

WebRTC

- The technology
 - Possible problems
 - Troubleshooting
- ICE and STUN traffic
- DTLS traffic
- SRTP traffic
 - Recognizing SRTP packets
 - Decoding SRTP packets
 - Decoded SRTP traffic
 - SRTP packet header
 - The list of SRTP and RTP streams taking part in a WebRTC session
 - SRTP stream analysis

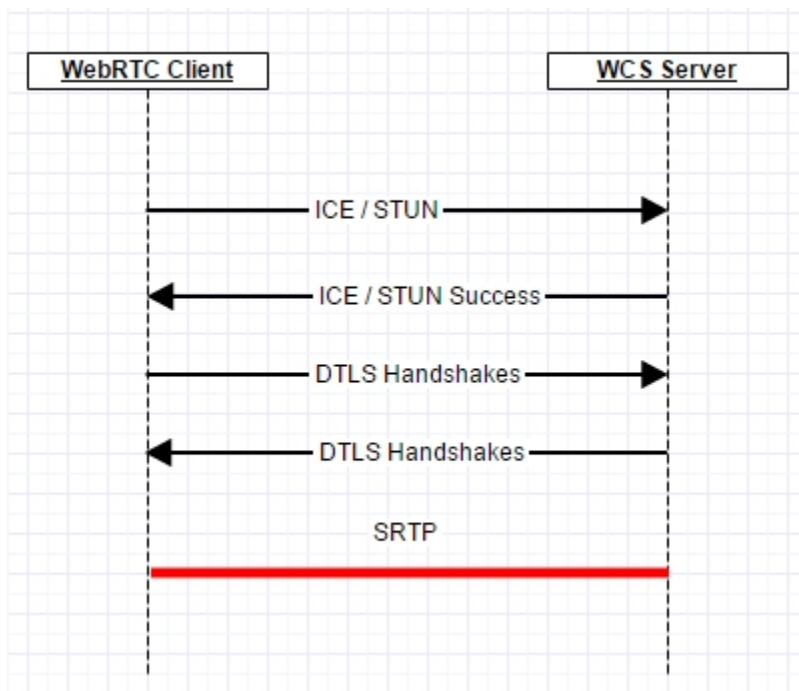
The technology

The WebRTC technology uses three main specifications in networking:

ICE and STUN
DTLS
SRTP

To establish a WebRTC connection, [ICE](#) is used. The web client sends [STUN](#)-requests to the WCS server, the WCS server responds to these requests and hence confirms it is ready to establish connection.

On the next stage, parties exchange SSL certificates via [DTLS](#) and establish an encrypted channel between the web client and the WCS server. When the connection is established, [SRTP](#) traffic is transmitted.



Possible problems

In most cases problems are related to UDP traffic of ICE, STUN, DTLS, SRTP not flowing between parts of the system.

Troubleshooting

Make sure all the traffic that takes part in establishing a WebRTC sessions and sending media data is unhindered and passes freely between call participants. Media ports of the WCS server in the range of [31000-32000] by default must be open to receive the incoming UDP traffic. If the WCS server is behind NAT and has an external IP address, make sure UDP packets sent to this external address are correctly routed to the corresponding ports of the WCS server behind NAT.

ICE and STUN traffic

Here is how ICE transactions exchange looks like before establishing connection. In the dump these are usual STUN requests and answers. After the connection is established, ICE keeps working to monitor the established connection. If monitoring indicates ICE does not pass, the call is interrupted.

logpcap [Wireshark 1.12.0 (v1.12.0-0-g4fab41a from master-1.12)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: stun Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
16	27.912936	92.127.221.146	188.40.69.75	STUN	146	Binding Request user: 9u13f:B28E8MfyfsvLgd58
17	27.921787	188.40.69.75	92.127.221.146	STUN	150	Binding Success Response XOR-MAPPED-ADDRESS: 92.127.221.146:58732 user: 9u13f:B28E8MfyfsvLgd58
18	27.929723	188.40.69.75	92.127.221.146	STUN	158	Binding Request user: 9u13f:B28E8MfyfsvLgd58:9u13f
19	28.018326	92.127.221.146	188.40.69.75	STUN	146	Binding Request user: 9u13f:B28E8MfyfsvLgd58
20	28.019742	188.40.69.75	92.127.221.146	STUN	150	Binding Success Response XOR-MAPPED-ADDRESS: 92.127.221.146:58732 user: 9u13f:B28E8MfyfsvLgd58
21	28.022820	188.40.69.75	92.127.221.146	STUN	158	Binding Request user: B28E8MfyfsvLgd58:9u13f
22	28.091373	92.127.221.146	188.40.69.75	STUN	106	Binding Success Response XOR-MAPPED-ADDRESS: 188.40.69.75:31030
24	28.123548	188.40.69.75	92.127.221.146	STUN	158	Binding Request user: B28E8MfyfsvLgd58:9u13f
25	28.137609	92.127.221.146	188.40.69.75	STUN	106	Binding Success Response XOR-MAPPED-ADDRESS: 188.40.69.75:31030
29	28.223897	92.127.221.146	188.40.69.75	STUN	106	Binding Success Response XOR-MAPPED-ADDRESS: 188.40.69.75:31030
34	28.330650	188.40.69.75	92.127.221.146	STUN	86	Binding Indication
85	28.572008	92.127.221.146	188.40.69.75	STUN	146	Binding Request user: 9u13f:B28E8MfyfsvLgd58
86	28.573147	188.40.69.75	92.127.221.146	STUN	150	Binding Success Response XOR-MAPPED-ADDRESS: 92.127.221.146:58732 user: 9u13f:B28E8MfyfsvLgd58
138	28.828370	188.40.69.75	92.127.221.146	TFTP	70	Read Request, File: d8;5\234\240e\n?351\340\345\305\2347b=r8[Malformed Packet]
262	29.448389	188.40.69.75	92.127.221.146	TFTP	101	Read Request[Malformed Packet]
280	29.534578	92.127.221.146	188.40.69.75	STUN	146	Binding Request user: 9u13f:B28E8MfyfsvLgd58
281	29.535797	188.40.69.75	92.127.221.146	STUN	150	Binding Success Response XOR-MAPPED-ADDRESS: 92.127.221.146:58732 user: 9u13f:B28E8MfyfsvLgd58
475	30.497059	92.127.221.146	188.40.69.75	STUN	146	Binding Request user: 9u13f:B28E8MfyfsvLgd58
476	30.497805	188.40.69.75	92.127.221.146	STUN	150	Binding Success Response XOR-MAPPED-ADDRESS: 92.127.221.146:58732 user: 9u13f:B28E8MfyfsvLgd58
647	31.330994	188.40.69.75	92.127.221.146	STUN	86	Binding Indication
647	31.457966	92.127.221.146	188.40.69.75	STUN	146	Binding Request user: 9u13f:B28E8MfyfsvLgd58
673	31.458722	188.40.69.75	92.127.221.146	STUN	150	Binding Success Response XOR-MAPPED-ADDRESS: 92.127.221.146:58732 user: 9u13f:B28E8MfyfsvLgd58
870	32.420408	92.127.221.146	188.40.69.75	STUN	146	Binding Request user: 9u13f:B28E8MfyfsvLgd58
871	32.421221	188.40.69.75	92.127.221.146	STUN	150	Binding Success Response XOR-MAPPED-ADDRESS: 92.127.221.146:58732 user: 9u13f:B28E8MfyfsvLgd58
1052	33.386004	92.127.221.146	188.40.69.75	STUN	146	Binding Request user: 9u13f:B28E8MfyfsvLgd58

Frame 16: 146 bytes on wire (1168 bits), 146 bytes captured (1168 bits)

0000 00 24 21 9c 37 ed 00 21 59 c5 74 e0 08 00 45 00 .\$.!7... Y.t...E.
 0010 00 84 7a 10 00 00 76 11 8e d3 5c 7f dd 92 bc 28 ..Z..V.. \....(
 0020 45 4b 05 6c 79 36 07 70 b5 00 01 00 54 21 12 EK.YV6.p C...T..
 0030 04 42 67 59 49 53 30 47 39 34 67 58 4d 67 00 06 .8yZ50g Y48kaQ..
 0040 00 16 39 75 31 33 66 34 42 32 38 45 38 4d 66 79 .9u13f: B28E8Mfy
 0050 72 66 56 46 67 64 53 28 00 80 22 00 00 4d 4d cfvLgd58 8 M

File: C:\tmp\3\logpcap 143 KB 00:00:33 Packets: 1057 - Displayed: 25 (2,4%) - Load time: 0:00:023 Profile: Default

DTLS traffic

DTLS starts working directly after ICE has established connection. Exchange of SSL certificates is several Handshake messages resulting in an established secure connection to transfer media data.

log.pcap [Wireshark 1.12.0 (v1.12.0-0-g4fab41a from master-1.12)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: dtls Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
23	28.093212	188.40.69.75	92.127.221.146	DTLSv1.	180	client Hello
26	28.137884	188.40.69.75	92.127.221.146	DTLSv1.	180	client Hello
27	28.196537	92.127.221.146	188.40.69.75	DTLSv1.	883	Server Hello, Certificate, Server Key Exchange, Certificate Request, Server Hello Done
28	28.216232	188.40.69.75	92.127.221.146	DTLSv1.	825	Certificate, Client Key Exchange, Certificate Verify, change Cipher spec, Encrypted Handshake Message
30	28.224506	188.40.69.75	92.127.221.146	DTLSv1.	825	Certificate, Client Key Exchange, Certificate Verify, change Cipher spec, Encrypted Handshake Message
31	28.322495	92.127.221.146	188.40.69.75	DTLSv1.	133	change Cipher Spec, Encrypted Handshake Message
32	28.324976	92.127.221.146	188.40.69.75	DTLSv1.	133	change Cipher Spec, Encrypted Handshake Message
1057	33.428850	92.127.221.146	188.40.69.75	DTLSv1.	103	Encrypted Alert

File: "C:\tmp\3\log.pcap" 143 kB 00:00:33 Packets: 1057 - Displayed: 8 (0,8%) - Load time: 0:00:020 Profile: Default

SRTTP traffic

Recognizing SRTP packets

When the data are sent via SRTP, Wireshark cannot recognize SRTP packets. Finding this packets is easy enough. By default, WCS uses the following port range to transmit media traffic [31000-32000], including WebRTC. In the dump we can see two unrecognized UDP packets, one of which is sent through the port 31030, and the second one is received to that port. For such packets, you should explicitly specify the protocol used.

log.pcap [Wireshark 1.12.0 (v1.12.0-0-g4fab41a from master-1.12)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

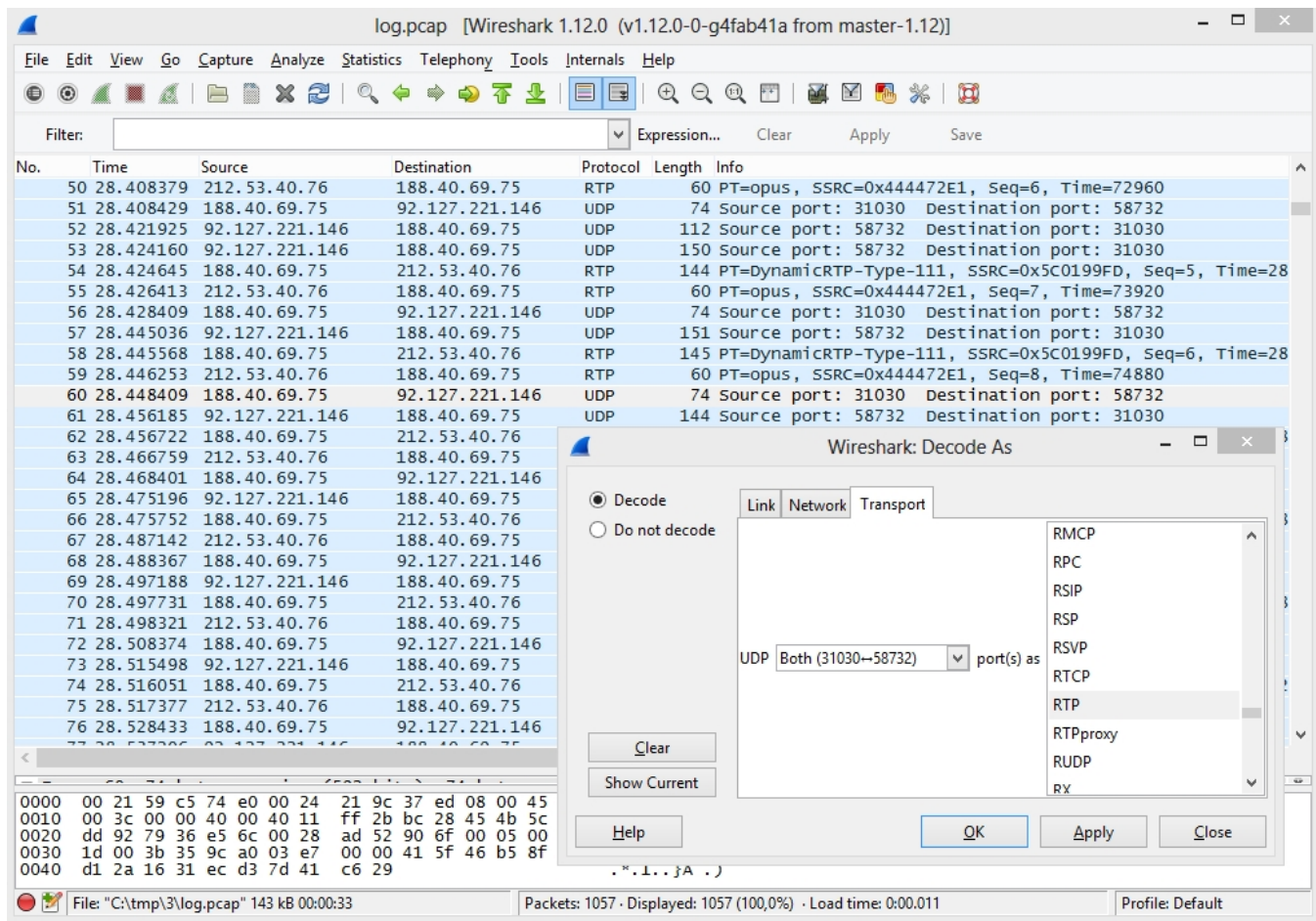
Filter: Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
50	28.408379	212.53.40.76	188.40.69.75	RTP	60	PT=opus, SSRC=0x444472E1, Seq=6, Time=72960
51	28.408429	188.40.69.75	92.127.221.146	UDP	74	Source port: 31030 Destination port: 58732
52	28.421925	92.127.221.146	188.40.69.75	UDP	112	Source port: 58732 Destination port: 31030
53	28.424160	92.127.221.146	188.40.69.75	UDP	150	Source port: 58732 Destination port: 31030
54	28.424645	188.40.69.75	212.53.40.76	RTP	144	PT=DynamicRTP-Type-111, SSRC=0x5C0199FD, Seq=5, Time=73000
55	28.426413	212.53.40.76	188.40.69.75	RTP	60	PT=opus, SSRC=0x444472E1, Seq=7, Time=73920
56	28.428409	188.40.69.75	92.127.221.146	UDP	74	Source port: 31030 Destination port: 58732
57	28.445036	92.127.221.146	188.40.69.75	UDP	151	Source port: 58732 Destination port: 31030
58	28.445568	188.40.69.75	212.53.40.76	RTP	145	PT=DynamicRTP-Type-111, SSRC=0x5C0199FD, Seq=6, Time=74000
59	28.446253	212.53.40.76	188.40.69.75	RTP	60	PT=opus, SSRC=0x444472E1, Seq=8, Time=74880
60	28.448409	188.40.69.75	92.127.221.146	UDP	74	Source port: 31030 Destination port: 58732
61	28.456185	92.127.221.146	188.40.69.75	UDP	144	Source port: 58732 Destination port: 31030
62	28.456722	188.40.69.75	212.53.40.76	RTP	138	PT=DynamicRTP-Type-111, SSRC=0x5C0199FD, Seq=7, Time=75000
63	28.466759	212.53.40.76	188.40.69.75	RTP	60	PT=opus, SSRC=0x444472E1, Seq=9, Time=75840
64	28.468401	188.40.69.75	92.127.221.146	UDP	74	Source port: 31030 Destination port: 58732
65	28.475196	92.127.221.146	188.40.69.75	UDP	150	Source port: 58732 Destination port: 31030
66	28.475752	188.40.69.75	212.53.40.76	RTP	144	PT=DynamicRTP-Type-111, SSRC=0x5C0199FD, Seq=8, Time=76000
67	28.487142	212.53.40.76	188.40.69.75	RTP	60	PT=opus, SSRC=0x444472E1, Seq=10, Time=76800
68	28.488367	188.40.69.75	92.127.221.146	UDP	74	Source port: 31030 Destination port: 58732
69	28.497188	92.127.221.146	188.40.69.75	UDP	144	Source port: 58732 Destination port: 31030
70	28.497731	188.40.69.75	212.53.40.76	RTP	138	PT=DynamicRTP-Type-111, SSRC=0x5C0199FD, Seq=9, Time=77000
71	28.498321	212.53.40.76	188.40.69.75	RTP	60	PT=opus, SSRC=0x444472E1, Seq=11, Time=77760
72	28.508374	188.40.69.75	92.127.221.146	UDP	74	Source port: 31030 Destination port: 58732
73	28.515498	92.127.221.146	188.40.69.75	UDP	141	Source port: 58732 Destination port: 31030
74	28.516051	188.40.69.75	212.53.40.76	RTP	135	PT=DynamicRTP-Type-111, SSRC=0x5C0199FD, Seq=10, Time=78000
75	28.517377	212.53.40.76	188.40.69.75	RTP	60	PT=opus, SSRC=0x444472E1, Seq=12, Time=78720
76	28.528433	188.40.69.75	92.127.221.146	UDP	74	Source port: 31030 Destination port: 58732

File: "C:\tmp\3\log.pcap" 143 kB 00:00:33 Packets: 1057 · Displayed: 1057 (100,0%) · Load time: 0:00.011 Profile: Default

Decoding SRTP packets

Wireshark can decode the UDP packets it found if we explicitly specify the protocol. In the packet properties select 'Decode As..', then select the RTP protocol for all packets that run between the browser (port 31030) and the WCS server (port 58732). These ports are reserved dynamically, so in your case the values might be different.



Decoded SRTP traffic

As a result of decoding the protocol, Wireshark will display the decoded SRTP traffic:

log.pcap [Wireshark 1.12.0 (v1.12.0-0-g4fab41a from master-1.12)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: Expression... Clear Apply Save

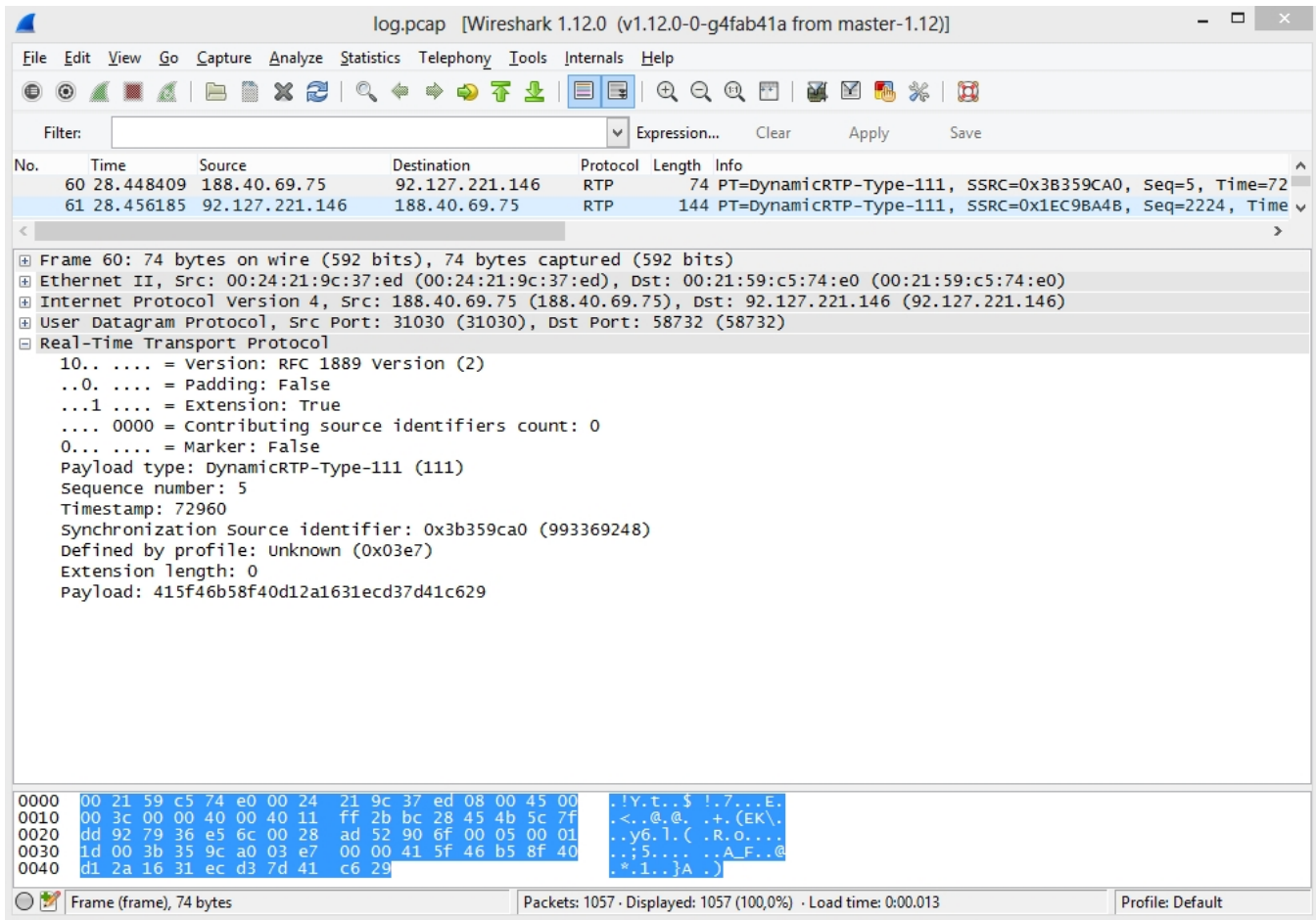
No.	Time	Source	Destination	Protocol	Length	Info
60	28.448409	188.40.69.75	92.127.221.146	RTP	74	PT=DynamicRTP-Type-111, SSRC=0x3B359CA0, Seq=5, Time=72
61	28.456185	92.127.221.146	188.40.69.75	RTP	144	PT=DynamicRTP-Type-111, SSRC=0x1EC9BA4B, Seq=2224, Time=72
62	28.456722	188.40.69.75	212.53.40.76	RTP	138	PT=DynamicRTP-Type-111, SSRC=0x5C0199FD, Seq=7, Time=28
63	28.466759	212.53.40.76	188.40.69.75	RTP	60	PT=opus, SSRC=0x444472E1, Seq=9, Time=75840
64	28.468401	188.40.69.75	92.127.221.146	RTP	74	PT=DynamicRTP-Type-111, SSRC=0x3B359CA0, Seq=6, Time=73
65	28.475196	92.127.221.146	188.40.69.75	RTP	150	PT=DynamicRTP-Type-111, SSRC=0x1EC9BA4B, Seq=2225, Time=73
66	28.475752	188.40.69.75	212.53.40.76	RTP	144	PT=DynamicRTP-Type-111, SSRC=0x5C0199FD, Seq=8, Time=28
67	28.487142	212.53.40.76	188.40.69.75	RTP	60	PT=opus, SSRC=0x444472E1, Seq=10, Time=76800
68	28.488367	188.40.69.75	92.127.221.146	RTP	74	PT=DynamicRTP-Type-111, SSRC=0x3B359CA0, Seq=7, Time=74
69	28.497188	92.127.221.146	188.40.69.75	RTP	144	PT=DynamicRTP-Type-111, SSRC=0x1EC9BA4B, Seq=2226, Time=74
70	28.497731	188.40.69.75	212.53.40.76	RTP	138	PT=DynamicRTP-Type-111, SSRC=0x5C0199FD, Seq=9, Time=28
71	28.498321	212.53.40.76	188.40.69.75	RTP	60	PT=opus, SSRC=0x444472E1, Seq=11, Time=77760
72	28.508374	188.40.69.75	92.127.221.146	RTP	74	PT=DynamicRTP-Type-111, SSRC=0x3B359CA0, Seq=8, Time=75
73	28.515498	92.127.221.146	188.40.69.75	RTP	141	PT=DynamicRTP-Type-111, SSRC=0x1EC9BA4B, Seq=2227, Time=75
74	28.516051	188.40.69.75	212.53.40.76	RTP	135	PT=DynamicRTP-Type-111, SSRC=0x5C0199FD, Seq=10, Time=2
75	28.517377	212.53.40.76	188.40.69.75	RTP	60	PT=opus, SSRC=0x444472E1, Seq=12, Time=78720
76	28.528433	188.40.69.75	92.127.221.146	RTP	74	PT=DynamicRTP-Type-111, SSRC=0x3B359CA0, Seq=9, Time=76
77	28.537206	92.127.221.146	188.40.69.75	RTP	146	PT=DynamicRTP-Type-111, SSRC=0x1EC9BA4B, Seq=2228, Time=76
78	28.537752	188.40.69.75	212.53.40.76	RTP	136	PT=DynamicRTP-Type-111, SSRC=0x5C0199FD, Seq=11, Time=2
79	28.539623	212.53.40.76	188.40.69.75	RTP	60	PT=opus, SSRC=0x444472E1, Seq=13, Time=79680
80	28.548413	188.40.69.75	92.127.221.146	RTP	74	PT=DynamicRTP-Type-111, SSRC=0x3B359CA0, Seq=10, Time=7
81	28.557717	212.53.40.76	188.40.69.75	RTP	60	PT=opus, SSRC=0x444472E1, Seq=14, Time=80640
82	28.561088	92.127.221.146	188.40.69.75	RTP	148	PT=DynamicRTP-Type-111, SSRC=0x1EC9BA4B, Seq=2229, Time=7
83	28.561589	188.40.69.75	212.53.40.76	RTP	138	PT=DynamicRTP-Type-111, SSRC=0x5C0199FD, Seq=12, Time=2
84	28.568348	188.40.69.75	92.127.221.146	RTP	70	PT=DynamicRTP-Type-111, SSRC=0x3B359CA0, Seq=11, Time=7
85	28.572008	92.127.221.146	188.40.69.75	STUN	146	Binding Request user: 9u13f:B28E8mfysfVLgdS8
86	28.573147	188.40.69.75	92.127.221.146	STUN	150	Binding Success Response XOR-MAPPED-ADDRESS: 92.127.221.146
87	28.573228	212.53.40.76	188.40.69.75	RTP	60	PT=opus, SSRC=0x444472E1, Seq=15, Time=81600

0000 00 21 59 c5 74 e0 00 24 21 9c 37 ed 08 00 45 00 .!Y.t..\$!.7...E.
0010 00 3c 00 00 40 00 40 11 ff 2b bc 28 45 4b 5c 7f .<..@.@. .+(EK).
0020 dd 92 79 36 e5 6c 00 28 ad 52 90 6f 00 05 00 01 ..y6.l.(.R.o....
0030 1d 00 3b 35 9c a0 03 e7 00 00 41 5f 46 b5 8f 40 ..;5... ..A.F..@
0040 d1 2a 16 31 ec d3 7d 41 c6 29 *.1..}A.)

File: "C:\tmp\3\log.pcap" 143 kB 00:00:33 Packets: 1057 · Displayed: 1057 (100,0%) · Load time: 0:00.013 Profile: Default

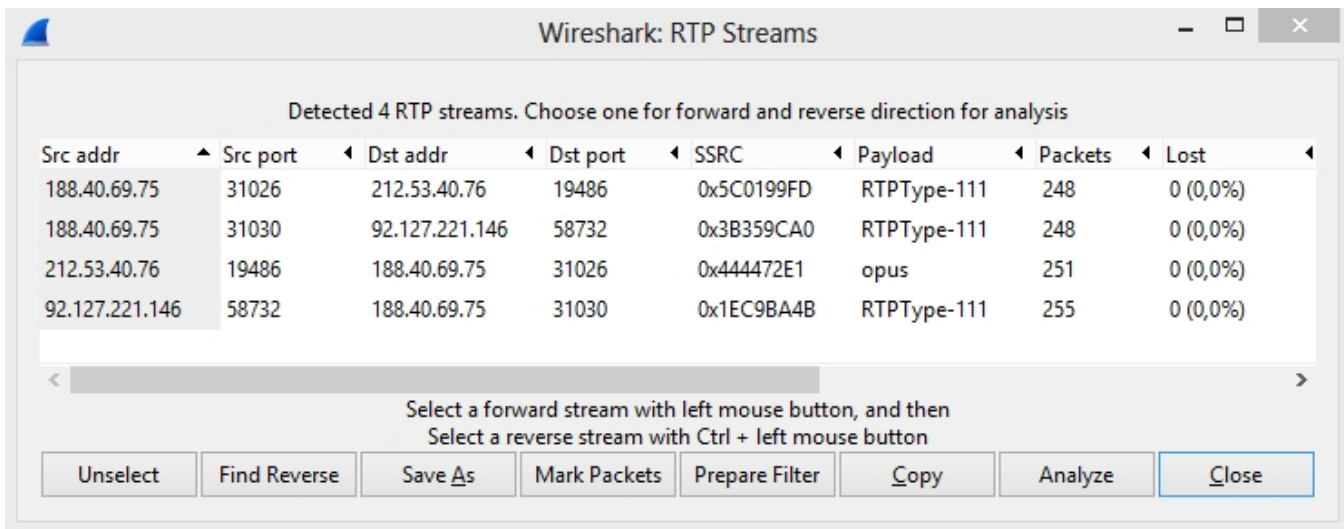
SRTP packet header

SRTP traffic is encrypted. This means if you try to play it, you will hear noises instead of normal speech. But only the content traffic is encrypted. The main RTP headers remain non-encrypted and they can be seen in the RTP packet. This is handy to analyze SRTP traffic parameters. The below example shows an SRTP packet with Payload Type, Sequence Number, Timestamp, SSRC.



The list of SRTP and RTP streams taking part in a WebRTC session

SRTP and RTP streams can be analyzed using Wireshark. To do this, use the 'Telephony - RTP - Show All Streams' menu.



In this case, streams with SSRC 0x3B359CA0 and 0x1EC9BA4B are SRTP streams between the web browser and WCS, because the source and destination address is the IP address of the web client (we know it beforehand). The other two streams, specifically, the first and the third ones from the top, are RTP streams between WCS and the SIP server (we know addresses of the WCS server and the SIP server too).

SRTP stream analysis

As described above, SRTP packet headers are not encrypted, so the SRTP stream is available for analysis of quality, losses, jitter, latency just like a conventional RTP stream:

Wireshark: RTP Streams

Wireshark: RTP Stream Analysis

Forward Direction

Reversed Direction

Analysing stream from 188.40.69.75 port 31030 to 92.127.221.146 port 58732 SSRC = 0x3B359CA0

Packet	Sequence	Delta(ms)	Filtered Jitter(ms)	Skew(ms)	IP BW(kbps)	Marker	Status
44	1	0,00	0,00	0,00	0,48		[Ok]
46	2	0,00	0,00	0,00	0,96		[Ok]
51	3	0,00	0,00	0,00	1,44		[Ok]
56	4	0,00	0,00	0,00	1,92		[Ok]
60	5	0,00	0,00	0,00	2,40		[Ok]
64	6	0,00	0,00	0,00	2,88		[Ok]
68	7	0,00	0,00	0,00	3,36		[Ok]
72	8	0,00	0,00	0,00	3,84		[Ok]

Max delta = 0,00 ms at packet no. 0
Max jitter = 0,00 ms. Mean jitter = 0,00 ms.
Max skew = 0,00 ms.
Total RTP packets = 248 (expected 248) Lost RTP packets = 0 (0,00%) Sequence errors = 0
Duration 4,88 s (0 ms clock drift, corresponding to 1 Hz (+0,00%))

Save payload...

Save as CSV...

Refresh

Jump to

Graph

Player

Next non-Ok

Close

Src addr

188.40.69.75

188.40.69.75

212.53.40.76

92.127.221.146

Unselect