Faculté des Sciences Appliquées

Tracking Middleboxes with Tracebox

IETF93: HOPS

Korian Edeline, Benoit Donnet
University of Liège

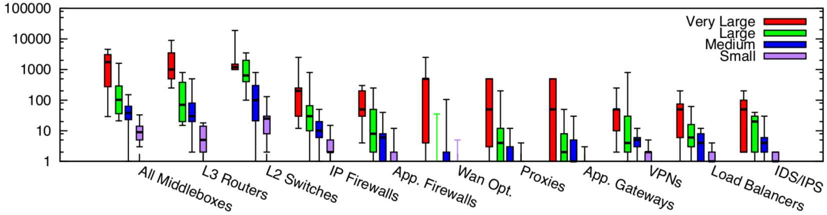
mPlane

Introduction

1. Middleboxes
2. How to detect them ?
3. Tracebox
4. Implementations

Plan

Deployment

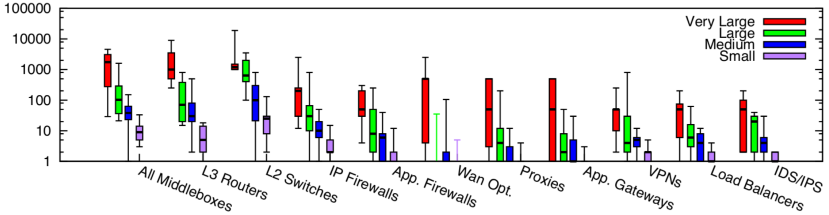


Box plot of middlebox deployments for small (fewer than 1k hosts), medium (1k-10k hosts), large (10k-100k hosts), and very large (more than 100k hosts) enterprise networks. Y-axis is in log scale. [1]

[1] Justine Sherry et al. “Making middleboxes someone else’s problem: network processing as a cloud service”. In: *ACM SIGCOMM Computer Communication Review* 42.4 (2012), pp. 13–24.

[2] Rahul Potharaju and Navendu Jain. “Demystifying the dark side of the middle: A field study of middlebox failures in datacenters”. In: *Proceedings of the 2013 conference on Internet measurement conference*. ACM. 2013, pp. 9–22.

Deployment



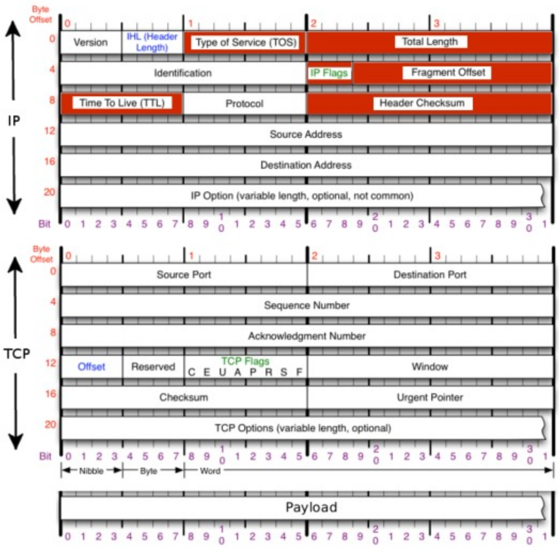
Box plot of middlebox deployments for small (fewer than 1k hosts), medium (1k-10k hosts), large (10k-100k hosts), and very large (more than 100k hosts) enterprise networks. Y-axis is in log scale. [1]

- The market for security-oriented middleboxes is estimated to exceed \$10B by 2016[2]

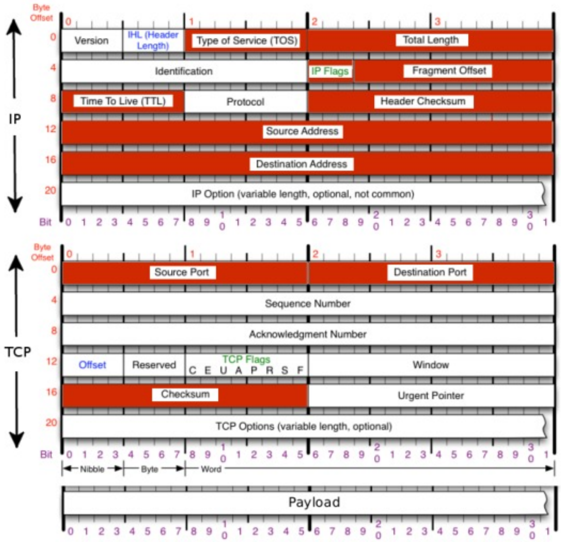
[1] Justine Sherry et al. "Making middleboxes someone else's problem: network processing as a cloud service". In: *ACM SIGCOMM Computer Communication Review* 42.4 (2012), pp. 13–24.

[2] Rahul Potharaju and Navendu Jain. "Demystifying the dark side of the middle: A field study of middlebox failures in datacenters". In: *Proceedings of the 2013 conference on Internet measurement conference*. ACM. 2013, pp. 9–22.

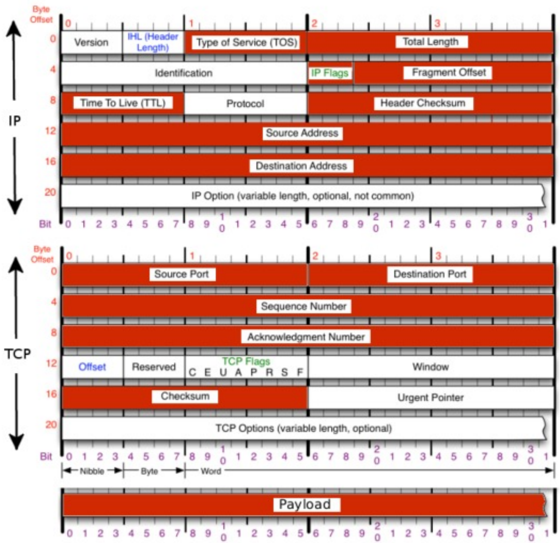
Router processing



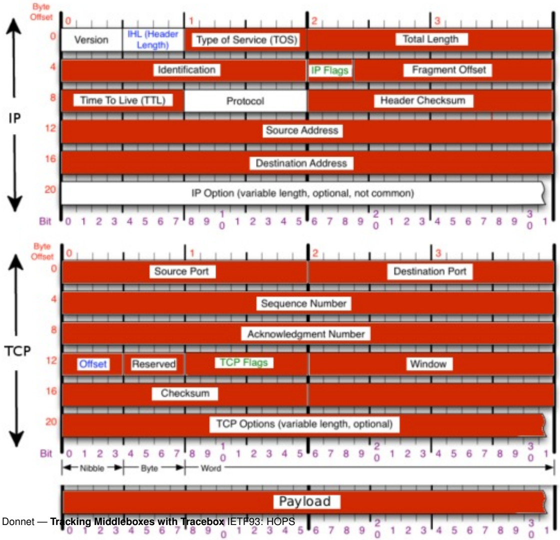
NAT processing



ALG processing



Potential processing over the whole Internet



Plan

1. Middleboxes
2. How to detect them ?
3. Tracebox
4. Implementations

TBIT

- `tbit`[3]

[3] Alberto Medina, Mark Allman, and Sally Floyd. “Measuring interactions between transport protocols and middleboxes”. In: *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*. ACM. 2004, pp. 336–341.

TBIT

- `tbit`[3]
- Basic idea
  - Send forged TCP packets over raw IP sockets

[3] Alberto Medina, Mark Allman, and Sally Floyd. “Measuring interactions between transport protocols and middleboxes”. In: *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*. ACM. 2004, pp. 336–341.

TBIT

- `tbit`[3]
- Basic idea
  - Send forged TCP packets over raw IP sockets
  - Host firewall prevents kernel from seeing response packets

[3] Alberto Medina, Mark Allman, and Sally Floyd. “Measuring interactions between transport protocols and middleboxes”. In: *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*. ACM. 2004, pp. 336–341.

TBIT

- `tbit`[3]
- Basic idea
  - Send forged TCP packets over raw IP sockets
  - Host firewall prevents kernel from seeing response packets
  - BPF delivers blocked packets to user process for analysis

[3] Alberto Medina, Mark Allman, and Sally Floyd. "Measuring interactions between transport protocols and middleboxes". In: *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*. ACM. 2004, pp. 336–341.

TBIT

- `tbit`[3]
- Basic idea
  - Send forged TCP packets over raw IP sockets
  - Host firewall prevents kernel from seeing response packets
  - BPF delivers blocked packets to user process for analysis
- Effect
  - a user-level, user-controllable TCP, without kernel changes

[3] Alberto Medina, Mark Allman, and Sally Floyd. "Measuring interactions between transport protocols and middleboxes". In: *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*. ACM. 2004, pp. 336–341.

TBIT

- tbit[3]
- Basic idea
  - Send forged TCP packets over raw IP sockets
  - Host firewall prevents kernel from seeing response packets
  - BPF delivers blocked packets to user process for analysis
- Effect
  - a user-level, user-controllable TCP, without kernel changes
- Purpose
  - detect whether ECN, IP options, and TCP options can be safely used

[3] Alberto Medina, Mark Allman, and Sally Floyd. “Measuring interactions between transport protocols and middleboxes”. In: *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*. ACM. 2004, pp. 336–341.

TCPExposure

- `TCPExposure`[4]

[4] Michio Honda et al. “Is it still possible to extend TCP?”. In: *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. ACM. 2011, pp. 181–194.

TCPExposure

- `TCPExposure`[4]
- Basic idea
  - Client and Server Python scripts

[4] Michio Honda et al. "Is it still possible to extend TCP?". In: *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. ACM. 2011, pp. 181–194.

TCPExposure

- `TCPExposure`[4]
- Basic idea
  - Client and Server Python scripts
  - Send forged TCP packets over raw IP sockets

[4] Michio Honda et al. “Is it still possible to extend TCP?”. In: *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. ACM. 2011, pp. 181–194.

TCPExposure

- `TCPExposure`[4]
- Basic idea
  - Client and Server Python scripts
  - Send forged TCP packets over raw IP sockets
  - Sent packets include payload commands bytes: *just ack*, *echo headers* or *don't advance ack*

[4] Michio Honda et al. “Is it still possible to extend TCP?”. In: *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. ACM. 2011, pp. 181–194.

TCPExposure

- `TCPExposure`[4]
- Basic idea
  - Client and Server Python scripts
  - Send forged TCP packets over raw IP sockets
  - Sent packets include payload commands bytes: *just ack*, *echo headers* or *don't advance ack*
  - Server sends back received&to-be-sent headers as payload

[4] Michio Honda et al. "Is it still possible to extend TCP?". In: *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. ACM. 2011, pp. 181–194.

TCPExposure

- `TCPExposure`[4]
- Basic idea
  - Client and Server Python scripts
  - Send forged TCP packets over raw IP sockets
  - Sent packets include payload commands bytes: *just ack*, *echo headers* or *don't advance ack*
  - Server sends back received&to-be-sent headers as payload
  - Compare what was sent to what was received

[4] Michio Honda et al. "Is it still possible to extend TCP?". In: *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. ACM. 2011, pp. 181–194.

TCPExposure

- `TCPExposure`[4]
- Basic idea
  - Client and Server Python scripts
  - Send forged TCP packets over raw IP sockets
  - Sent packets include payload commands bytes: *just ack*, *echo headers* or *don't advance ack*
  - Server sends back received&to-be-sent headers as payload
  - Compare what was sent to what was received
- Effect
  - Detect last modification, errors
  - Differentiate inbound & outbound modifications

[4] Michio Honda et al. "Is it still possible to extend TCP?". In: *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. ACM. 2011, pp. 181–194.

TCP HICCUPS

- TCP HICCUPS[5]

[5] Ryan Craven, Robert Beverly, and Mark Allman. “A middlebox-cooperative TCP for a non end-to-end internet”. In: *Proceedings of the 2014 ACM conference on SIGCOMM*. ACM. 2014, pp. 151–162.

TCP HICCUPS

- `TCP HICCUPS`[5]
- Lightweight TCP extension that exposes in flight packet header modification to end points

[5]Ryan Craven, Robert Beverly, and Mark Allman. “A middlebox-cooperative TCP for a non end-to-end internet”. In: *Proceedings of the 2014 ACM conference on SIGCOMM*. ACM. 2014, pp. 151–162.

TCP HICCUPS

- TCP HICCUPS[5]
- Lightweight TCP extension that exposes in flight packet header modification to end points
- Basic question behind TCP HICCUPS
  - did my packet arrive at the destination with the same headers as sent?

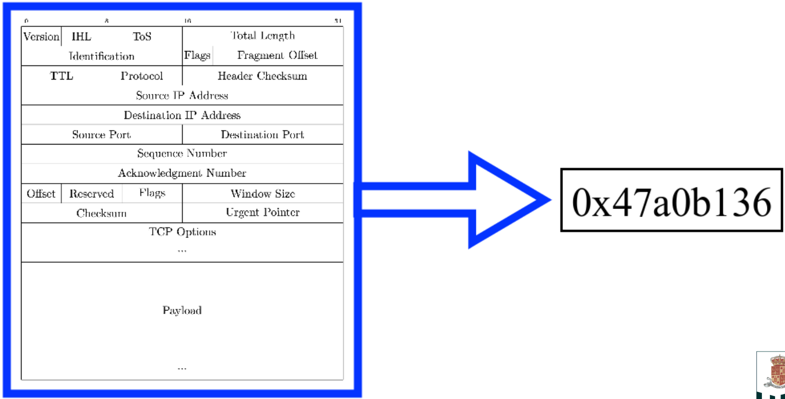
[5] Ryan Craven, Robert Beverly, and Mark Allman. “A middlebox-cooperative TCP for a non end-to-end internet”. In: *Proceedings of the 2014 ACM conference on SIGCOMM*. ACM. 2014, pp. 151–162.

TCP HICCUPS

- HICCUPS overloads 3 header fields in the TCP 3-way handshake
  - ISN, IPID, RWIN

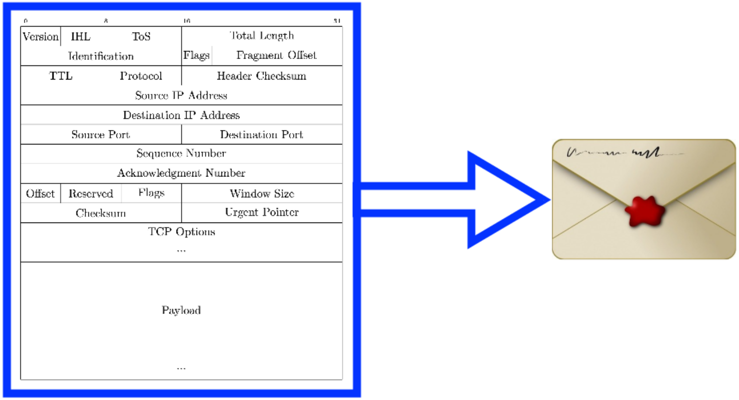
TCP HICCUPS

- HICCUPS overloads 3 header fields in the TCP 3-way handshake
  - ISN, IPID, RWIN
- ... with a function of the packet header



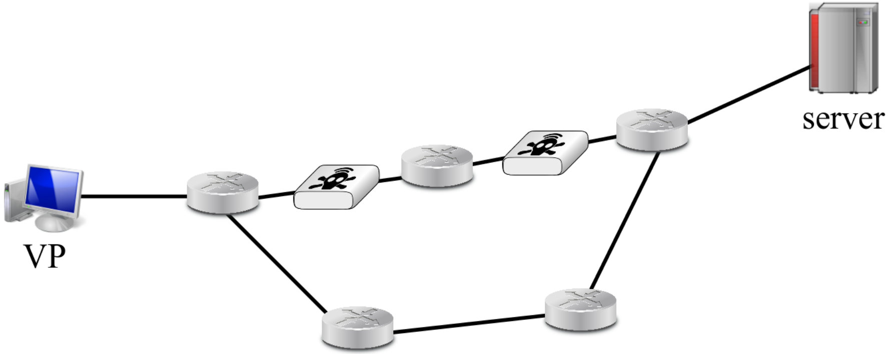
TCP HICCUPS

- All in all, it creates an end-to-end tamper-evident seal over the packet headers
- Different than a checksum
  - if some mods occur, the packet is still accepted



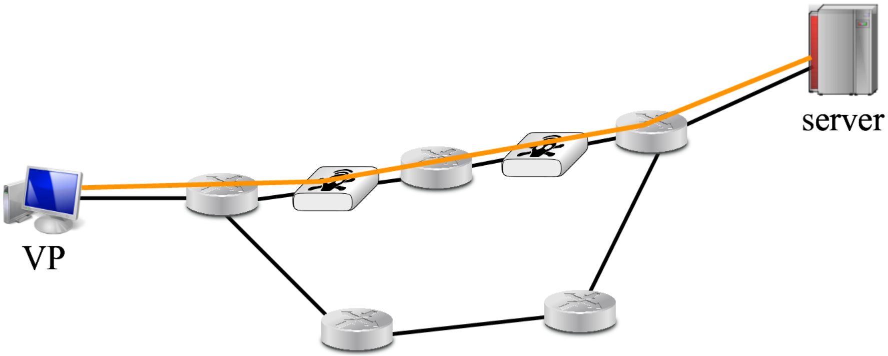
Controlling both ends

- Controlling both ends allows to detect middleboxes on one path



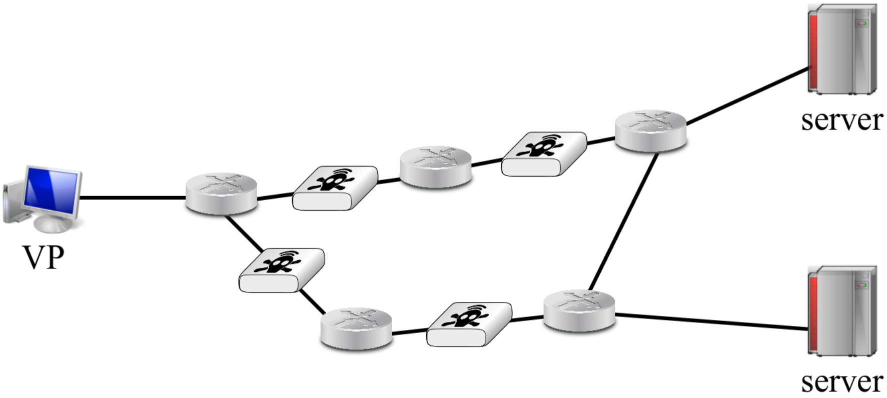
Controlling both ends

- Controlling both ends allows to detect middleboxes on one path



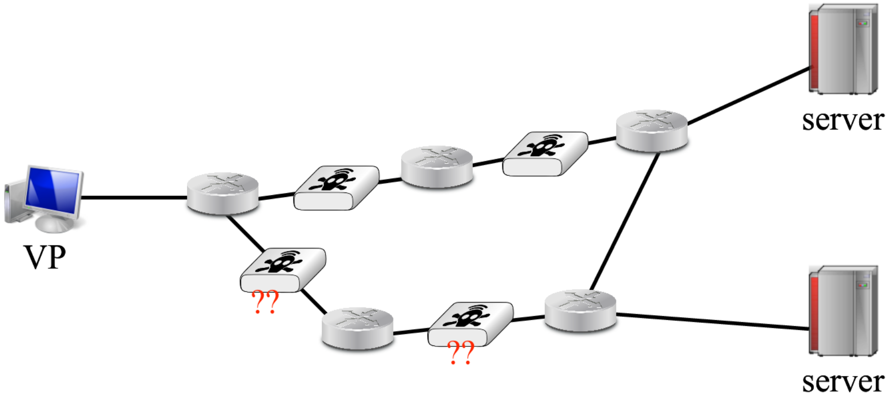
Controlling both ends

- What happens with uncontrolled server(s)?
  - potentially miss a lot of middleboxes



Controlling both ends

- What happens with uncontrolled server(s)?
  - potentially miss a lot of middleboxes



Tracebox

- Tracebox[6]
- Extension to traceroute
  - send TTL limited TCP probes
  - inspect incoming ICMP time-exceeded packets
  - compare the TCP probe quoted and the TCP probe sent
  - in case of difference(s), a middlebox is found along the path

[6] Gregory Detal et al. "Revealing middlebox interference with tracebox". In: *Proceed of the 2013 conference on Internet measurement conference.* ACM. 2013, pp. 1–8.

Tracebox

- Tracebox[6]
- Extension to traceroute
  - send TTL limited TCP probes
  - inspect incoming ICMP time-exceeded packets
  - compare the TCP probe quoted and the TCP probe sent
  - in case of difference(s), a middlebox is found along the path
- Server-independant, "One-sided"
- Detect multiple modifications

[6] Gregory Detal et al. “Revealing middlebox interference with tracebox”. In: *Proceed of the 2013 conference on Internet measurement conference.* ACM. 2013, pp. 1–8.

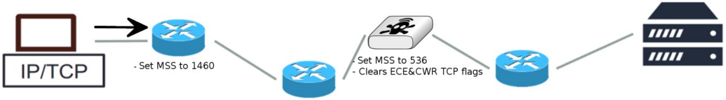
Tracebox

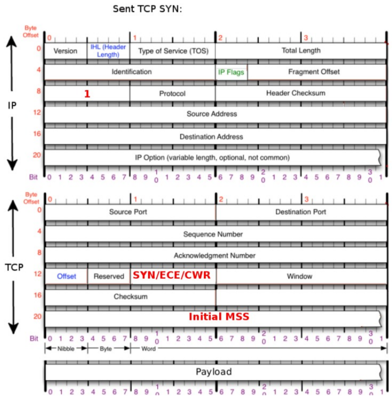
- Tracebox[6]
- Extension to traceroute
  - send TTL limited TCP probes
  - inspect incoming ICMP time-exceeded packets
  - compare the TCP probe quoted and the TCP probe sent
  - in case of difference(s), a middlebox is found along the path
- Server-independant, "One-sided"
- Detect multiple modifications
- Purpose
  - Middlebox detection
  - Middlebox location

[6]Gregory Detal et al. "Revealing middlebox interference with tracebox". In: *Proceed of the 2013 conference on Internet measurement conference.* ACM. 2013, pp. 1–8.

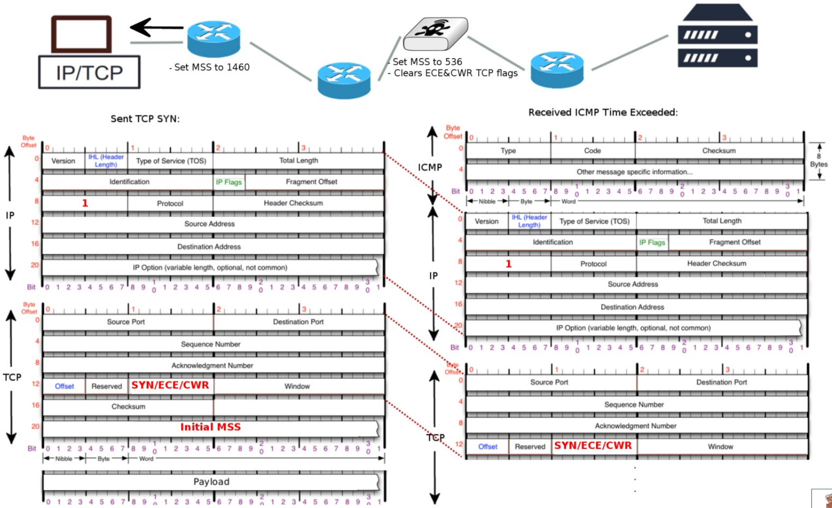
Plan

Tracebox

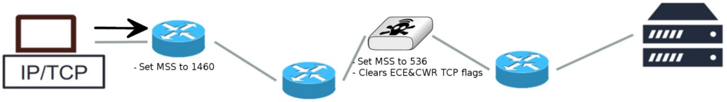

IP/TCP
- Set MSS to 1460
- Set MSS to 536
- Clears ECE&CWR TCP flags


Sent TCP SYN:
IP
Version
IHL (Header Length)
Type of Service (TOS)
Total Length
Identification
IP Flags
Fragment Offset
1
Protocol
Header Checksum
Source Address
Destination Address
IP Option (variable length, optional, not common)
TCP
Source Port
Destination Port
Sequence Number
Acknowledgment Number
Offset
Reserved
SYN/ECE/CWR
Window
Checksum
Initial MSS
Nibble
Byte
Word
Payload

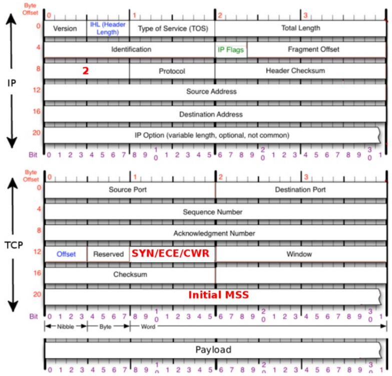
Tracebox


IP/TCP
- Set MSS to 1460
- Set MSS to 536
- Clears ECE&CWR TCP flags
Sent TCP SYN:
Received ICMP Time Exceeded:
IP
TCP
ICMP
Version
IHL (Header Length)
Type of Service (TOS)
Total Length
Identification
IP Flags
Fragment Offset
1
Protocol
Header Checksum
Source Address
Destination Address
IP Option (variable length, optional, not common)
Source Port
Destination Port
Sequence Number
Acknowledgment Number
Offset
Reserved
SYN/ECE/CWR
Window
Checksum
Initial MSS
Payload
Type
Code
Checksum
Other message specific information...
8 Bytes
Byte Offset
Bit
Nibble
Byte
Word

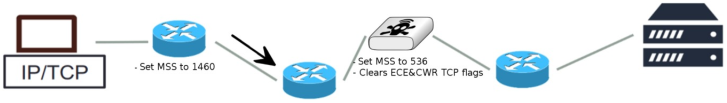
Tracebox



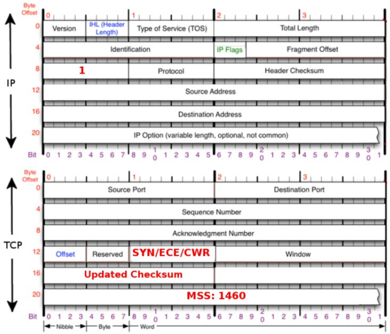
Sent TCP SYN:



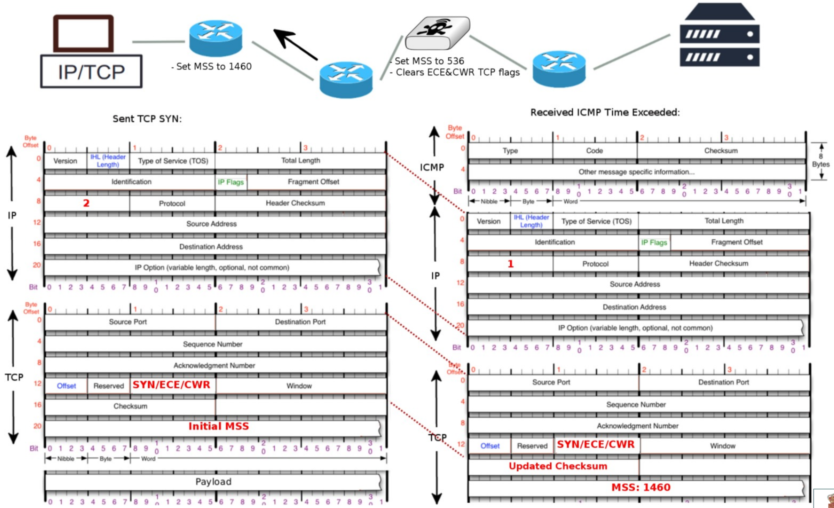
Tracebox



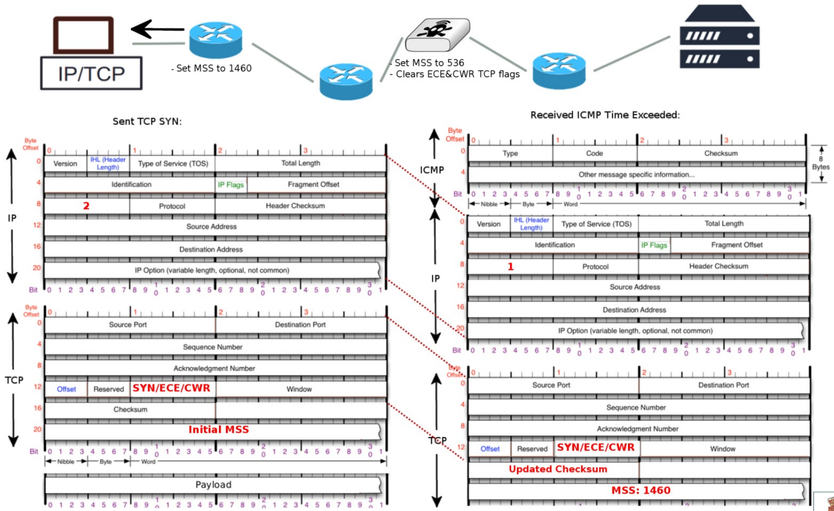
Sent TCP SYN:



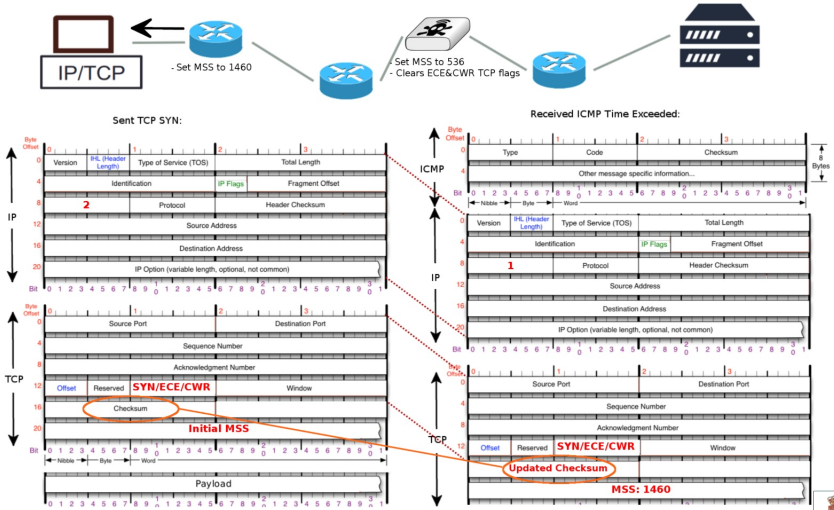
Tracebox


IP/TCP
- Set MSS to 1460
- Set MSS to 536
- Clears ECE&CWR TCP flags
Sent TCP SYN:
Received ICMP Time Exceeded:
IP
TCP
ICMP
2
1
SYN/ECE/CWR
Initial MSS
Payload
Updated Checksum
MSS: 1460
Version
IHL (Header Length)
Type of Service (TOS)
Total Length
Identification
IP Flags
Fragment Offset
Protocol
Header Checksum
Source Address
Destination Address
IP Option (variable length, optional, not common)
Source Port
Destination Port
Sequence Number
Acknowledgment Number
Offset
Reserved
Window
Checksum
Type
Code
Other message specific information...
8 Bytes

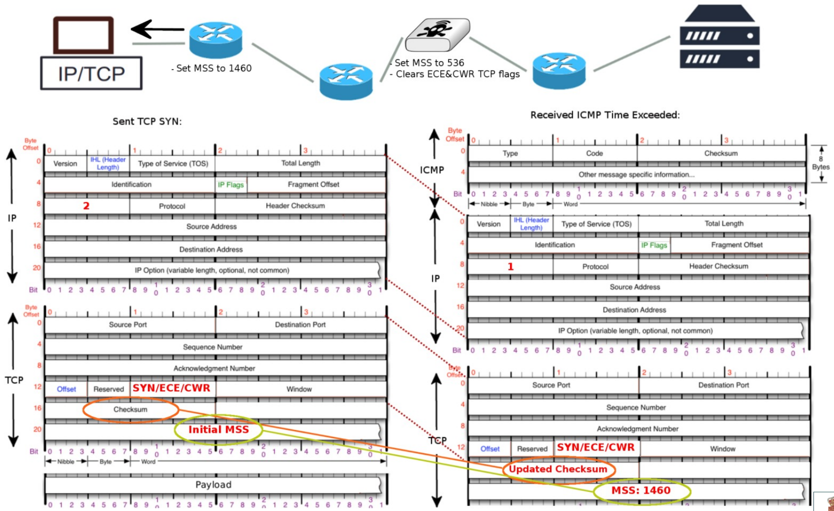
Tracebox


IP/TCP
- Set MSS to 1460
- Set MSS to 536
- Clears ECE&CWR TCP flags
Sent TCP SYN:
Received ICMP Time Exceeded:
IP
TCP
ICMP
2
1
SYN/ECE/CWR
Initial MSS
Payload
Updated Checksum
MSS: 1460
8 Bytes

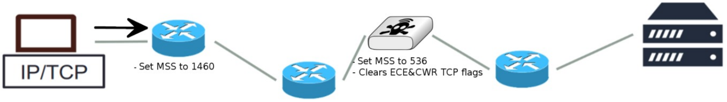
Tracebox


IP/TCP
- Set MSS to 1460
- Set MSS to 536
- Clears ECE&CWR TCP flags
Sent TCP SYN:
Received ICMP Time Exceeded:
IP
TCP
ICMP
Version
IHL (Header Length)
Type of Service (TOS)
Total Length
Identification
IP Flags
Fragment Offset
2
1
Protocol
Header Checksum
Source Address
Destination Address
IP Option (variable length, optional, not common)
Source Port
Destination Port
Sequence Number
Acknowledgment Number
Offset
Reserved
SYN/ECE/CWR
Window
Checksum
Initial MSS
Payload
Type
Code
Other message specific information...
8 Bytes
Updated Checksum
MSS: 1460
Byte Offset
Bit
Nibble
Byte
Word

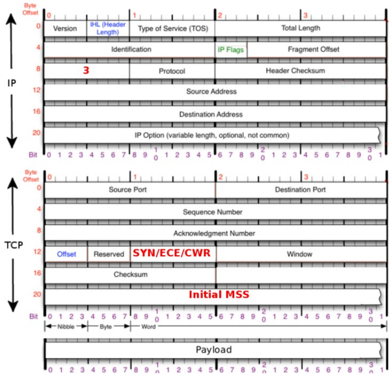
Tracebox


IP/TCP
- Set MSS to 1460
- Set MSS to 536
- Clears ECE&CWR TCP flags
Sent TCP SYN:
Received ICMP Time Exceeded:
IP
TCP
ICMP
Version
IHL (Header Length)
Type of Service (TOS)
Total Length
Identification
IP Flags
Fragment Offset
2
1
Protocol
Header Checksum
Source Address
Destination Address
IP Option (variable length, optional, not common)
Type
Code
Checksum
Other message specific information...
8 Bytes
Source Port
Destination Port
Sequence Number
Acknowledgment Number
Offset
Reserved
SYN/ECE/CWR
Window
Checksum
Initial MSS
Updated Checksum
MSS: 1460
Payload
Nibble
Byte
Word

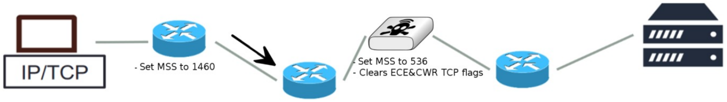
Tracebox



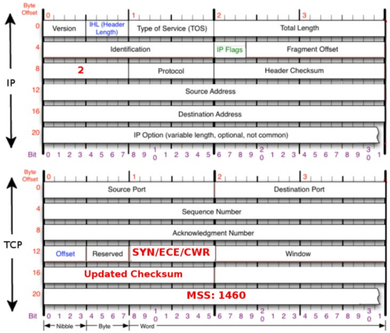
Sent TCP SYN:



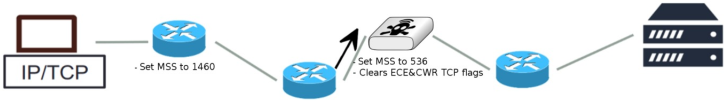
Tracebox



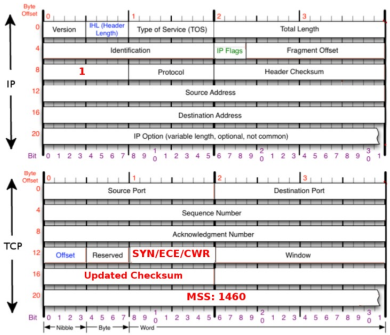
Sent TCP SYN:



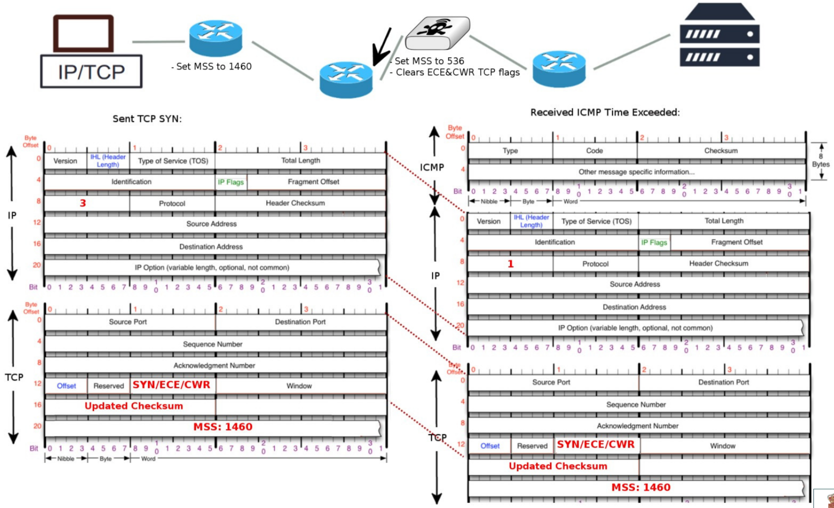
Tracebox



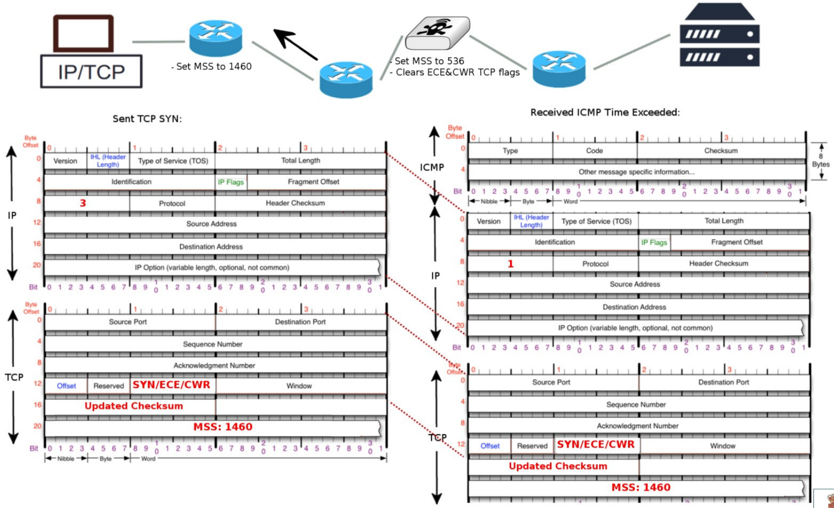
Sent TCP SYN:



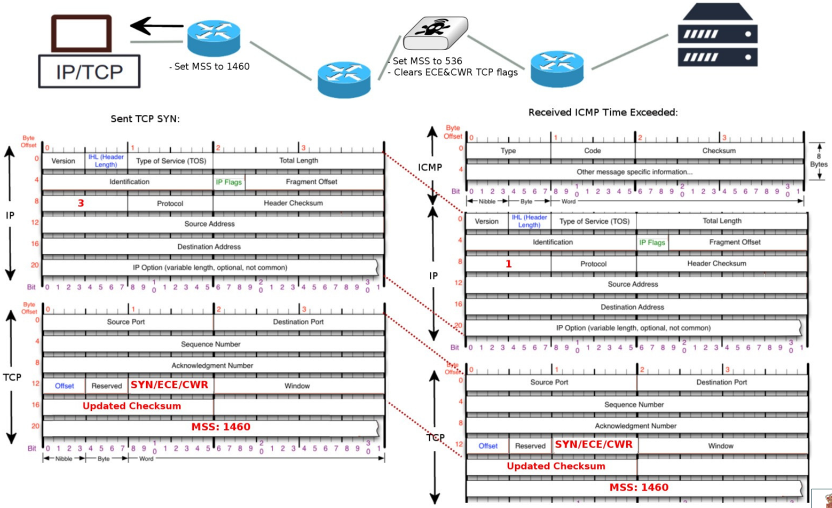
Tracebox


IP/TCP
- Set MSS to 1460
- Set MSS to 536
- Clears ECE&CWR TCP flags
Sent TCP SYN:
Received ICMP Time Exceeded:
IP
TCP
ICMP
Version
IHL (Header Length)
Type of Service (TOS)
Total Length
Identification
IP Flags
Fragment Offset
3
1
Protocol
Header Checksum
Source Address
Destination Address
IP Option (variable length, optional, not common)
Source Port
Destination Port
Sequence Number
Acknowledgment Number
Offset
Reserved
SYN/ECE/CWR
Window
Updated Checksum
MSS: 1460
Type
Code
Checksum
Other message specific information...
8 Bytes
Byte Offset
Bit
Nibble
Byte
Word

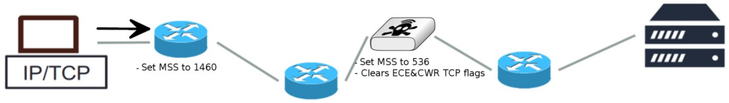
Tracebox


IP/TCP
- Set MSS to 1460
- Set MSS to 536
- Clears ECE&CWR TCP flags
Sent TCP SYN:
Received ICMP Time Exceeded:
IP
TCP
ICMP
Version
IHL (Header Length)
Type of Service (TOS)
Total Length
Identification
IP Flags
Fragment Offset
3
1
Protocol
Header Checksum
Source Address
Destination Address
IP Option (variable length, optional, not common)
Source Port
Destination Port
Sequence Number
Acknowledgment Number
Offset
Reserved
SYN/ECE/CWR
Window
Updated Checksum
MSS: 1460
Type
Code
Checksum
Other message specific information...
8 Bytes
Byte Offset
Bit
Nibble
Byte
Word

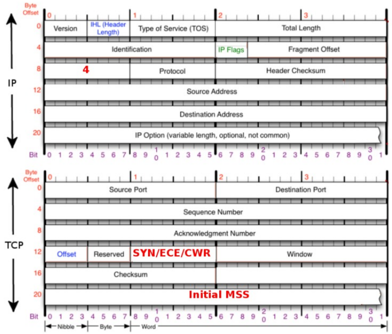
Tracebox


IP/TCP
- Set MSS to 1460
- Set MSS to 536
- Clears ECE&CWR TCP flags
Sent TCP SYN:
Received ICMP Time Exceeded:
IP
TCP
ICMP
3
1
SYN/ECE/CWR
Updated Checksum
MSS: 1460
Version
IHL (Header Length)
Type of Service (TOS)
Total Length
Identification
IP Flags
Fragment Offset
Protocol
Header Checksum
Source Address
Destination Address
IP Option (variable length, optional, not common)
Source Port
Destination Port
Sequence Number
Acknowledgment Number
Offset
Reserved
Window
Type
Code
Checksum
Other message specific information...
8 Bytes
Byte Offset
Bit
Nibble
Byte
Word

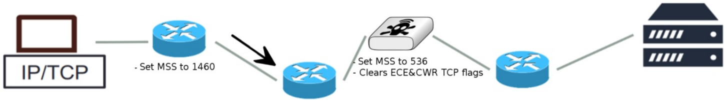
Tracebox



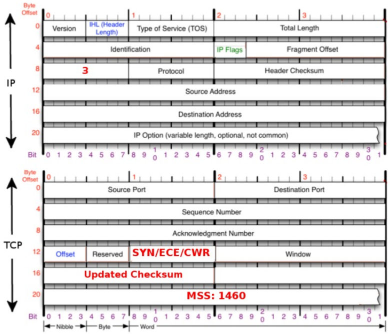
Sent TCP SYN:



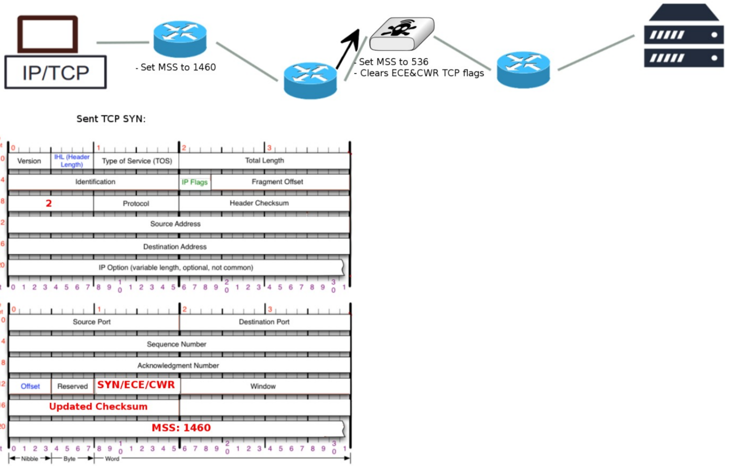
Tracebox



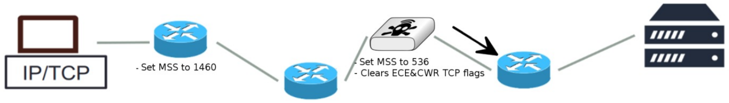
Sent TCP SYN:



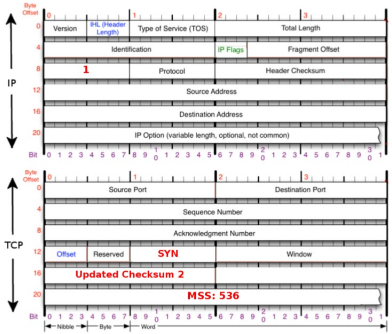
Tracebox


IP/TCP
- Set MSS to 1460
- Set MSS to 536
- Clears ECE&CWR TCP flags
Sent TCP SYN:
IP
TCP
Version
IHL (Header Length)
Type of Service (TOS)
Total Length
Identification
IP Flags
Fragment Offset
2
Protocol
Header Checksum
Source Address
Destination Address
IP Option (variable length, optional, not common)
Source Port
Destination Port
Sequence Number
Acknowledgment Number
Offset
Reserved
SYN/ECE/CWR
Window
Updated Checksum
MSS: 1460
Byte Offset
Bit
Nibble
Byte
Word

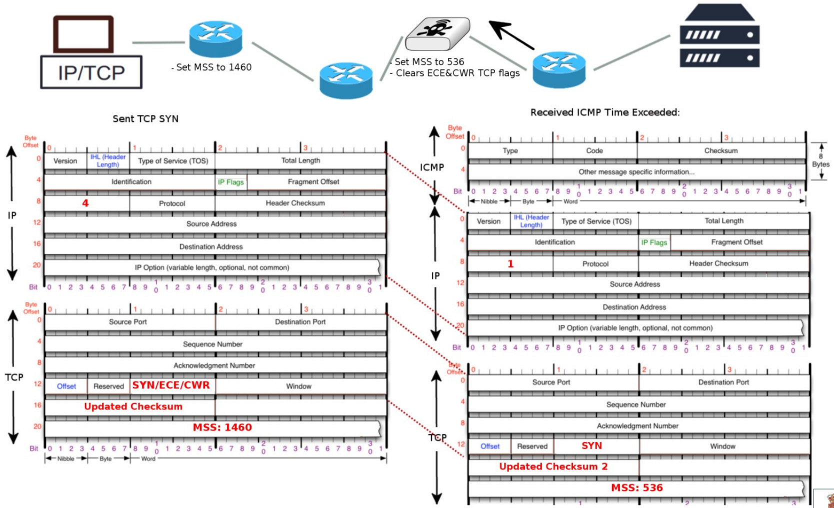
Tracebox



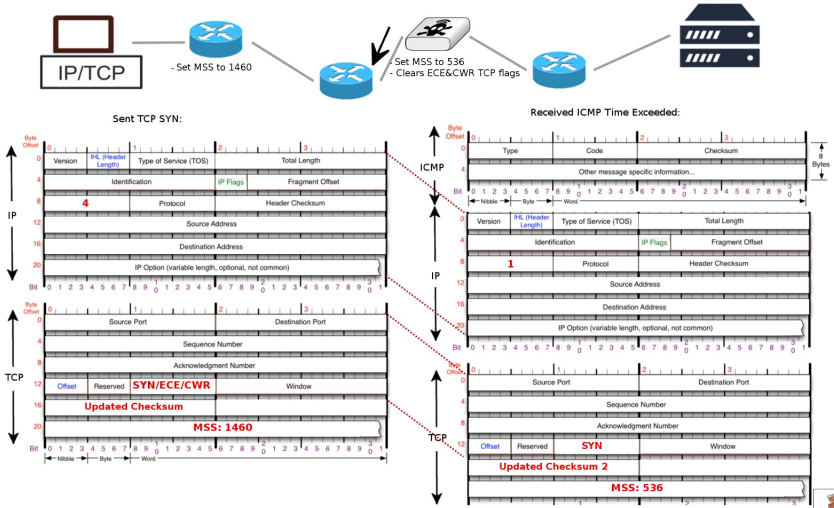
Sent TCP SYN:



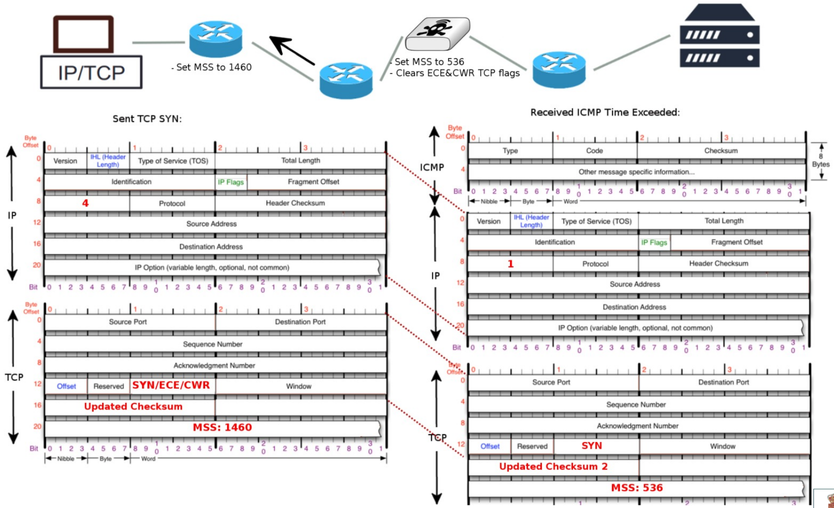
Tracebox


IP/TCP
- Set MSS to 1460
- Set MSS to 536
- Clears ECE&CWR TCP flags
Sent TCP SYN
Received ICMP Time Exceeded:
IP
TCP
ICMP
Version
IHL (Header Length)
Type of Service (TOS)
Total Length
Identification
IP Flags
Fragment Offset
4
1
Protocol
Header Checksum
Source Address
Destination Address
IP Option (variable length, optional, not common)
Source Port
Destination Port
Sequence Number
Acknowledgment Number
Offset
Reserved
SYN/ECE/CWR
SYN
Window
Updated Checksum
Updated Checksum 2
MSS: 1460
MSS: 536
Type
Code
Checksum
Other message specific information...
8 Bytes
Byte Offset
Bit
Nibble
Byte
Word

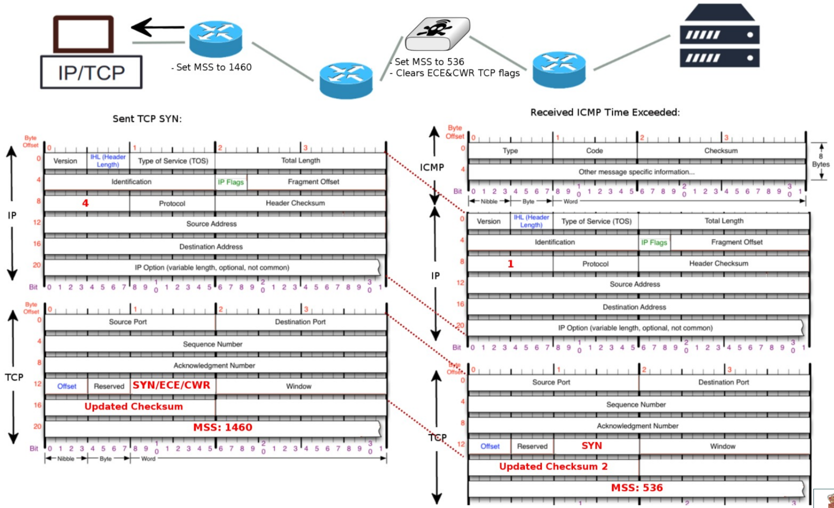
Tracebox


IP/TCP
- Set MSS to 1460
- Set MSS to 536
- Clears ECE&CWR TCP flags
Sent TCP SYN:
Received ICMP Time Exceeded:
IP
TCP
ICMP
4
1
SYN/ECE/CWR
Updated Checksum
MSS: 1460
SYN
Updated Checksum 2
MSS: 536
Version
IHL (Header Length)
Type of Service (TOS)
Total Length
Identification
IP Flags
Fragment Offset
Protocol
Header Checksum
Source Address
Destination Address
IP Option (variable length, optional, not common)
Source Port
Destination Port
Sequence Number
Acknowledgment Number
Offset
Reserved
Window
Type
Code
Checksum
Other message specific information...
8 Bytes
Byte Offset
Bit
Nibble
Byte
Word

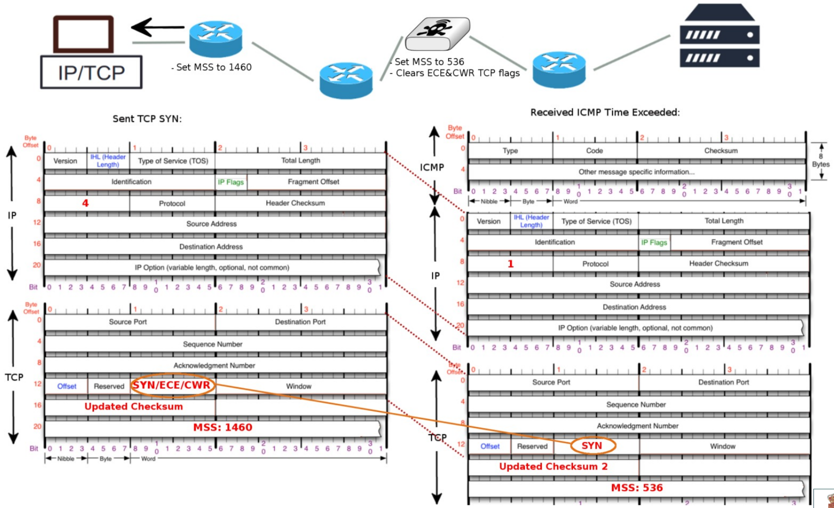
Tracebox


IP/TCP
- Set MSS to 1460
- Set MSS to 536
- Clears ECE&CWR TCP flags
Sent TCP SYN:
Received ICMP Time Exceeded:
IP
TCP
ICMP
Version
IHL (Header Length)
Type of Service (TOS)
Total Length
Identification
IP Flags
Fragment Offset
4
1
Protocol
Header Checksum
Source Address
Destination Address
IP Option (variable length, optional, not common)
Source Port
Destination Port
Sequence Number
Acknowledgment Number
Offset
Reserved
SYN/ECE/CWR
SYN
Window
Updated Checksum
Updated Checksum 2
MSS: 1460
MSS: 536
Type
Code
Checksum
Other message specific information...
8 Bytes
Byte Offset
Bit
Nibble
Byte
Word

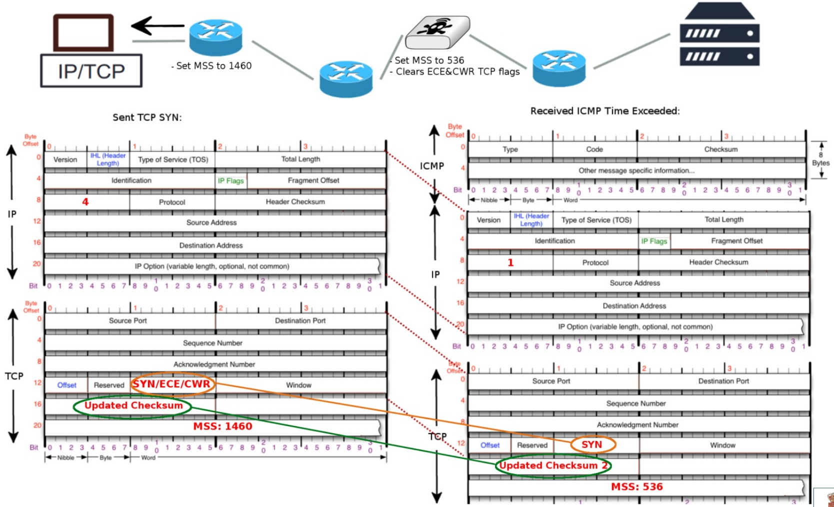
Tracebox


IP/TCP
- Set MSS to 1460
- Set MSS to 536
- Clears ECE&CWR TCP flags
Sent TCP SYN:
Received ICMP Time Exceeded:
IP
TCP
ICMP
Version
IHL (Header Length)
Type of Service (TOS)
Total Length
Identification
IP Flags
Fragment Offset
4
1
Protocol
Header Checksum
Source Address
Destination Address
IP Option (variable length, optional, not common)
Source Port
Destination Port
Sequence Number
Acknowledgment Number
Offset
Reserved
SYN/ECE/CWR
SYN
Window
Updated Checksum
Updated Checksum 2
MSS: 1460
MSS: 536
Type
Code
Checksum
Other message specific information...
8 Bytes
Byte Offset
Bit
Nibble
Byte
Word

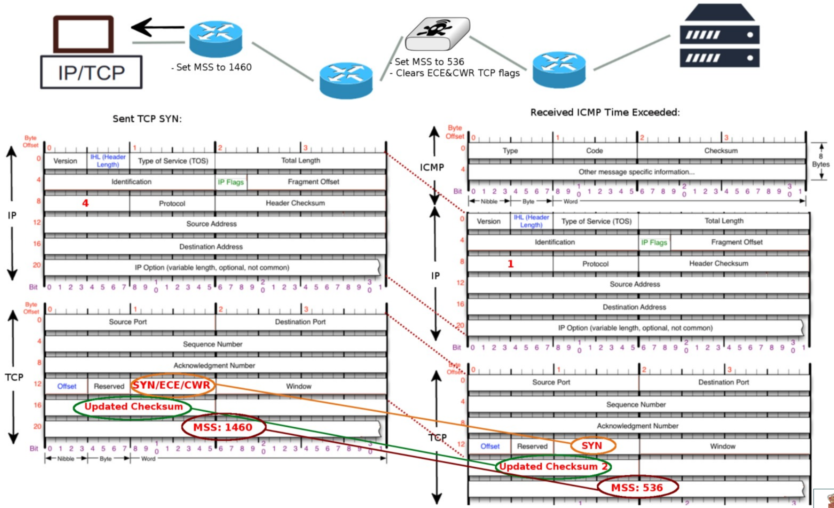
Tracebox


IP/TCP
- Set MSS to 1460
- Set MSS to 536
- Clears ECE&CWR TCP flags
Sent TCP SYN:
Received ICMP Time Exceeded:
IP
TCP
ICMP
4
1
SYN/ECE/CWR
Updated Checksum
MSS: 1460
SYN
Updated Checksum 2
MSS: 536

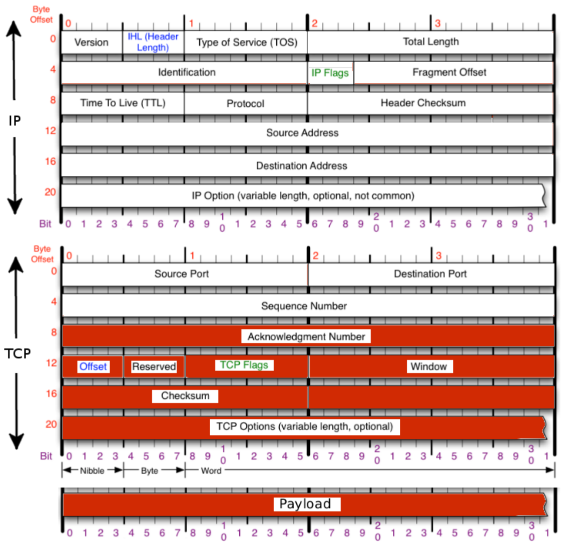
Tracebox


IP/TCP
- Set MSS to 1460
- Set MSS to 536
- Clears ECE&CWR TCP flags
Sent TCP SYN:
Received ICMP Time Exceeded:
IP
TCP
ICMP
4
1
SYN/ECE/CWR
SYN
Updated Checksum
Updated Checksum 2
MSS: 1460
MSS: 536
8 Bytes

Tracebox


IP/TCP
- Set MSS to 1460
- Set MSS to 536
- Clears ECE&CWR TCP flags
Sent TCP SYN:
Received ICMP Time Exceeded:
IP
TCP
ICMP
4
1
SYN/ECE/CWR
Updated Checksum
MSS: 1460
SYN
Updated Checksum 2
MSS: 536

Cannot detect all changes



ICMP Payload size

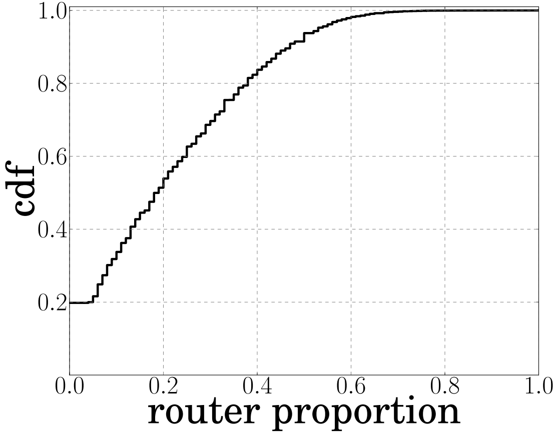
- ICMP only includes the network header plus the first 8 bytes of he transport header.
    - RFC792 (ICMPv4):
      "Internet Header + 64 bits of Data Datagram"
    - RFC1812 (ICMPv4):
      "the ICMP datagram SHOULD contain as much of the original datagram as possible without the length of the ICMP datagram exceeding 576 bytes."
    - RFC4443 (ICMPv6):
      "As much of invoking packet as possible without the ICMPv6 packet exceeding the minimum IPv6 MTU"

ICMP Payload size

- ICMP only includes the network header plus the first 8 bytes of he transport header.
  - RFC792 (ICMPv4):
    "Internet Header + 64 bits of Data Datagram"
  - RFC1812 (ICMPv4):
    "the ICMP datagram SHOULD contain as much of the original datagram as possible without the length of the ICMP datagram exceeding 576 bytes."
  - RFC4443 (ICMPv6):
    "As much of invoking packet as possible without the ICMPv6 packet exceeding the minimum IPv6 MTU"
- Maximal quoting by default on Linux, Cisco IOX, HP routers, Alcatel routers, PaloAlto Fiewall, etc.

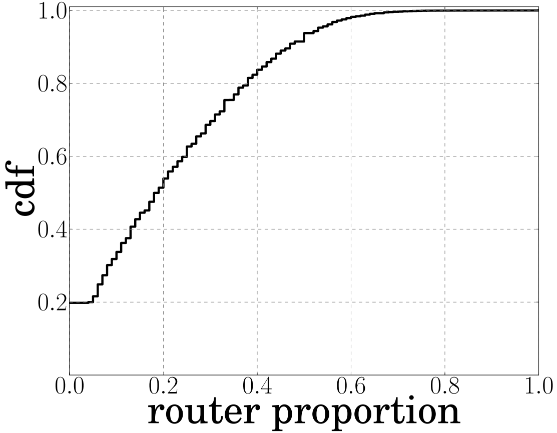
ICMPv4 Payload size

- RFC1812-compliant routers (2013, 72 PL VPs to Alexa 5000)



ICMPv4 Payload size

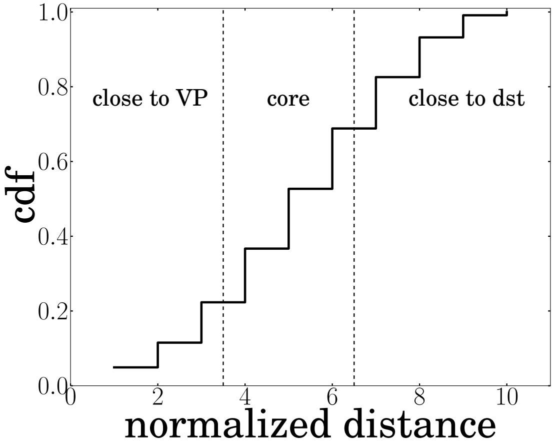
- RFC1812-compliant routers (2013, 72 PL VPs to Alexa 5000)



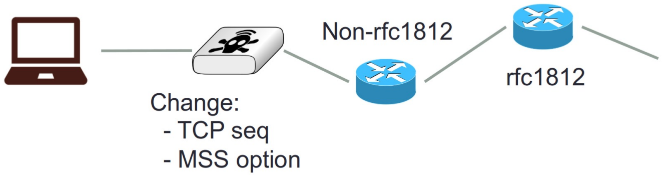
- 80 % of Internet paths contains at least on RFC1812-capable router

ICMPv4 Payload size

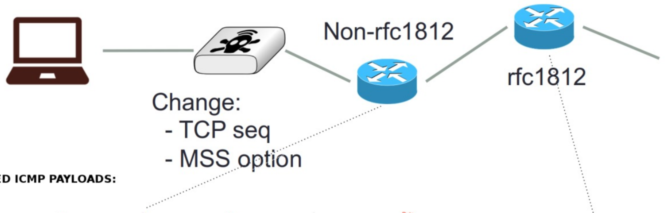
- RFC1812-compliant routers location (2013, 72 PL VPs to Alexa 5000)



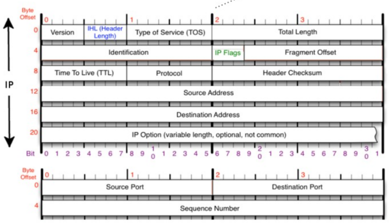
ICMP detection limitation

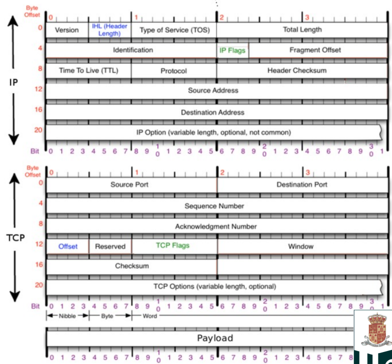


ICMP detection limitation

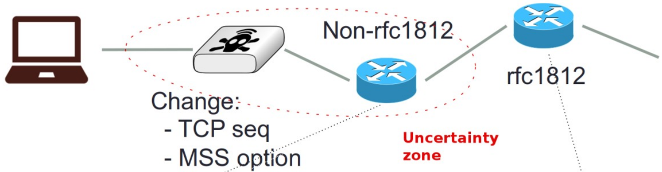


RECEIVED ICMP PAYLOADS:

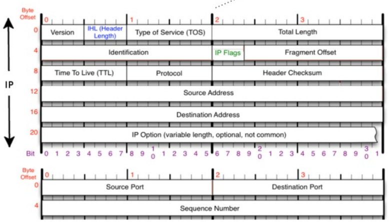


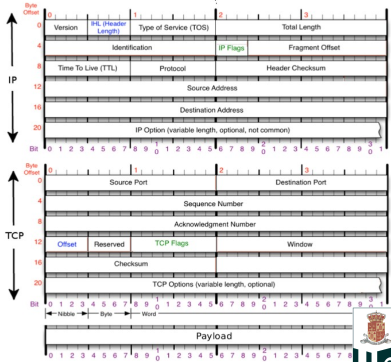


ICMP detection limitation



RECEIVED ICMP PAYLOADS:





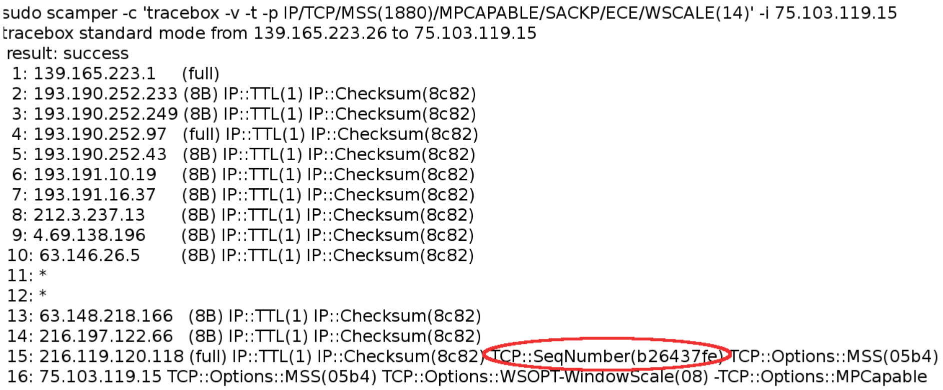
Use cases

- Testing new protocols deployability
  - MPTCP, TCP FO, TCP EDO, ...
- Testing new hardware/configurations
  - CGN deployment, ...
- Locating an issue
- Network management/debugging

Output Example

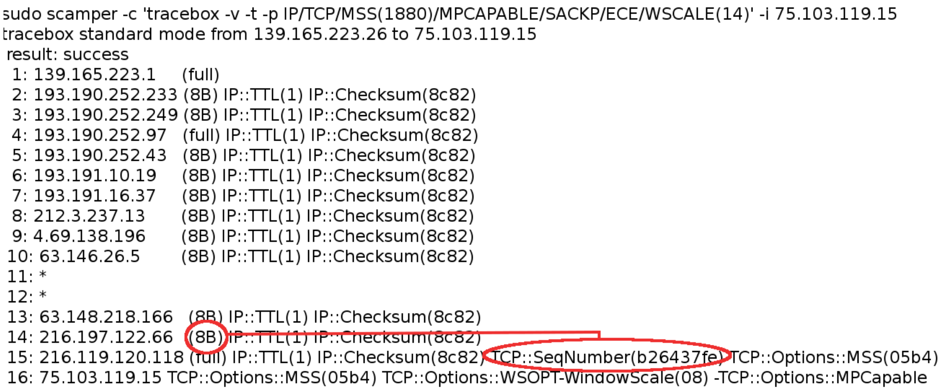
```
sudo scamper -c 'tracebox -v -t -p IP/TCP/MSS(1880)/MPCAPABLE/SACKP/ECE/WSCALE(14)' -i 75.103.119.15
tracebox standard mode from 139.165.223.26 to 75.103.119.15
 result: success
  1: 139.165.223.1    (full)
  2: 193.190.252.233 (8B) IP::TTL(1) IP::Checksum(8c82)
  3: 193.190.252.249 (8B) IP::TTL(1) IP::Checksum(8c82)
  4: 193.190.252.97   (full) IP::TTL(1) IP::Checksum(8c82)
  5: 193.190.252.43   (8B) IP::TTL(1) IP::Checksum(8c82)
  6: 193.191.10.19    (8B) IP::TTL(1) IP::Checksum(8c82)
  7: 193.191.16.37    (8B) IP::TTL(1) IP::Checksum(8c82)
  8: 212.3.237.13     (8B) IP::TTL(1) IP::Checksum(8c82)
  9: 4.69.138.196     (8B) IP::TTL(1) IP::Checksum(8c82)
 10: 63.146.26.5      (8B) IP::TTL(1) IP::Checksum(8c82)
 11: *
 12: *
 13: 63.148.218.166  (8B) IP::TTL(1) IP::Checksum(8c82)
 14: 216.197.122.66  (8B) IP::TTL(1) IP::Checksum(8c82)
 15: 216.119.120.118 (full) IP::TTL(1) IP::Checksum(8c82) TCP::SeqNumber(b26437fe) TCP::Options::MSS(05b4)
 16: 75.103.119.15 TCP::Options::MSS(05b4) TCP::Options::WSOPT-WindowScale(08) -TCP::Options::MPCapable
```

Output Example



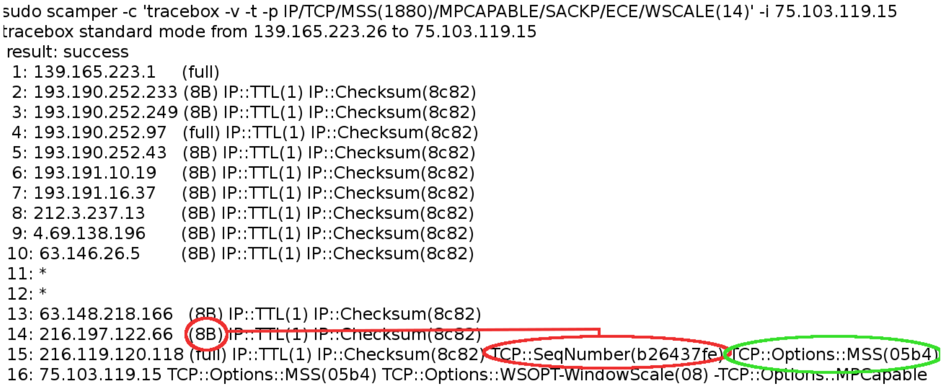
```
sudo scamper -c 'tracebox -v -t -p IP/TCP/MSS(1880)/MPCAPABLE/SACKP/ECE/WSCALE(14)' -i 75.103.119.15
tracebox standard mode from 139.165.223.26 to 75.103.119.15
 result: success
  1: 139.165.223.1     (full)
  2: 193.190.252.233 (8B) IP::TTL(1) IP::Checksum(8c82)
  3: 193.190.252.249 (8B) IP::TTL(1) IP::Checksum(8c82)
  4: 193.190.252.97   (full) IP::TTL(1) IP::Checksum(8c82)
  5: 193.190.252.43   (8B) IP::TTL(1) IP::Checksum(8c82)
  6: 193.191.10.19     (8B) IP::TTL(1) IP::Checksum(8c82)
  7: 193.191.16.37     (8B) IP::TTL(1) IP::Checksum(8c82)
  8: 212.3.237.13       (8B) IP::TTL(1) IP::Checksum(8c82)
  9: 4.69.138.196       (8B) IP::TTL(1) IP::Checksum(8c82)
 10: 63.146.26.5        (8B) IP::TTL(1) IP::Checksum(8c82)
 11: *
 12: *
 13: 63.148.218.166  (8B) IP::TTL(1) IP::Checksum(8c82)
 14: 216.197.122.66  (8B) IP::TTL(1) IP::Checksum(8c82)
 15: 216.119.120.118 (full) IP::TTL(1) IP::Checksum(8c82) TCP::SeqNumber(b26437fe) TCP::Options::MSS(05b4)
 16: 75.103.119.15 TCP::Options::MSS(05b4) TCP::Options::WSOPT-WindowScale(08) -TCP::Options::MPCapable
```

Output Example



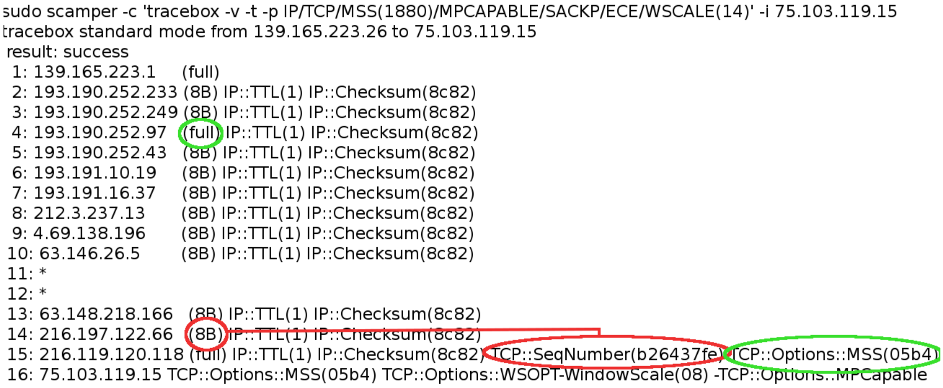
```
sudo scamper -c 'tracebox -v -t -p IP/TCP/MSS(1880)/MPCAPABLE/SACKP/ECE/WSCALE(14)' -i 75.103.119.15
tracebox standard mode from 139.165.223.26 to 75.103.119.15
 result: success
  1: 139.165.223.1    (full)
  2: 193.190.252.233 (8B) IP::TTL(1) IP::Checksum(8c82)
  3: 193.190.252.249 (8B) IP::TTL(1) IP::Checksum(8c82)
  4: 193.190.252.97   (full) IP::TTL(1) IP::Checksum(8c82)
  5: 193.190.252.43   (8B) IP::TTL(1) IP::Checksum(8c82)
  6: 193.191.10.19    (8B) IP::TTL(1) IP::Checksum(8c82)
  7: 193.191.16.37    (8B) IP::TTL(1) IP::Checksum(8c82)
  8: 212.3.237.13     (8B) IP::TTL(1) IP::Checksum(8c82)
  9: 4.69.138.196     (8B) IP::TTL(1) IP::Checksum(8c82)
 10: 63.146.26.5      (8B) IP::TTL(1) IP::Checksum(8c82)
 11: *
 12: *
 13: 63.148.218.166   (8B) IP::TTL(1) IP::Checksum(8c82)
 14: 216.197.122.66   (8B) IP::TTL(1) IP::Checksum(8c82)
 15: 216.119.120.118 (full) IP::TTL(1) IP::Checksum(8c82) TCP::SeqNumber(b26437fe) TCP::Options::MSS(05b4)
 16: 75.103.119.15 TCP::Options::MSS(05b4) TCP::Options::WSOPT-WindowScale(08) -TCP::Options::MPCapable
```

Output Example



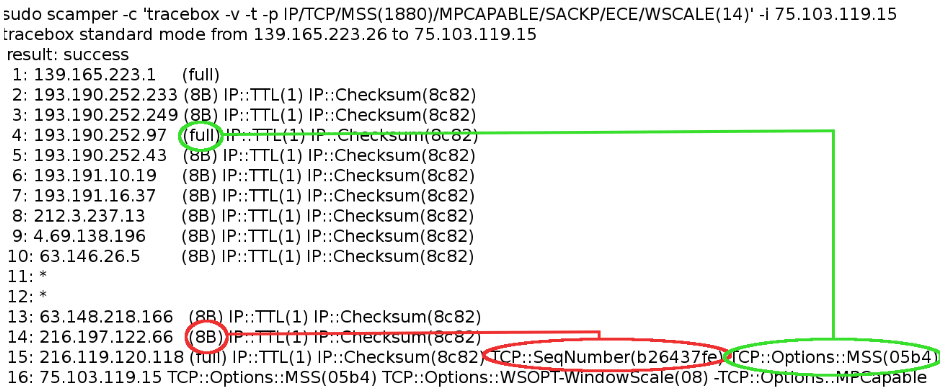
```
sudo scamper -c 'tracebox -v -t -p IP/TCP/MSS(1880)/MPCAPABLE/SACKP/ECE/WSCALE(14)' -i 75.103.119.15
tracebox standard mode from 139.165.223.26 to 75.103.119.15
 result: success
  1: 139.165.223.1    (full)
  2: 193.190.252.233 (8B) IP::TTL(1) IP::Checksum(8c82)
  3: 193.190.252.249 (8B) IP::TTL(1) IP::Checksum(8c82)
  4: 193.190.252.97   (full) IP::TTL(1) IP::Checksum(8c82)
  5: 193.190.252.43   (8B) IP::TTL(1) IP::Checksum(8c82)
  6: 193.191.10.19    (8B) IP::TTL(1) IP::Checksum(8c82)
  7: 193.191.16.37    (8B) IP::TTL(1) IP::Checksum(8c82)
  8: 212.3.237.13     (8B) IP::TTL(1) IP::Checksum(8c82)
  9: 4.69.138.196     (8B) IP::TTL(1) IP::Checksum(8c82)
 10: 63.146.26.5      (8B) IP::TTL(1) IP::Checksum(8c82)
 11: *
 12: *
 13: 63.148.218.166  (8B) IP::TTL(1) IP::Checksum(8c82)
 14: 216.197.122.66  (8B) IP::TTL(1) IP::Checksum(8c82)
 15: 216.119.120.118 (full) IP::TTL(1) IP::Checksum(8c82) TCP::SeqNumber(b26437fe) TCP::Options::MSS(05b4)
 16: 75.103.119.15 TCP::Options::MSS(05b4) TCP::Options::WSOPT-WindowScale(08) -TCP::Options::MPCapable
```

Output Example



```
sudo scamper -c 'tracebox -v -t -p IP/TCP/MSS(1880)/MPCAPABLE/SACKP/ECE/WSCALE(14)' -i 75.103.119.15
tracebox standard mode from 139.165.223.26 to 75.103.119.15
 result: success
  1: 139.165.223.1    (full)
  2: 193.190.252.233 (8B) IP::TTL(1) IP::Checksum(8c82)
  3: 193.190.252.249 (8B) IP::TTL(1) IP::Checksum(8c82)
  4: 193.190.252.97   (full) IP::TTL(1) IP::Checksum(8c82)
  5: 193.190.252.43   (8B) IP::TTL(1) IP::Checksum(8c82)
  6: 193.191.10.19    (8B) IP::TTL(1) IP::Checksum(8c82)
  7: 193.191.16.37    (8B) IP::TTL(1) IP::Checksum(8c82)
  8: 212.3.237.13     (8B) IP::TTL(1) IP::Checksum(8c82)
  9: 4.69.138.196     (8B) IP::TTL(1) IP::Checksum(8c82)
 10: 63.146.26.5      (8B) IP::TTL(1) IP::Checksum(8c82)
 11: *
 12: *
 13: 63.148.218.166  (8B) IP::TTL(1) IP::Checksum(8c82)
 14: 216.197.122.66  (8B) IP::TTL(1) IP::Checksum(8c82)
 15: 216.119.120.118 (full) IP::TTL(1) IP::Checksum(8c82) TCP::SeqNumber(b26437fe) TCP::Options::MSS(05b4)
 16: 75.103.119.15 TCP::Options::MSS(05b4) TCP::Options::WSOPT-WindowScale(08) -TCP::Options::MPCapable
```

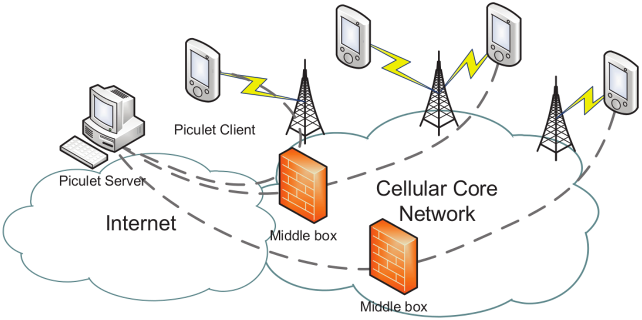
Output Example



```
sudo scamper -c 'tracebox -v -t -p IP/TCP/MSS(1880)/MPCAPABLE/SACKP/ECE/WSCALE(14)' -i 75.103.119.15
tracebox standard mode from 139.165.223.26 to 75.103.119.15
 result: success
  1: 139.165.223.1    (full)
  2: 193.190.252.233 (8B) IP::TTL(1) IP::Checksum(8c82)
  3: 193.190.252.249 (8B) IP::TTL(1) IP::Checksum(8c82)
  4: 193.190.252.97   (full) IP::TTL(1) IP::Checksum(8c82)
  5: 193.190.252.43   (8B) IP::TTL(1) IP::Checksum(8c82)
  6: 193.191.10.19    (8B) IP::TTL(1) IP::Checksum(8c82)
  7: 193.191.16.37    (8B) IP::TTL(1) IP::Checksum(8c82)
  8: 212.3.237.13     (8B) IP::TTL(1) IP::Checksum(8c82)
  9: 4.69.138.196     (8B) IP::TTL(1) IP::Checksum(8c82)
 10: 63.146.26.5      (8B) IP::TTL(1) IP::Checksum(8c82)
 11: *
 12: *
 13: 63.148.218.166  (8B) IP::TTL(1) IP::Checksum(8c82)
 14: 216.197.122.66  (8B) IP::TTL(1) IP::Checksum(8c82)
 15: 216.119.120.118 (full) IP::TTL(1) IP::Checksum(8c82) TCP::SeqNumber(b26437fe) TCP::Options::MSS(05b4)
 16: 75.103.119.15 TCP::Options::MSS(05b4) TCP::Options::WSOPT-WindowScale(08) -TCP::Options::MPCapable
```

What about cellular networks ?

- There are middleboxes too[7]:



[7] Zhaoguang Wang et al. “An untold story of middleboxes in cellular networks”. In: *A SIGCOMM Computer Communication Review* 41.4 (2011), pp. 374–385.

TraceboxAndroid[8]

- On-demand & Background probing

[8] Valentin Thirion, Korian Edeline, and Benoit Donnet. “Tracking Middleboxes in the Mobile World with TraceboxAndroid”. In: *Traffic Monitoring and Analysis*. Springer, 201 pp. 79–91.

TraceboxAndroid[8]

- On-demand & Background probing
- A rooted version
    - Require to root the phone

[8] Valentin Thirion, Korian Edeline, and Benoit Donnet. “Tracking Middleboxes in the Mobile World with TraceboxAndroid”. In: *Traffic Monitoring and Analysis*. Springer, 201 pp. 79–91.

TraceboxAndroid[8]

- On-demand & Background probing
- A rooted version
  - Require to root the phone
- A non-rooted version
  - Non-rooted traceroutes to retreive path-level information
  - Self-controlled server
  - Troubleshooting incentives

[8]Valentin Thirion, Korian Edeline, and Benoit Donnet. “Tracking Middleboxes in the Mobile World with TraceboxAndroid”. In: *Traffic Monitoring and Analysis*. Springer, 201 pp. 79–91.

TraceboxAndroid[8]

- On-demand & Background probing
- A rooted version
  - Require to root the phone
- A non-rooted version
  - Non-rooted traceroutes to retreive path-level information
  - Self-controlled server
  - Troubleshooting incentives
- Interested ?
  Send me an email at **korian.edeline@ulg.ac.be** to be notified when the new version is released.

[8] Valentin Thirion, Korian Edeline, and Benoit Donnet. “Tracking Middleboxes in the Mobile World with TraceboxAndroid”. In: *Traffic Monitoring and Analysis*. Springer, 201 pp. 79–91.

Plan

Tracebox implementations

- Standalone Tracebox
- Scamper

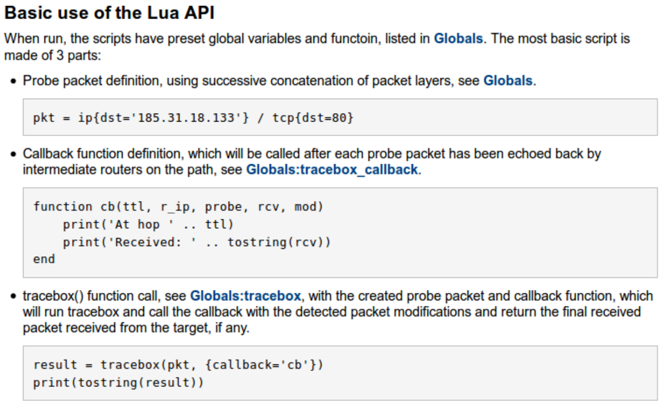
Standalone Tracebox

- Uses the previous mechanism to detect middleboxes.
- Implemented in C++ with Lua embedded.
- Libcrafter allows for efficiently describe probes as Scapy.
- Open source
- Supports Linux and Mac OSX.
- `http://github.com/tracebox/tracebox`
- `http://www.tracebox.org/`
- More details:[9]

[9] Gregory Detal et al. "Revealing middlebox interference with tracebox". In: *Proceed of the 2013 conference on Internet measurement conference.* ACM. 2013, pp. 1–8.

Standalone Tracebox

- Uses the previous mechanism to detect middleboxes.
- Implemented in C++ with Lua embedded.



Basic use of the Lua API

When run, the scripts have preset global variables and functoin, listed in Globals. The most basic script is made of 3 parts:

- Probe packet definition, using successive concatenation of packet layers, see Globals.

```
pkt = ip{dst='185.31.18.133'} / tcp{dst=80}
```

- Callback function definition, which will be called after each probe packet has been echoed back by intermediate routers on the path, see Globals:tracebox_callback.

```
function cb(ttl, r_ip, probe, rcv, mod)
    print('At hop ' .. ttl)
    print('Received: ' .. tostring(rcv))
end
```

- tracebox() function call, see Globals:tracebox, with the created probe packet and callback function, which will run tracebox and call the callback with the detected packet modifications and return the final received packet received from the target, if any.

```
result = tracebox(pkt, {callback='cb'})
print(tostring(result))
```

Standalone Tracebox

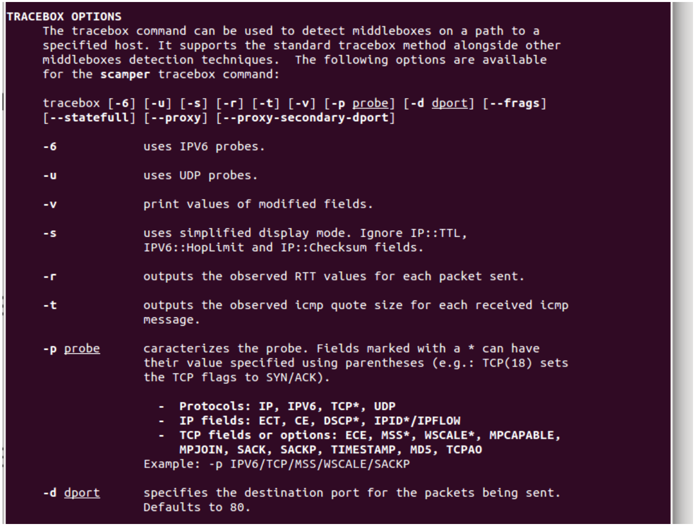
- Uses the previous mechanism to detect middleboxes.
- Implemented in C++ with Lua embedded.
- Libcrafter allows for efficiently describe probes as Scapy.
- Open source
- Supports Linux and Mac OSX.
- `http://github.com/tracebox/tracebox`
- `http://www.tracebox.org/`
- More details:[10]

[10] Gregory Detal et al. "Revealing middlebox interference with tracebox". In: *Proceed of the 2013 conference on Internet measurement conference.* ACM. 2013, pp. 1–8.

Scamper

- All-around parallelized topology/performance analyzing tool.
- Implements various simple and complex measurement methods (ping, traceroute, dealias, tbit, ...).

Scamper



```
TRACEBOX OPTIONS
     The tracebox command can be used to detect middleboxes on a path to a
     specified host. It supports the standard tracebox method alongside other
     middleboxes detection techniques.  The following options are available
     for the scamper tracebox command:

     tracebox [-6] [-u] [-s] [-r] [-t] [-v] [-p probe] [-d dport] [--frags]
     [--statefull] [--proxy] [--proxy-secondary-dport]

     -6             uses IPV6 probes.

     -u             uses UDP probes.

     -v             print values of modified fields.

     -s             uses simplified display mode. Ignore IP::TTL,
                    IPV6::HopLimit and IP::Checksum fields.

     -r             outputs the observed RTT values for each packet sent.

     -t             outputs the observed icmp quote size for each received icmp
                    message.

     -p probe       caracterizes the probe. Fields marked with a * can have
                    their value specified using parentheses (e.g.: TCP(18) sets
                    the TCP flags to SYN/ACK).

                      -  Protocols: IP, IPV6, TCP*, UDP
                      -  IP fields: ECT, CE, DSCP*, IPID*/IPFLOW
                      -  TCP fields or options: ECE, MSS*, WSCALE*, MPCAPABLE,
                         MPJOIN, SACK, SACKP, TIMESTAMP, MD5, TCPAO
                    Example: -p IPV6/TCP/MSS/WSCALE/SACKP

     -d dport       specifies the destination port for the packets being sent.
                    Defaults to 80.
```

Scamper

- Native output format: warts.
- IPv6 support
- Open source
- Supports FreeBSD, OpenBSD, NetBSD, Linux, MacOS X, Solaris, Windows, and more.
- `http://www.caida.org/tools/measurement/scamper/`
- Debian/Ubuntu packages, FreeBSD ports, ...
- More details:[11]

[11] Matthew Luckie. “Scamper: a scalable and extensible packet prober for active measurement of the internet”. In: *Proceedings of the 10th ACM SIGCOMM conference Internet measurement*. ACM. 2010, pp. 239–245.

Thank you !