



Wireshark

MANUEL MESAS GUTIÉRREZ

RAÚL DEL POZO MORENO

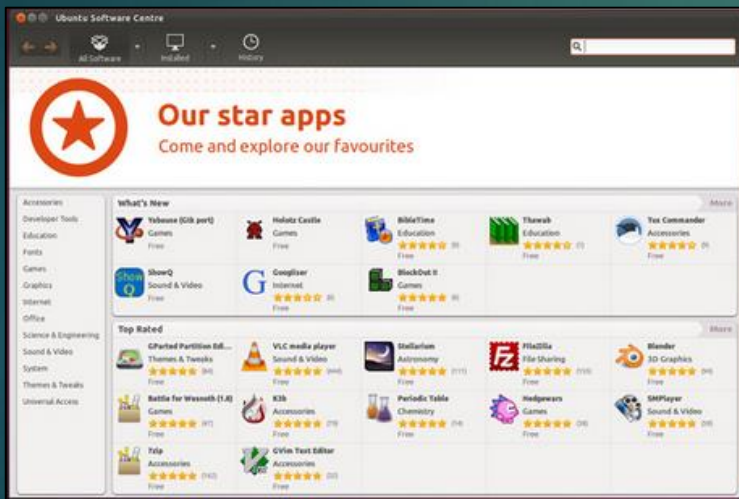
Un poco de historia

- ▶ Lanzado en 1999 como Ethereal y mas tarde renombrado como Wireshark por su creador, Gerald Combs.
- ▶ No tiene fines lucrativos.
- ▶ Su código libre y multiplataforma.
- ▶ Programado en C y C++.
- ▶ Es uno de los sniffers mas conocidos en la actualidad, ofreciendo múltiples funcionalidades y una extensa variedad de protocolos.
- ▶ Licencia GPL.
- ▶ La mayor ventaja que ofrece respecto a los demás analizadores es su GUI y su gran capacidad para organizar y filtrar la información capturada.



Instalación de Wireshark

- ▶ En su pagina oficial dispone de instaladores gráficos para Windows, tanto de 32 como de 64 bits, para macOS y del código fuente.



```
sudo apt install wireshark
```

Stable Release (2.6.0)

↓ Windows Installer (64-bit)
Windows Installer (32-bit)
Windows PortableApps® (32-bit)
macOS 10.6 and later Intel 64-bit .dmg
Source Code

Old Stable Release (2.4.6)

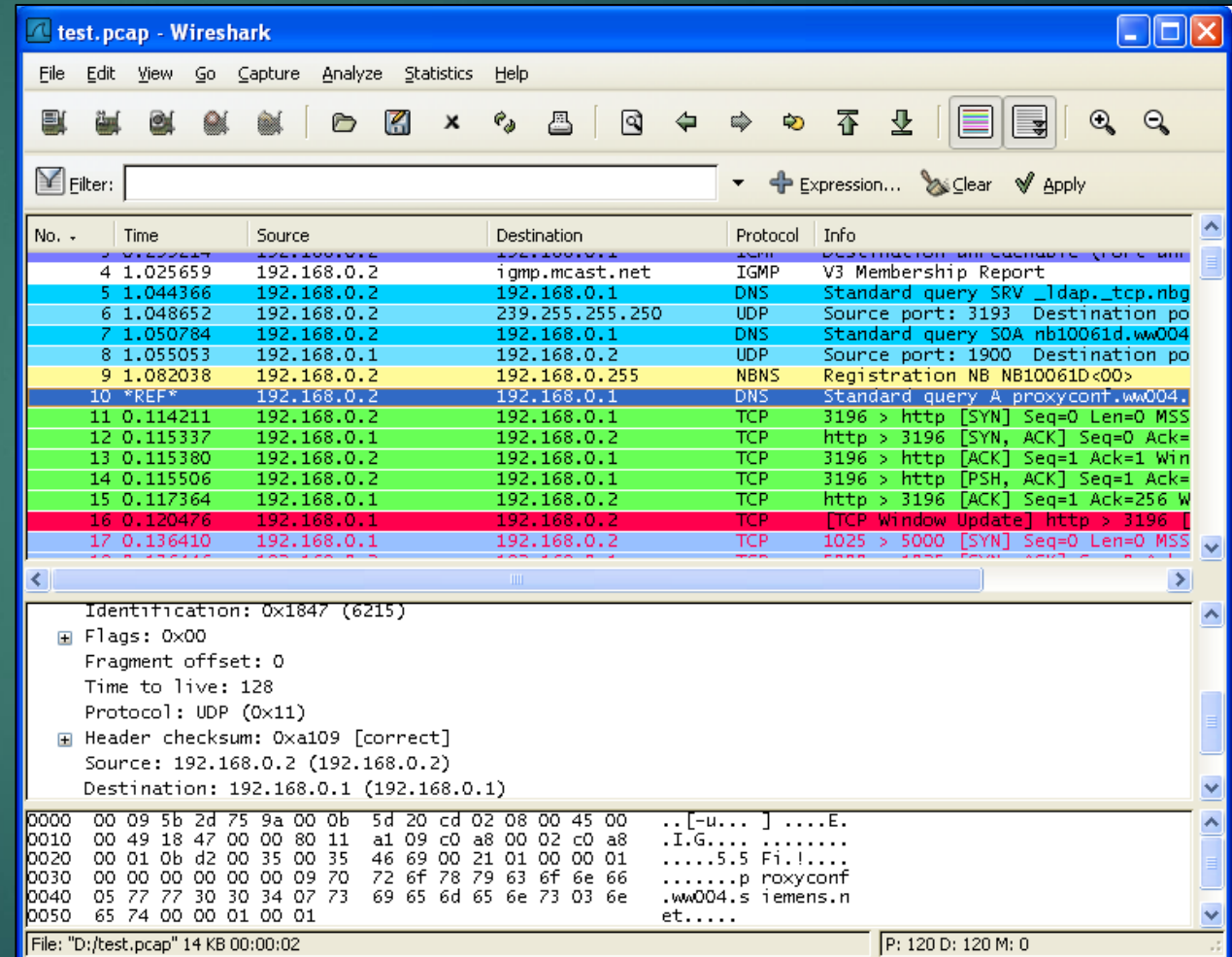
Development Release (2.5.1)

Documentation

- ▶ Para distribuciones Linux, se puede instalar desde el código fuente, desde los repositorios oficiales de Linux o el centro de software.

Como es Wireshark

- ▶ Filtrado de protocolos.
- ▶ Diferenciación por colores según el protocolo.
- ▶ Cuadro con información de cada conexión.
- ▶ Información de forma hexadecimal del paquete tal cual se envía por la red.



Como es Wireshark

223	4.929462	192.168.1.202	8.8.8.8	DNS	69 Standard query 0x0930 A google.es
224	4.965469	192.168.1.202	8.8.4.4	DNS	69 Standard query 0x0930 A google.es
225	4.971383	8.8.8.8	192.168.1.202	DNS	85 Standard query response 0x0930 A google.es A 216.58.211.35
226	5.005540	8.8.4.4	192.168.1.202	DNS	85 Standard query response 0x0930 A google.es A 216.58.211.35

>

Frame 223: 69 bytes on wire (552 bits), 69 bytes captured (552 bits) on interface 0

>

Ethernet II, Src: QuantaCo_e4:5b:d4 (c4:54:44:e4:5b:d4), Dst: Zte_da:15:67 (c4:a3:66:da:15:67)

>

Internet Protocol Version 4, Src: 192.168.1.202, Dst: 8.8.8.8

>

User Datagram Protocol, Src Port: 62892, Dst Port: 53

▼

Domain Name System (query)

Transaction ID: 0x0930

> Flags: 0x0100 Standard query

Questions: 1

Answer RRs: 0

Authority RRs: 0

Additional RRs: 0

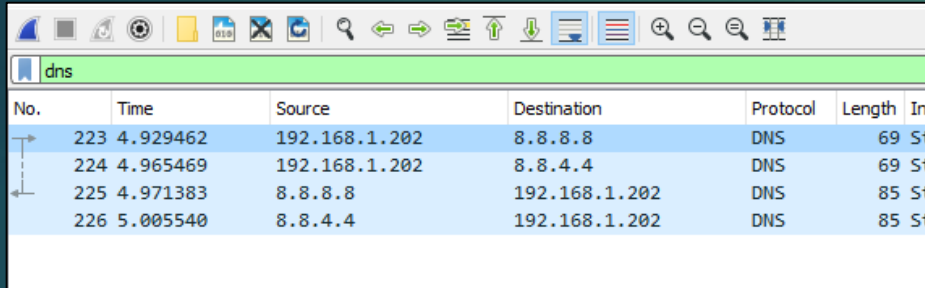
▼ Queries

▼ google.es: type A, class IN

Name: google.es

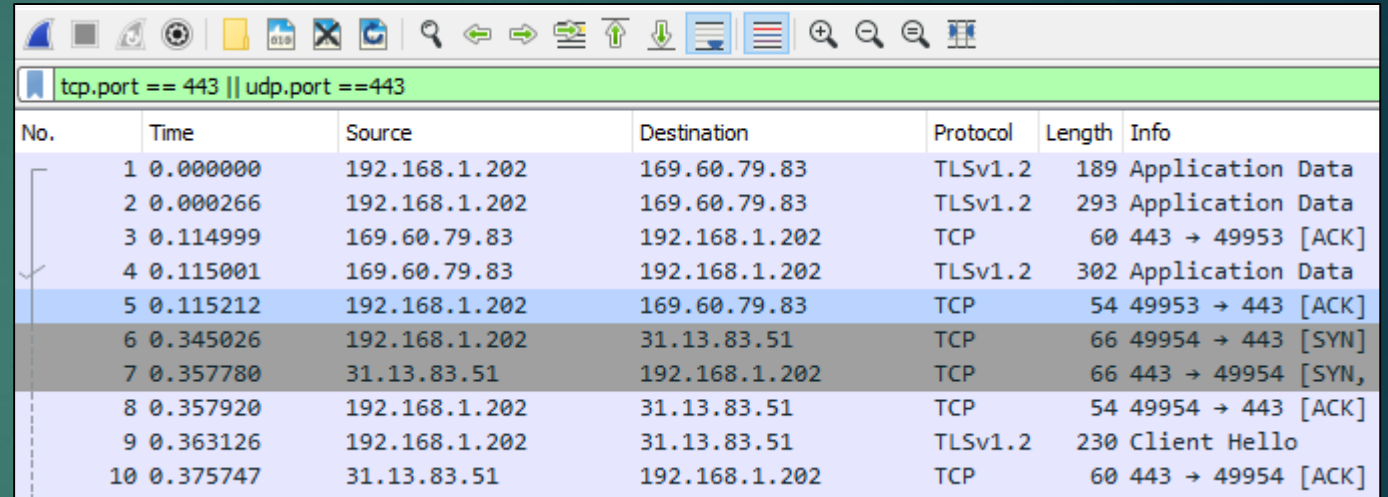
Como es Wireshark

Filtrado de protocolos



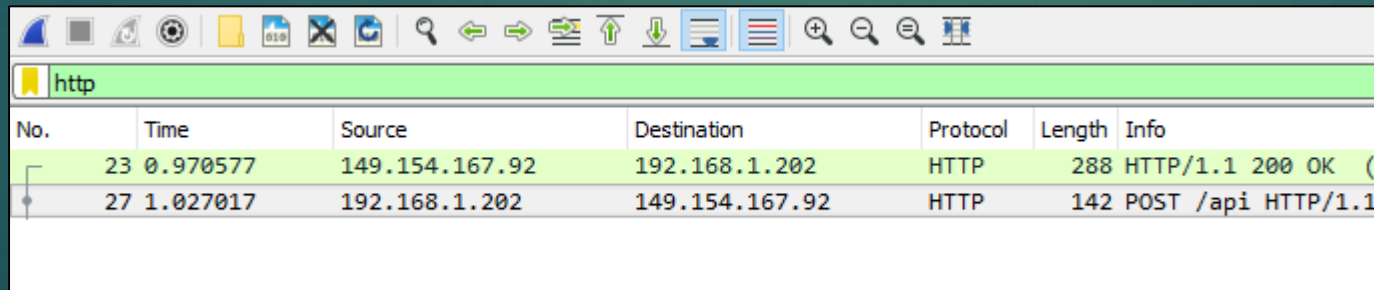
Wireshark interface showing a filter for 'dns' and a list of captured DNS packets.

No.	Time	Source	Destination	Protocol	Length	Info
223	4.929462	192.168.1.202	8.8.8.8	DNS	69	Standard query query
224	4.965469	192.168.1.202	8.8.4.4	DNS	69	Standard query query
225	4.971383	8.8.8.8	192.168.1.202	DNS	85	Standard query response
226	5.005540	8.8.4.4	192.168.1.202	DNS	85	Standard query response



Wireshark interface showing a filter for 'tcp.port == 443 || udp.port == 443' and a list of captured packets including TLS and TCP.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.202	169.60.79.83	TLSv1.2	189	Application Data
2	0.000266	192.168.1.202	169.60.79.83	TLSv1.2	293	Application Data
3	0.114999	169.60.79.83	192.168.1.202	TCP	60	443 → 49953 [ACK]
4	0.115001	169.60.79.83	192.168.1.202	TLSv1.2	302	Application Data
5	0.115212	192.168.1.202	169.60.79.83	TCP	54	49953 → 443 [ACK]
6	0.345026	192.168.1.202	31.13.83.51	TCP	66	49954 → 443 [SYN]
7	0.357780	31.13.83.51	192.168.1.202	TCP	66	443 → 49954 [SYN, Seq=49954, Win=0, Len=0]
8	0.357920	192.168.1.202	31.13.83.51	TCP	54	49954 → 443 [ACK]
9	0.363126	192.168.1.202	31.13.83.51	TLSv1.2	230	Client Hello
10	0.375747	31.13.83.51	192.168.1.202	TCP	60	443 → 49954 [ACK]

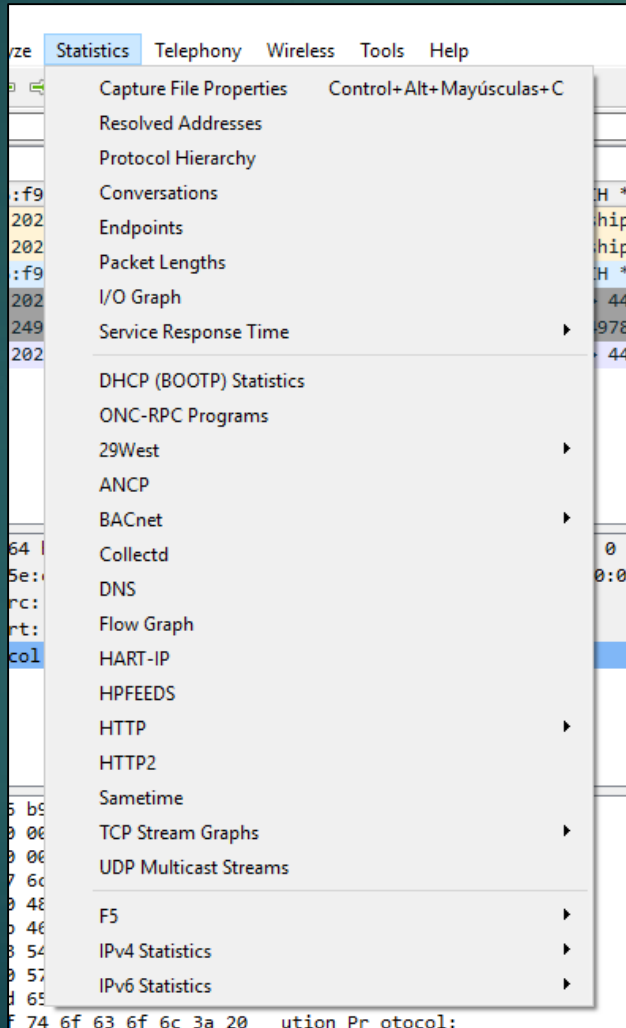


Wireshark interface showing a filter for 'http' and a list of captured HTTP packets.

No.	Time	Source	Destination	Protocol	Length	Info
23	0.970577	149.154.167.92	192.168.1.202	HTTP	288	HTTP/1.1 200 OK (text/css)
27	1.027017	192.168.1.202	149.154.167.92	HTTP	142	POST /api HTTP/1.1

Como es Wireshark

Pestaña Statistic



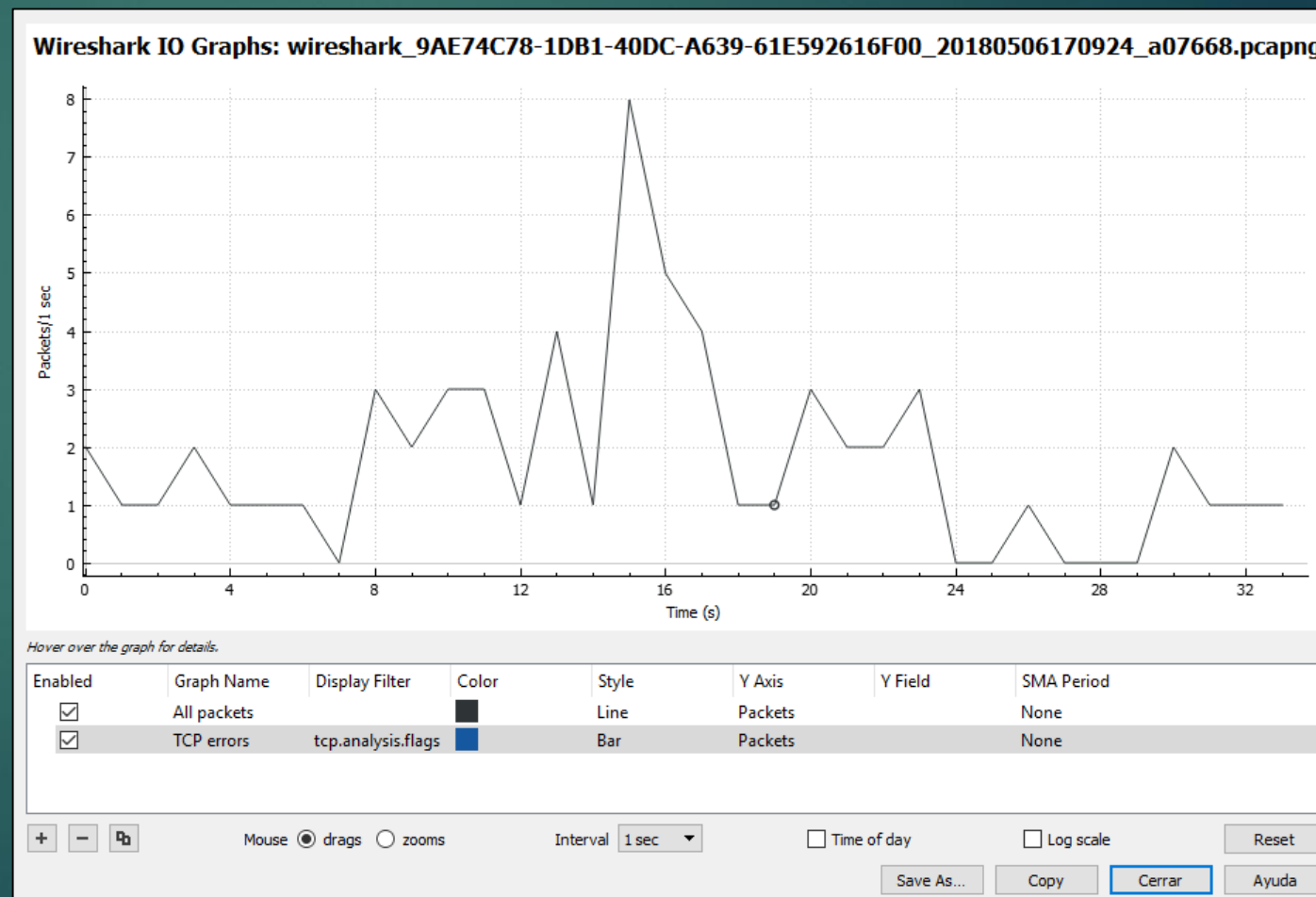
► Multitud de opciones

- I/O Graph
- Flow Graph
- Endpoints
- HTTP
- IPv4 / IPv6

Como es Wireshark

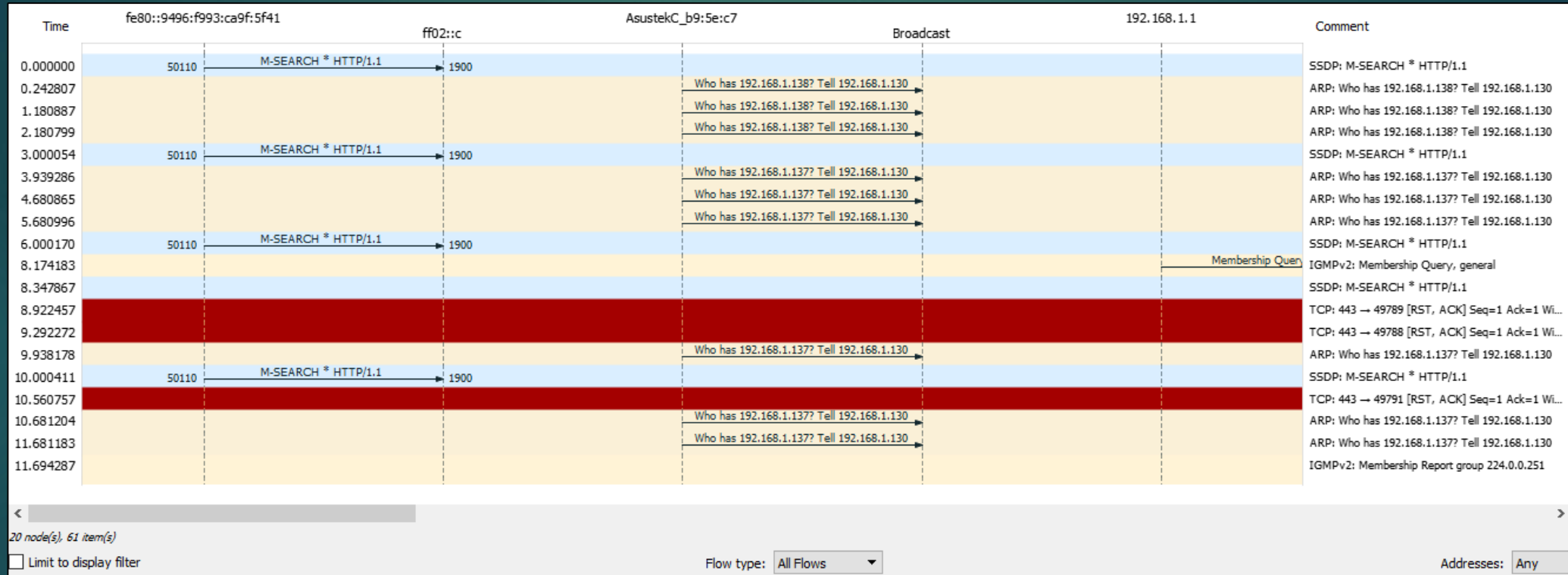
I/O Graph

- Muestra la cantidad de paquetes por segundo a lo largo en el tiempo.



Como es Wireshark

Flow Graph

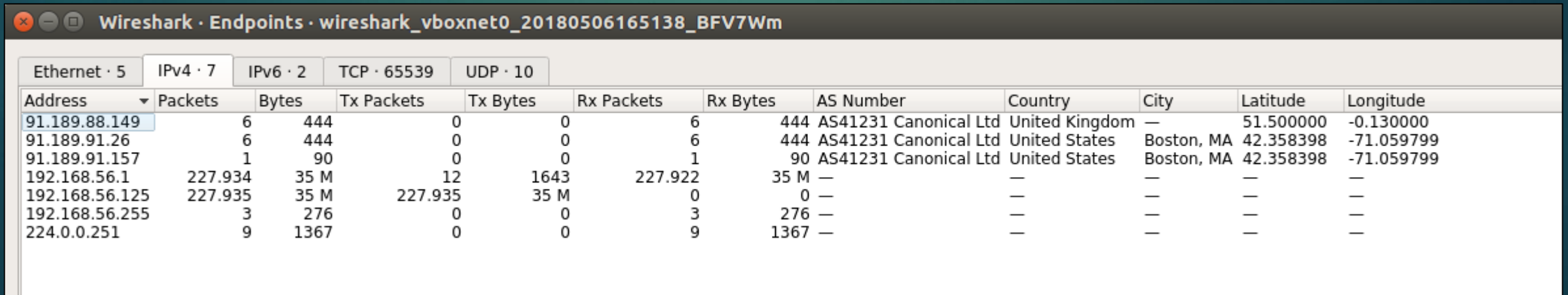


- Muestra un diagrama de flujo donde podemos ver los protocolos involucrados.

Como es Wireshark

Endpoints

- Muestra los paquetes por cada IP, proporcionando información sobre el peso total de los paquetes o la localización de dicha ip.



Wireshark · Endpoints · wireshark_vboxnet0_20180506165138_BFV7Wm

Endpoints											
Ethernet · 5 IPv4 · 7 IPv6 · 2 TCP · 65539 UDP · 10											
Address	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes	AS Number	Country	City	Latitude	Longitude
91.189.88.149	6	444	0	0	6	444	AS41231 Canonical Ltd	United Kingdom	—	51.500000	-0.130000
91.189.91.26	6	444	0	0	6	444	AS41231 Canonical Ltd	United States	Boston, MA	42.358398	-71.059799
91.189.91.157	1	90	0	0	1	90	AS41231 Canonical Ltd	United States	Boston, MA	42.358398	-71.059799
192.168.56.1	227.934	35 M	12	1643	227.922	35 M	—	—	—	—	—
192.168.56.125	227.935	35 M	227.935	35 M	0	0	—	—	—	—	—
192.168.56.255	3	276	0	0	3	276	—	—	—	—	—
224.0.0.251	9	1367	0	0	9	1367	—	—	—	—	—

Para que podemos usar Wireshark

- ▶ Analizar el tráfico de red (protocolos, puertos ...)
- ▶ Detectar ataques (DDoS, MITM ...)
- ▶ Testear la seguridad de una red informática (puertos abiertos, IP, encriptación ...)
- ▶ Averiguar información (IP, Dominios, contraseñas, emails, imágenes ...)



Algunos ejemplos

Demos

- ▶ Ping.
- ▶ Detección ataque DDoS.
- ▶ Comprobar el funcionamiento de nuestro servidor.
- ▶ Ver la transferencia de archivos.
- ▶ Peligros de no usar el protocolo https en paginas con login.

Algunos ejemplos

Ping

icmp						
No.	Time	Source	Destination	Protocol	Length	Info
5	4.917428134	192.168.1.204	188.78.168.217	ICMP	120	Destination unreachable (Port unreachable)
15	28.145204441	192.168.1.204	8.8.8.8	ICMP	98	Echo (ping) request id=0x0bbd, seq=1/256, ttl=64 (reply in 16)
16	28.163913253	8.8.8.8	192.168.1.204	ICMP	98	Echo (ping) reply id=0x0bbd, seq=1/256, ttl=56 (request in 15)
17	29.147040198	192.168.1.204	8.8.8.8	ICMP	98	Echo (ping) request id=0x0bbd, seq=2/512, ttl=64 (reply in 18)
18	29.166175422	8.8.8.8	192.168.1.204	ICMP	98	Echo (ping) reply id=0x0bbd, seq=2/512, ttl=56 (request in 17)
20	30.148489016	192.168.1.204	8.8.8.8	ICMP	98	Echo (ping) request id=0x0bbd, seq=3/768, ttl=64 (reply in 21)
21	30.167371500	8.8.8.8	192.168.1.204	ICMP	98	Echo (ping) reply id=0x0bbd, seq=3/768, ttl=56 (request in 20)
23	31.150542412	192.168.1.204	8.8.8.8	ICMP	98	Echo (ping) request id=0x0bbd, seq=4/1024, ttl=64 (reply in 24)
24	31.169122465	8.8.8.8	192.168.1.204	ICMP	98	Echo (ping) reply id=0x0bbd, seq=4/1024, ttl=56 (request in 23)
26	32.152289087	192.168.1.204	8.8.8.8	ICMP	98	Echo (ping) request id=0x0bbd, seq=5/1280, ttl=64 (reply in 27)
27	32.171593070	8.8.8.8	192.168.1.204	ICMP	98	Echo (ping) reply id=0x0bbd, seq=5/1280, ttl=56 (request in 26)
30	33.153738868	192.168.1.204	8.8.8.8	ICMP	98	Echo (ping) request id=0x0bbd, seq=6/1536, ttl=64 (reply in 31)
31	33.172689097	8.8.8.8	192.168.1.204	ICMP	98	Echo (ping) reply id=0x0bbd, seq=6/1536, ttl=56 (request in 30)
35	34.154794300	192.168.1.204	8.8.8.8	ICMP	98	Echo (ping) request id=0x0bbd, seq=7/1792, ttl=64 (reply in 36)
36	34.173511156	8.8.8.8	192.168.1.204	ICMP	98	Echo (ping) reply id=0x0bbd, seq=7/1792, ttl=56 (request in 35)
37	35.156613401	192.168.1.204	8.8.8.8	ICMP	98	Echo (ping) request id=0x0bbd, seq=8/2048, ttl=64 (reply in 38)
38	35.174988302	8.8.8.8	192.168.1.204	ICMP	98	Echo (ping) reply id=0x0bbd, seq=8/2048, ttl=56 (request in 37)
39	36.158141491	192.168.1.204	8.8.8.8	ICMP	98	Echo (ping) request id=0x0bbd, seq=9/2304, ttl=64 (reply in 40)
40	36.176754136	8.8.8.8	192.168.1.204	ICMP	98	Echo (ping) reply id=0x0bbd, seq=9/2304, ttl=56 (request in 39)
41	37.160006419	192.168.1.204	8.8.8.8	ICMP	98	Echo (ping) request id=0x0bbd, seq=10/2560, ttl=64 (reply in 42)
42	37.178453049	8.8.8.8	192.168.1.204	ICMP	98	Echo (ping) reply id=0x0bbd, seq=10/2560, ttl=56 (request in 41)
44	38.161642900	192.168.1.204	8.8.8.8	ICMP	98	Echo (ping) request id=0x0bbd, seq=11/2816, ttl=64 (reply in 45)
45	38.180199927	8.8.8.8	192.168.1.204	ICMP	98	Echo (ping) reply id=0x0bbd, seq=11/2816, ttl=56 (request in 44)

Algunos ejemplos

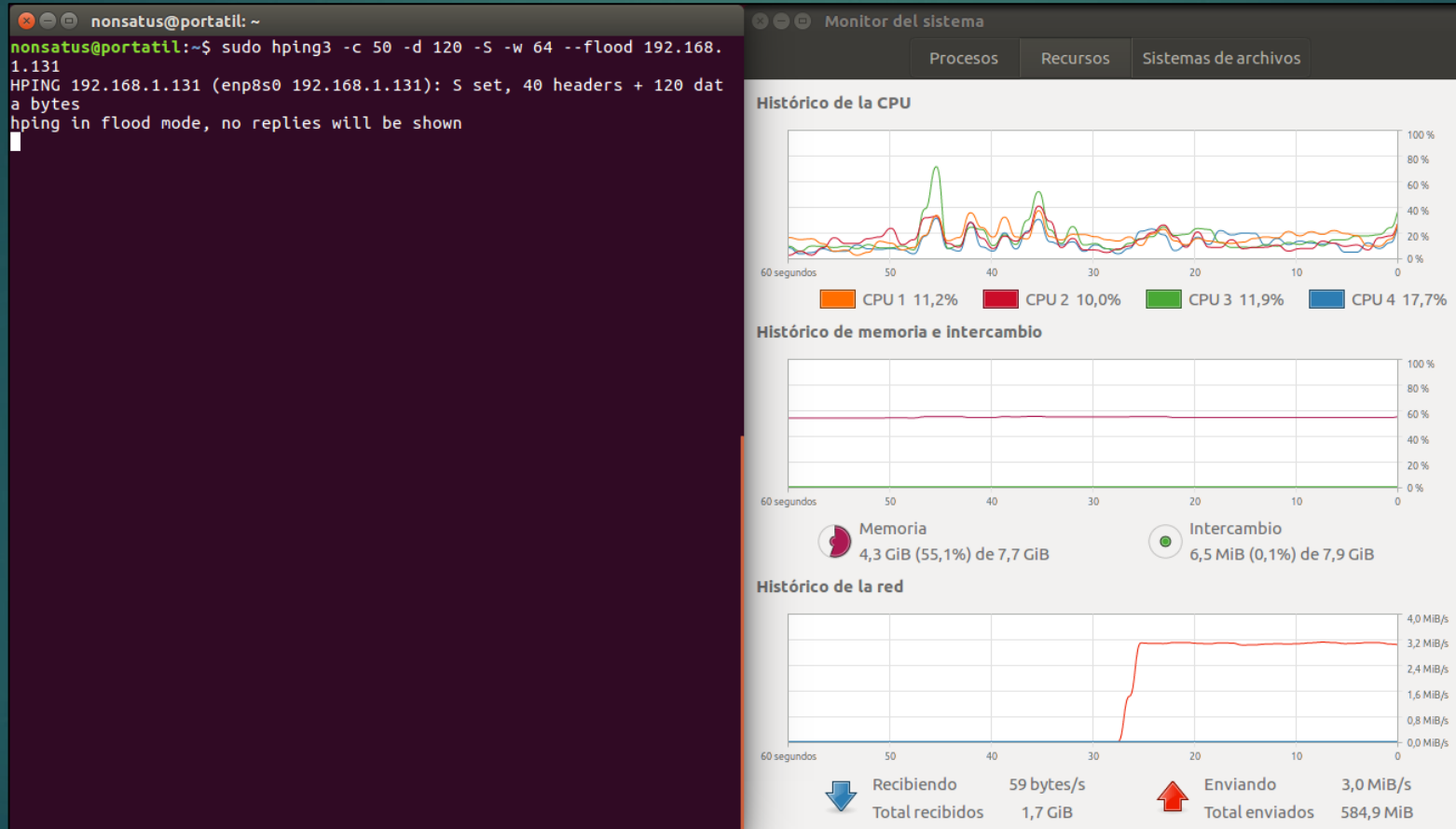
Detección ataque DDoS mediante hping3

35	9.014163000	10.0.2.15	google-public-dns-a.gc	ICMP	98	Echo (ping) request	id=0x1122, seq=29/7424, ttl=64 (reply in
36	9.029061000	google-public-dns-a.google.com	10.0.2.15	ICMP	98	Echo (ping) reply	id=0x1122, seq=29/7424, ttl=63 (request i
37	10.016893000	10.0.2.15	google-public-dns-a.gc	ICMP	98	Echo (ping) request	id=0x1122, seq=30/7680, ttl=64 (reply in
38	10.031698000	google-public-dns-a.google.com	10.0.2.15	ICMP	98	Echo (ping) reply	id=0x1122, seq=30/7680, ttl=63 (request i
39	11.018958000	10.0.2.15	google-public-dns-a.gc	ICMP	98	Echo (ping) request	id=0x1122, seq=31/7936, ttl=64 (reply in
40	11.033134000	google-public-dns-a.google.com	10.0.2.15	ICMP	98	Echo (ping) reply	id=0x1122, seq=31/7936, ttl=63 (request i
41	11.981345000	192.168.56.102	192.168.56.101	TCP	174	[TCP segment of a reassembled PDU]	
42	11.981428000	192.168.56.101	192.168.56.102	TCP	54	http > netdb-export [RST, ACK] Seq=1 Ack=121 Win=0 Len=0	
43	11.982672000	192.168.56.102	192.168.56.101	TCP	174	[TCP segment of a reassembled PDU]	
44	11.982702000	192.168.56.101	192.168.56.102	TCP	54	http > streetperfect [RST, ACK] Seq=1 Ack=121 Win=0 Len=0	
45	11.982721000	192.168.56.102	192.168.56.101	TCP	174	[TCP segment of a reassembled PDU]	
46	11.982724000	192.168.56.101	192.168.56.102	TCP	54	http > intersan [RST, ACK] Seq=1 Ack=121 Win=0 Len=0	
47	11.982731000	192.168.56.102	192.168.56.101	TCP	174	[TCP segment of a reassembled PDU]	

173012	26.765624000	192.168.56.102	192.168.56.101	TCP	174	[TCP Port numbers reused] [TCP segment of a reassembled PDU]	
173013	26.765627000	192.168.56.101	192.168.56.102	TCP	54	http > 59536 [RST, ACK] Seq=1 Ack=121 Win=0 Len=0	
173014	26.765634000	192.168.56.102	192.168.56.101	TCP	174	[TCP Port numbers reused] [TCP segment of a reassembled PDU]	
173015	26.765636000	192.168.56.101	192.168.56.102	TCP	54	http > 59537 [RST, ACK] Seq=1 Ack=121 Win=0 Len=0	
173016	27.058408000	10.0.2.15	google-public-dns-a.gc	ICMP	98	Echo (ping) request	id=0x1122, seq=47/12032, ttl=64 (reply in
173017	27.073331000	google-public-dns-a.google.com	10.0.2.15	ICMP	98	Echo (ping) reply	id=0x1122, seq=47/12032, ttl=63 (request
173018	28.060134000	10.0.2.15	google-public-dns-a.gc	ICMP	98	Echo (ping) request	id=0x1122, seq=48/12288, ttl=64 (reply in
173019	28.075853000	google-public-dns-a.google.com	10.0.2.15	ICMP	98	Echo (ping) reply	id=0x1122, seq=48/12288, ttl=63 (request
173020	29.062170000	10.0.2.15	google-public-dns-a.gc	ICMP	98	Echo (ping) request	id=0x1122, seq=49/12544, ttl=64 (reply in

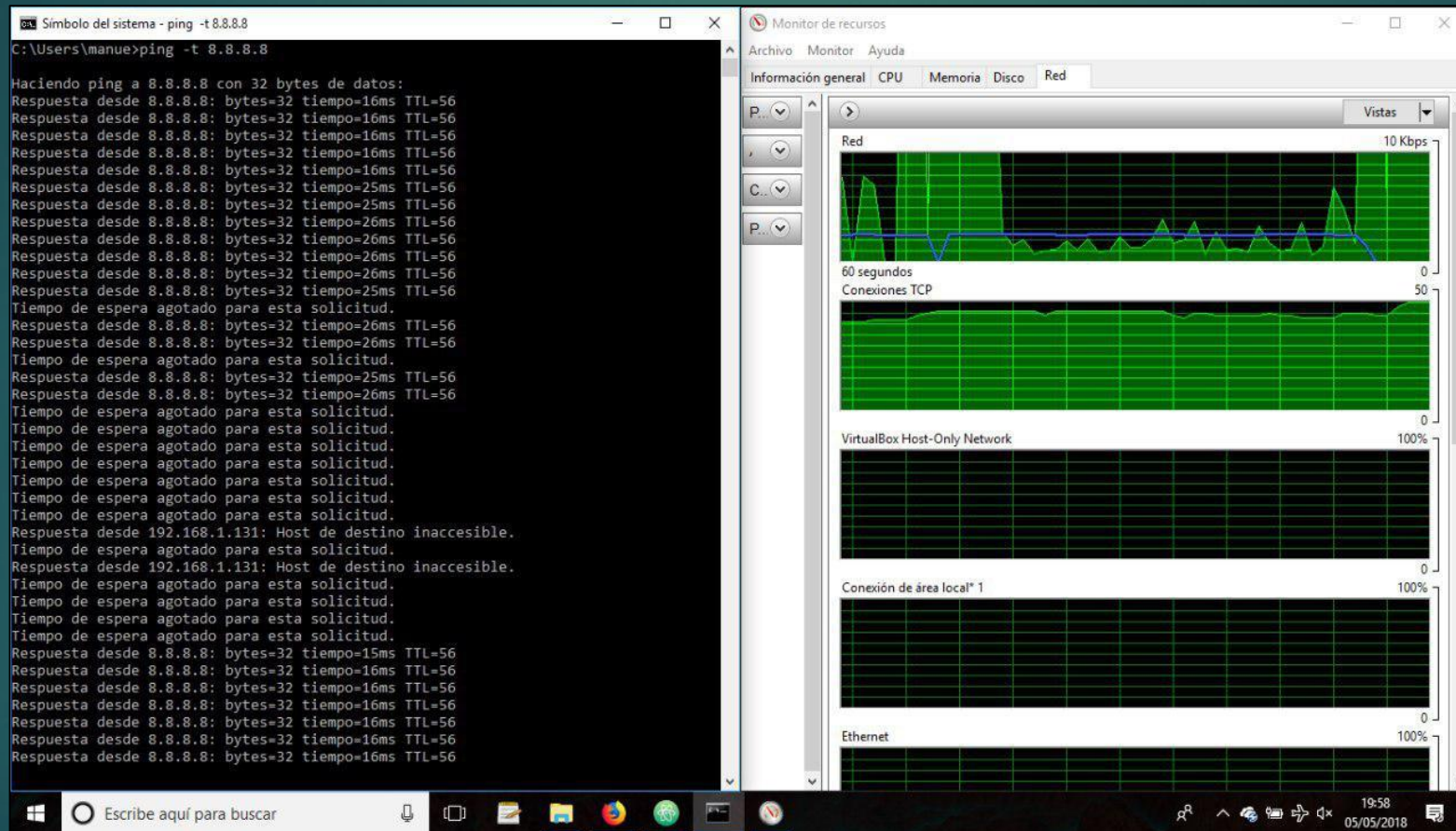
Algunos ejemplos

Detección ataque DDoS mediante hping3



Algunos ejemplos

Detección ataque DDoS mediante hping3



Algunos ejemplos

Funcionamiento balanceador

Filter:	http			▼	Expression...	Clear	Apply	Guardar
No.	Time	Source	Destination	Protocol	Length	Info		
4	0.002542000	192.168.56.101	192.168.56.125	HTTP	153	GET /hola.html HTTP/1.1		
6	0.003863000	192.168.56.125	192.168.56.101	HTTP	363	HTTP/1.1 200 OK (text/html)		
14	0.011580000	192.168.56.101	192.168.56.125	HTTP	153	GET /hola.html HTTP/1.1		
16	0.012727000	192.168.56.125	192.168.56.101	HTTP	363	HTTP/1.1 200 OK (text/html)		

► Frame 6: 363 bytes on wire (2904 bits), 363 bytes captured (2904 bits) on interface 0

► Ethernet II, Src: CadmusCo_4d:5c:b8 (08:00:27:4d:5c:b8), Dst: CadmusCo_c1:8a:bc (08:00:27:c1:8a:bc)

► Internet Protocol Version 4, Src: 192.168.56.125 (192.168.56.125), Dst: 192.168.56.101 (192.168.56.101)

► Transmission Control Protocol, Src Port: http (80), Dst Port: 58066 (58066), Seq: 1, Ack: 88, Len: 297

► Hypertext Transfer Protocol

▼ Line-based text data: text/html

```
<HTML>\n<body>\n  Maquina1\n</body>\n</HTML>\n
```

Filter:		<input type="text" value="http"/>		Expression...		Clear	Apply	Guardar
No.	Time	Source	Destination	Protocol	Length	Info		
4	0.002542000	192.168.56.101	192.168.56.125	HTTP	153	GET /hola.html HTTP/1.1		
6	0.003863000	192.168.56.125	192.168.56.101	HTTP	363	HTTP/1.1 200 OK (text/html)		
14	0.011580000	192.168.56.101	192.168.56.125	HTTP	153	GET /hola.html HTTP/1.1		
16	0.012727000	192.168.56.125	192.168.56.101	HTTP	363	HTTP/1.1 200 OK (text/html)		

► Frame 16: 363 bytes on wire (2904 bits), 363 bytes captured (2904 bits) on interface 0

► Ethernet II, Src: CadmusCo_4d:5c:b8 (08:00:27:4d:5c:b8), Dst: CadmusCo_c1:8a:bc (08:00:27:c1:8a:bc)

► Internet Protocol Version 4, Src: 192.168.56.125 (192.168.56.125), Dst: 192.168.56.101 (192.168.56.101)

► Transmission Control Protocol, Src Port: http (80), Dst Port: 58066 (58066), Seq: 1, Ack: 88, Len: 297

► Hypertext Transfer Protocol

▼ Line-based text data: text/html

```
<HTML>\n<body>\n  Maquina2\n</body>\n</HTML>\n
```

Wireshark

Algunos ejemplos

Funcionamiento balanceador

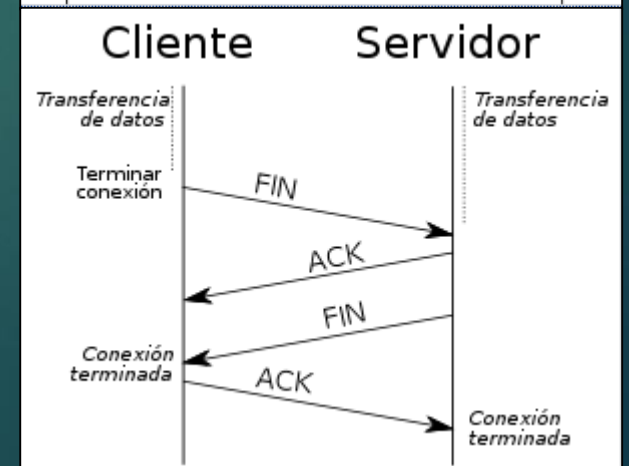
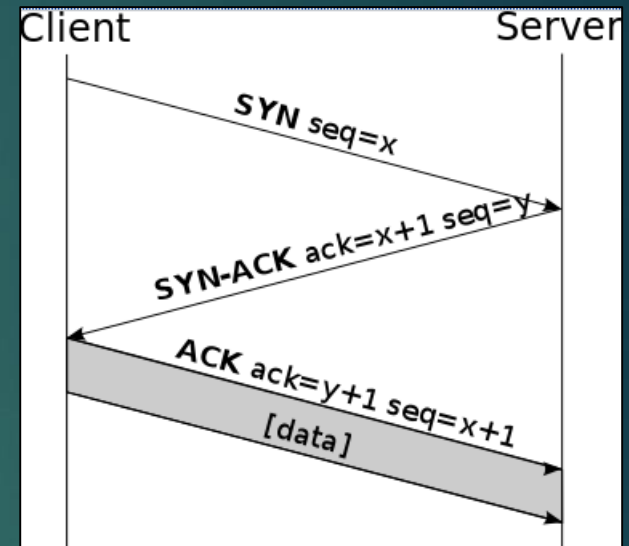
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.56.101	192.168.56.105	TCP	74	57034 > http [SYN] Seq=0 Win=
2	0.000251000	192.168.56.105	192.168.56.101	TCP	74	http > 57034 [SYN, ACK] Seq=0
3	0.000266000	192.168.56.101	192.168.56.105	TCP	66	57034 > http [ACK] Seq=1 Ack=
4	0.000542000	192.168.56.101	192.168.56.105	HTTP	153	GET /hola.html HTTP/1.1
5	0.000692000	192.168.56.105	192.168.56.101	TCP	66	http > 57034 [ACK] Seq=1 Ack=
6	0.000943000	192.168.56.105	192.168.56.101	HTTP	340	HTTP/1.1 200 OK (text/html)
7	0.000949000	192.168.56.101	192.168.56.105	TCP	66	57034 > http [ACK] Seq=88 Ack=
8	0.001231000	192.168.56.101	192.168.56.105	TCP	66	57034 > http [FIN, ACK] Seq=8
9	0.001397000	192.168.56.105	192.168.56.101	TCP	66	http > 57034 [FIN, ACK] Seq=2
10	0.001403000	192.168.56.101	192.168.56.105	TCP	66	57034 > http [ACK] Seq=89 Ack=
11	5.010198000	CadmusCo_48:d4:71	CadmusCo_c1:8a:bc	ARP	60	Who has 192.168.56.101? Tell
12	5.010216000	CadmusCo_c1:8a:bc	CadmusCo_48:d4:71	ARP	42	192.168.56.101 is at 08:00:27

Filter: Expression... Clear Apply Guardar

▶ Frame 6: 340 bytes on wire (2720 bits), 340 bytes captured (2720 bits) on interface 0
 ▶ Ethernet II, Src: CadmusCo_48:d4:71 (08:00:27:48:d4:71), Dst: CadmusCo_c1:8a:bc (08:00:27:c1:8a:bc)
 ▶ Internet Protocol Version 4, Src: 192.168.56.105 (192.168.56.105), Dst: 192.168.56.101 (192.168.56.101)
 ▶ Transmission Control Protocol, Src Port: http (80), Dst Port: 57034 (57034), Seq: 1, Ack: 88, Len: 274
 ▶ Hypertext Transfer Protocol
 ▼ Line-based text data: text/html

```

<HTML>\n
  <body>\n
    Maquina1\n
  </body>\n
</HTML>\n
  
```



Algunos ejemplos

Transferencia archivo

1	0.000...	192.168.56.1	192.168.56.105	TCP	74 55452 → ssh(22) [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=6839498 TSecr=0
2	0.000...	192.168.56.105	192.168.56.1	TCP	74 ssh(22) → 55452 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=633105
3	0.000...	192.168.56.1	192.168.56.105	TCP	66 55452 → ssh(22) [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=6839498 TSecr=633105
4	0.004...	192.168.56.105	192.168.56.1	SSHv2	107 Server: Protocol (SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.2)
5	0.004...	192.168.56.1	192.168.56.105	TCP	66 55452 → ssh(22) [ACK] Seq=1 Ack=42 Win=29312 Len=0 TSval=6839500 TSecr=633106
6	0.012...	192.168.56.1	192.168.56.105	SSHv2	107 Client: Protocol (SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.4)
7	0.013...	192.168.56.105	192.168.56.1	TCP	66 ssh(22) → 55452 [ACK] Seq=42 Ack=42 Win=29056 Len=0 TSval=633108 TSecr=6839502
8	0.013...	192.168.56.1	192.168.56.105	SSHv2	1402 Client: Key Exchange Init
9	0.013...	192.168.56.105	192.168.56.1	TCP	66 ssh(22) → 55452 [ACK] Seq=42 Ack=1378 Win=31872 Len=0 TSval=633108 TSecr=6839502
10	0.014...	192.168.56.105	192.168.56.1	SSHv2	1042 Server: Key Exchange Init
11	0.016...	192.168.56.1	192.168.56.105	SSHv2	114 Client: Diffie-Hellman Key Exchange Init
12	0.021...	192.168.56.105	192.168.56.1	SSHv2	430 Server: Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packet (len=84)
13	0.060...	192.168.56.1	192.168.56.105	TCP	66 55452 → ssh(22) [ACK] Seq=1426 Ack=1382 Win=34048 Len=0 TSval=6839514 TSecr=633110
14	1.706...	192.168.56.1	192.168.56.105	SSHv2	82 Client: New Keys
15	1.743...	192.168.56.105	192.168.56.1	TCP	66 ssh(22) → 55452 [ACK] Seq=1382 Ack=1442 Win=31872 Len=0 TSval=633541 TSecr=6839925
16	1.996...	192.168.56.1	192.168.56.105	SSHv2	110 Client: Encrypted packet (len=44)
17	1.996...	192.168.56.105	192.168.56.1	TCP	66 ssh(22) → 55452 [ACK] Seq=1382 Ack=1486 Win=31872 Len=0 TSval=633604 TSecr=6839997
18	1.996...	192.168.56.105	192.168.56.1	SSHv2	110 Server: Encrypted packet (len=44)
19	1.996...	192.168.56.1	192.168.56.105	TCP	66 55452 → ssh(22) [ACK] Seq=1486 Ack=1426 Win=34048 Len=0 TSval=6839998 TSecr=633604
20	1.996...	192.168.56.1	192.168.56.105	SSHv2	134 Client: Encrypted packet (len=68)
21	1.998...	192.168.56.105	192.168.56.1	SSHv2	118 Server: Encrypted packet (len=52)
22	1.998...	192.168.56.1	192.168.56.105	SSHv2	438 Client: Encrypted packet (len=372)
23	1.998...	192.168.56.105	192.168.56.1	SSHv2	118 Server: Encrypted packet (len=52)

49	3.672...	192.168.56.1	192.168.56.105	TCP	66 55452 → ssh(22) [FIN, ACK]
50	3.673...	192.168.56.105	192.168.56.1	TCP	66 ssh(22) → 55452 [ACK] Seq=
51	3.676...	192.168.56.105	192.168.56.1	TCP	66 ssh(22) → 55452 [FIN, ACK]
52	3.676...	192.168.56.1	192.168.56.105	TCP	66 55452 → ssh(22) [ACK] Seq=

Algunos ejemplos

Seguridad web HTTP

http						
No.	Time	Source	Destination	Protocol	Length	Info
58	1.488...	192.168.1.202	104.18.46.40	HTTP	798	POST /user?destination=portada HTTP/1.1 (application/x-www-form-urlencoded)
95	4.891...	104.18.46.40	192.168.1.202	HTTP	74	HTTP/1.1 200 OK (text/html)

Referer: http://www.elcotodecaza.com/\r\n
Content-Type: application/x-www-form-urlencoded\r\n
▶ Content-Length: 64\r\n
▶ Cookie: has_js=1; __cfduid=d0808cebc8685f5347c122bb8e4c80d461525192006; SESS2a00d8913ec85ac5889c2efa683f6fcd=75349ea78c382de91d079984eefa3857; _ga=GA1.2.83880
Connection: keep-alive\r\n
Upgrade-Insecure-Requests: 1\r\n
\r\n
[Full request URI: http://www.elcotodecaza.com/user?destination=portada]
[HTTP request 1/1]
[Response in frame: 95]
File Data: 64 bytes

▼ HTML Form URL Encoded: application/x-www-form-urlencoded

- ▶ Form item: "form_id" = "user_login"
- ▼ Form item: "name" = "Victima"
 - Key: name
 - Value: Victima
- ▼ Form item: "pass" = "pass"
 - Key: pass
 - Value: pass
- ▶ Form item: "op" = "Iniciar sesión"