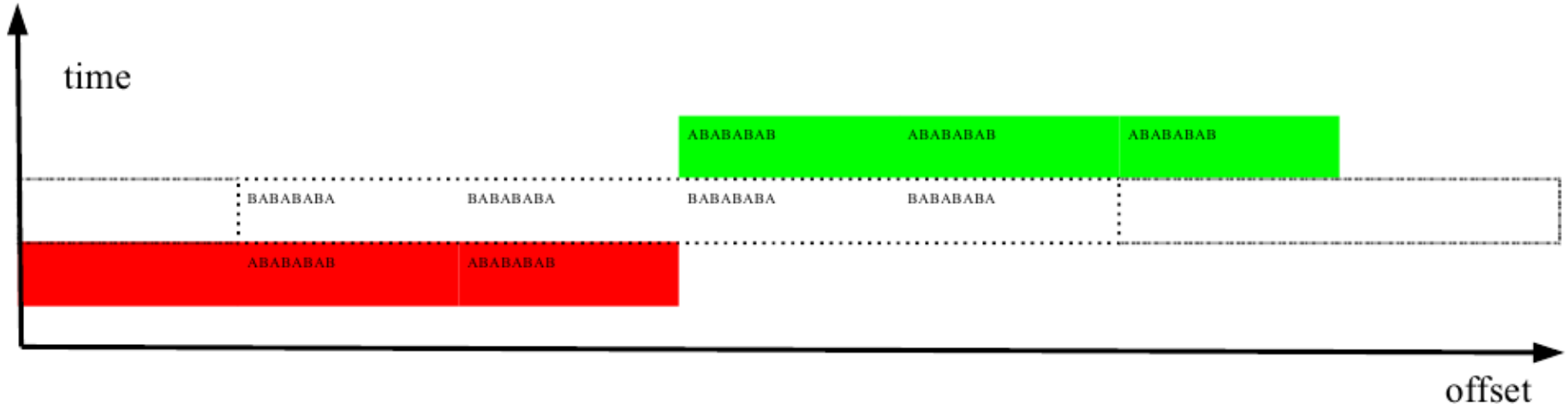
Results

- FreeBSD, Windows 7 and Ubuntu 11.10 are immune to these attacks.
- Ubuntu 10.04 and OpenBSD are susceptible to these attacks.
  - OpenBSD: BSD reassembly policy
  - Ubuntu 10.04: Linux reassembly policy

So, up to now it seems that linux kernel 2.6.40, FreeBSD 8.2/9 and Windows 7 are immune to fragmentation overlapping attacks, right?

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

A simple 3-packet model

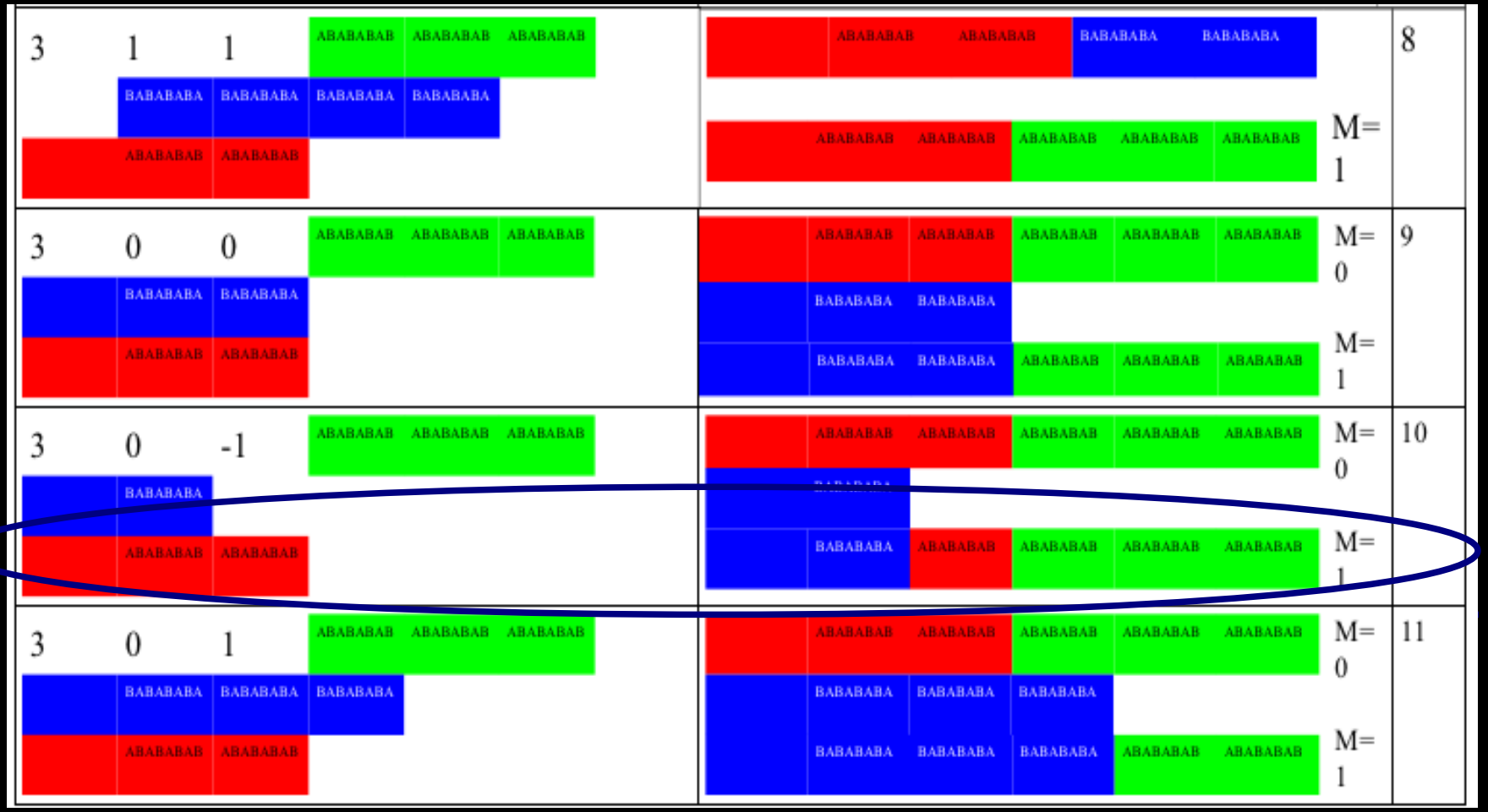


Brief summary of Ubuntu 10.04 responses

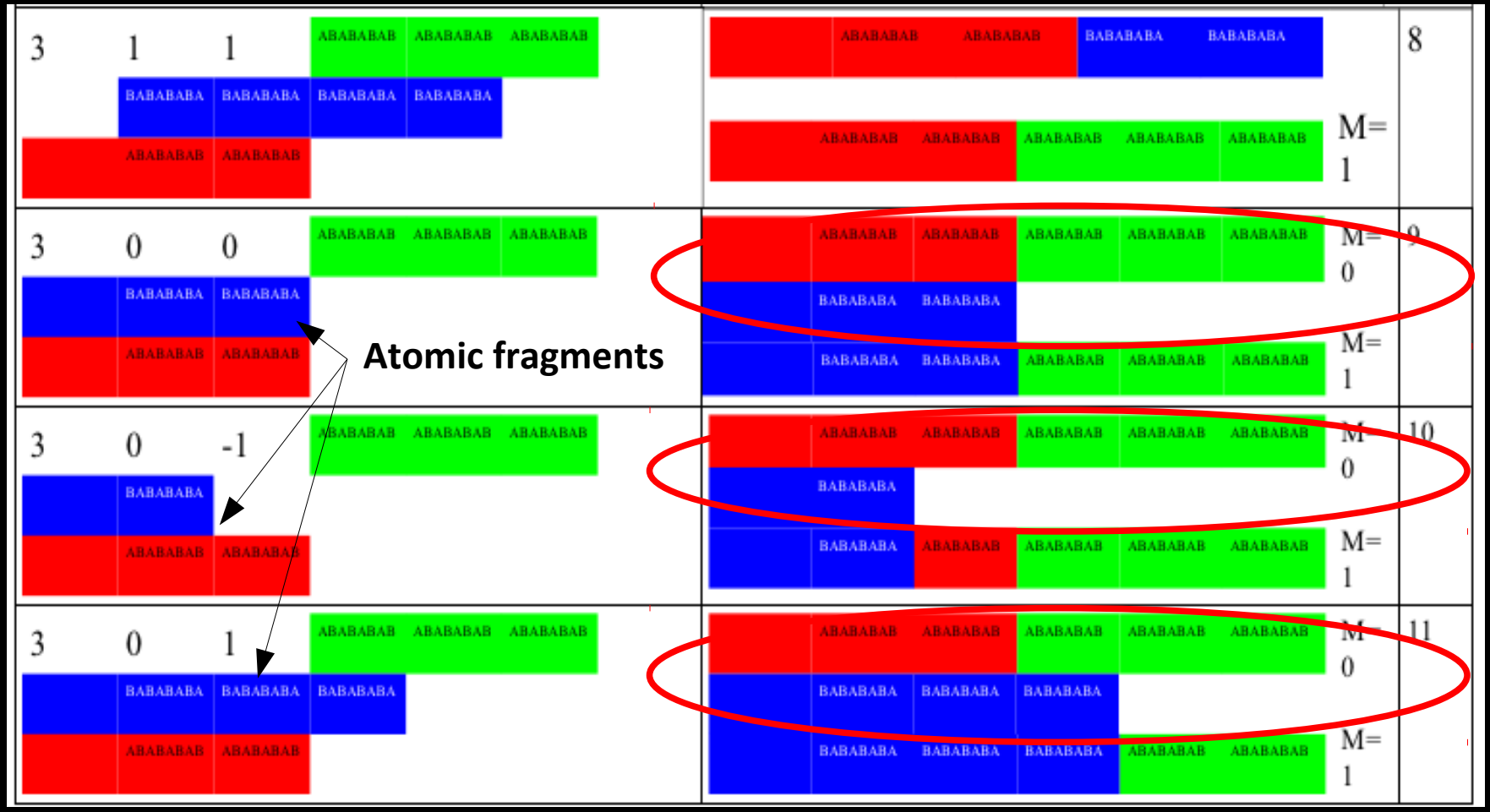
- The non-favoured packets are not discarded completely but they trimmed.
- The Linux reassembly policy was confirmed with one exception (when the 2nd fragment has a 0 offset and M=1).
- Three notable behaviours are when atomic fragments overlap with other. In these cases we have two separate responses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

Sample of Ubuntu 10.04 Responses

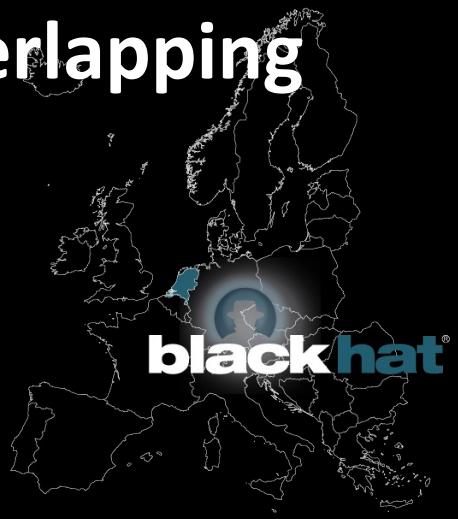
Sample of Ubuntu 10.04 Responses



Sample of Ubuntu 10.04 Responses



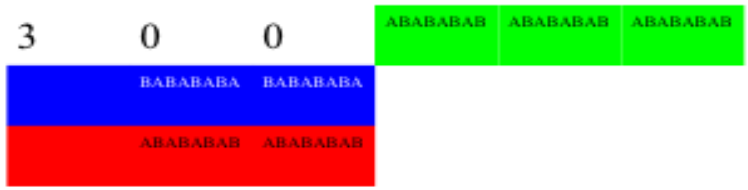
Demo:
Two responses from Ubuntu 10.04 in case of atomic fragments overlapping with others



Brief summary of OpenBSD 5 responses

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

Sample of FreeBSD Responses

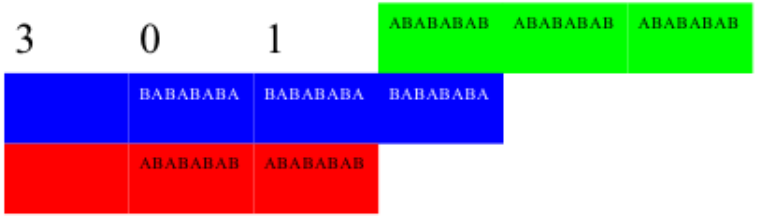
3 2 2			M=0 M=1	7
3 1 1			M=0 M=1	8
3 0 0			M=0 M=1	9
3 0 -1			M=0 M=1	10
3 0 1			M=0 M=1	11

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 are accepted (which do not overlap).

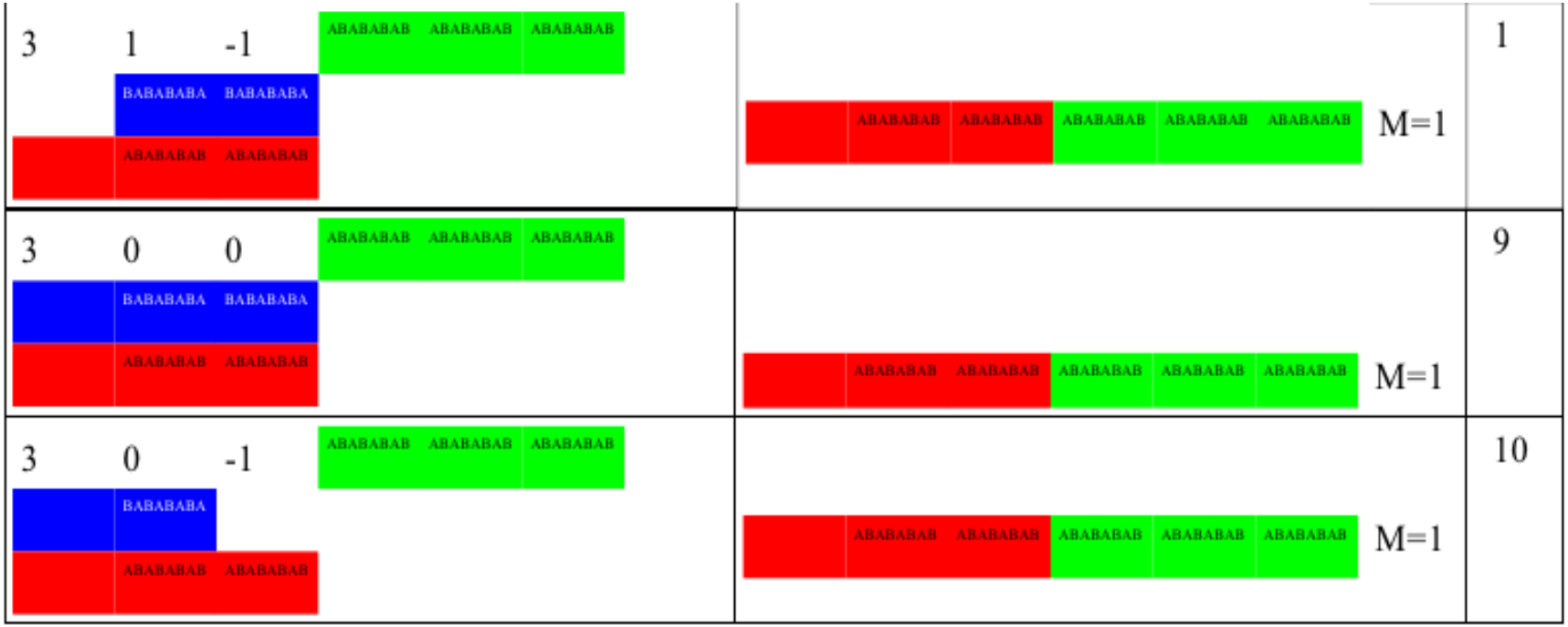
Ubuntu 11.10 Responses

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

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

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 offset.



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 overlapped.

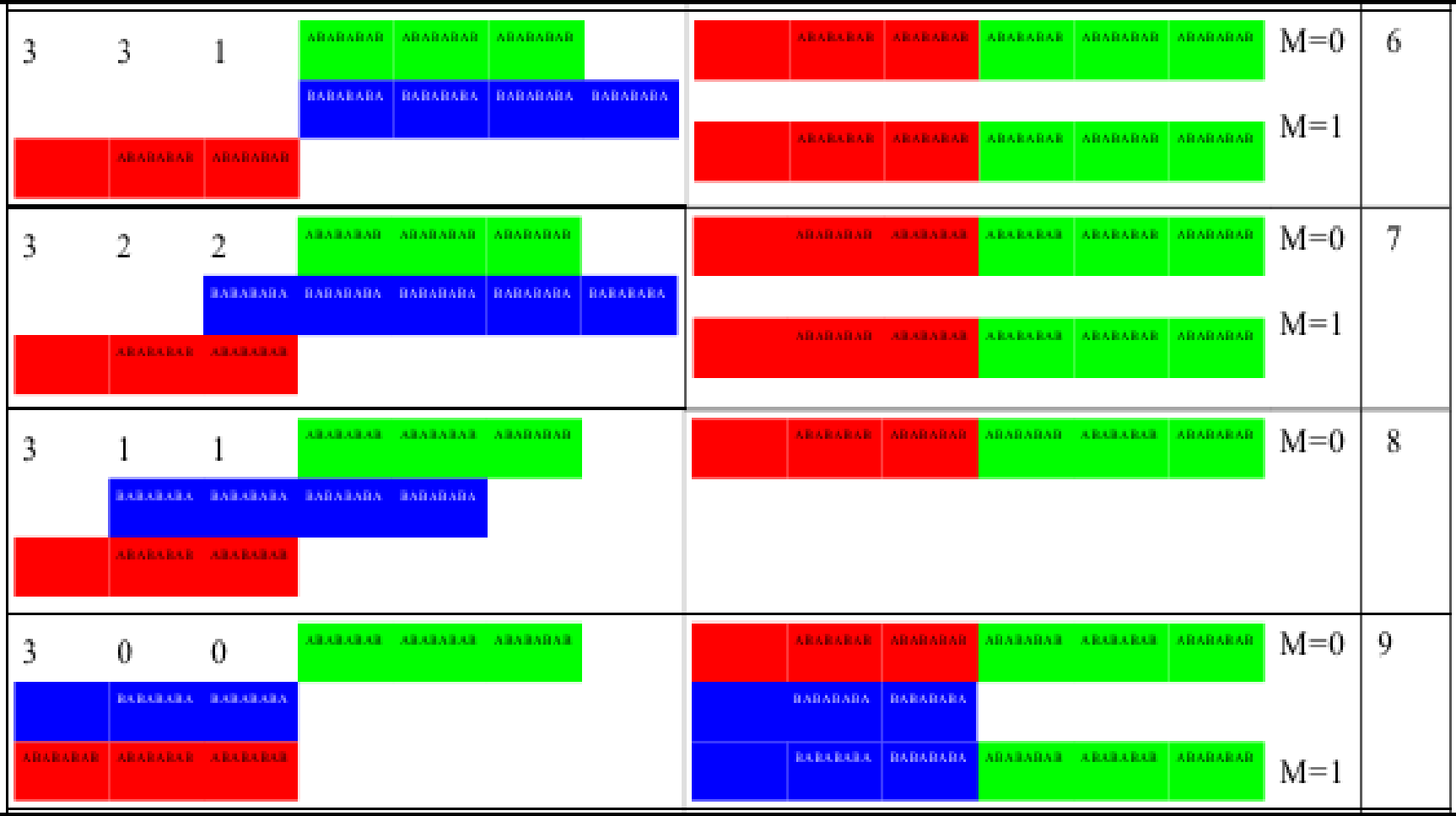
Demo:
Ubuntu 11.10 and Windows 7 testing

Reversing the sending order of the fragments

Ubuntu 11.10 responses for reverse sending order

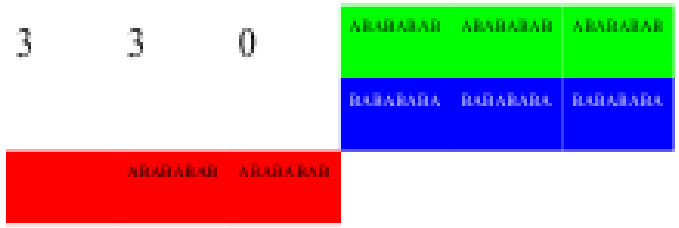
- More responses are received than when the normal sending order is used.
  - When atomic fragments overlap with non-atomic ones.
  - In most of the other cases, only the overlapping fragment is discarded.

Sample of Ubuntu 11.10 Responses when reversing the order



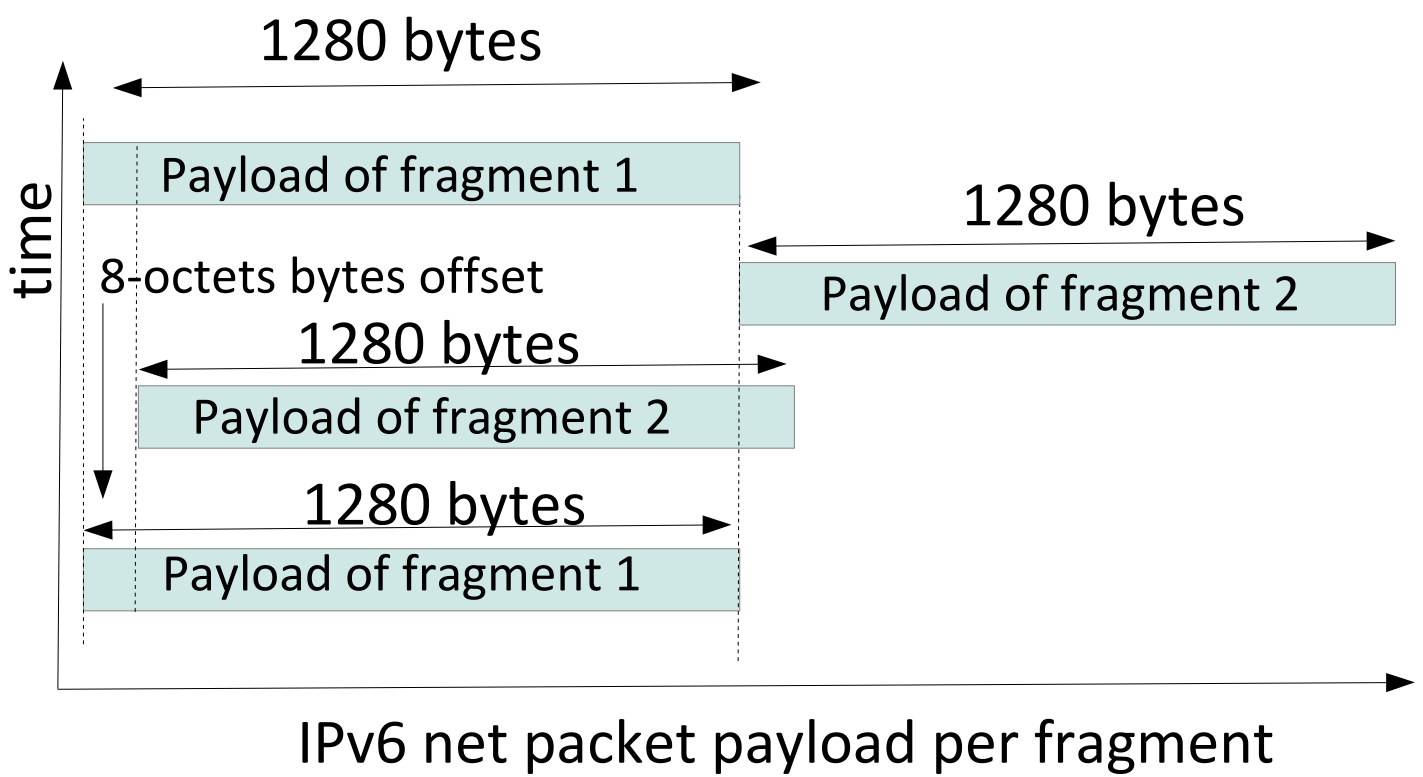
Windows 7 Responses when reversing the order

- Responses when fragments 2 and 3 completely and exactly overlap, in which case Windows 7 considering them probably as repeated packets.

Table 2.B Accepted overlapping results of Windows 7 for a reverse arrival order		Case
3 3 0 	M=0 M=1	3

Some final tests

Fragmentation Overlapping Sending Double Packets



(part of) the code

```
ipv6_1=IPv6(src=sip, dst=dip, plen=1288)
icmpv6=ICMPv6EchoRequest(cksum=0xb47b, data=payload1)#fec0::3
#Fragment
frag1=IPv6ExtHdrFragment(offset=0, m=1, id=712, nh=58)
frag2=IPv6ExtHdrFragment(offset=8, m=0, id=712, nh=58)
frag3=IPv6ExtHdrFragment(offset=160, m=0, id=712, nh=58)
packet1=ipv6_1/frag1/icmpv6
packet2=ipv6_1/frag2/payload2
packet3=ipv6_1/frag3/payload2
send(packet1)
send(packet2)
send(packet3)
send(packet1)
```

Results

- Ubuntu 10.04 and OpenBSD 5 send two responses back.
- The two FreeBSDs send back a response even if the packet numbered 4 is not sent, showing again that they just discard the overlapping fragment.
- Ubuntu 11.10 and Windows 7 do send a response only if all the four packets are sent (including the last one, with the 0 offset).
- If the packet numbered 1 is not sent, none of the three sends back a response.

Demo:
Sending double overlapping packets

Conclusions

Conclusions (1/5)

- 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.
- None of the tested OS is RFC 5722 compliant.

Conclusions (2/5)

- **Ubuntu 10.04 LTS** (using linux kernel 2.6.32) and **OpenBSD 5** were proven the most susceptible to fragmentation overlapping attacks among the tested OS, each one following the corresponding well-known reassembly policies (Linux and BSD respectively).

Conclusions (3/5)

- **FreeBSD 8.2/9** discards any overlapping fragments appearing to have the most consistent behaviour.
- Although this is a very good practice, it does not fully comply with RFC 5722 which suggest the rejection of any constituent fragments too (including the ones not yet received).

Conclusions (4/5)

- The two **Ubuntu** send two responses back when ***atomic*** fragments overlap with non-atomic ones.
- The behaviour of **Ubuntu 11.10** seems to deteriorate significantly when the sending order of the fragments is reversed.
- **Windows 7**, although seem to have the fewer issues, there are cases that they also accept overlapping fragments.

Conclusions (5/5)

- **The impact of these issues**, since it varies between the tested OS, starts from OS fingerprinting and can be extended, if used properly, to IDS insertion / evasion and in some cases, even to firewall evasions.
- OS vendors need to create fully RFC compliant products.

Please complete the speakers' feedback survey forms.

Thank you!

antonios.atlasis@cscss.org