THE PENTEST EXPERTS.

IT SECURITY KNOW-HOW

Moritz Abrell

NEW WAYS OF COMMUNICATING

– When End-to-End-Encryption Gains a New Meaning

June 2020

Introduction

Millions of people use different conference solutions every day. Lectures, work meetings, workshops or private audio/video chats are held. The rapid spread of COVID 19 has led to another enormous increase in the use of conference software and forced a large number of companies to take action. Where possible, employees have had to work at home and communication has been almost completely digitalized. The pressure to act quickly posed new technological challenges for many companies. In addition to remote access to company resources, digital communication from now on has primarily become a very complex subject area. The choice of a suitable solution is therefore very difficult. It is not easy to keep track on the seemingly infinite market for software solutions and functions. And last but not least, attention should also be paid to the security and data protection aspect together with audio/video and screencast functions. Manufacturers of different conference solutions therefore prefer advertisements containing statements such as "end-to-end encrypted" and "100% encrypted".

These promises raise the following questions: Are the data really safe from prying eyes? How far can marketing messages be trusted? What protection measures are available?

Using these questions as a basis, SySS GmbH conducted a research project in order to analyze four commercial or free conference solutions and their implementations for audio and video transmission. The results presented in this paper do not involve new vulnerabilities, but rather represent an introduction to the technologies, a study of encryption methods during online conferences and a comparison of the different solutions.

What is an online conference?

An online conference or a web conference is regarded as a meeting between participants in a virtual conference room which in most cases is organized by a moderator. The main functions of an online conference are audio and video transmission, as well as desktop sharing. In addition to these main functions, there are also other functions such as chats, file sharing, survey options, etc. depending on the particular product.

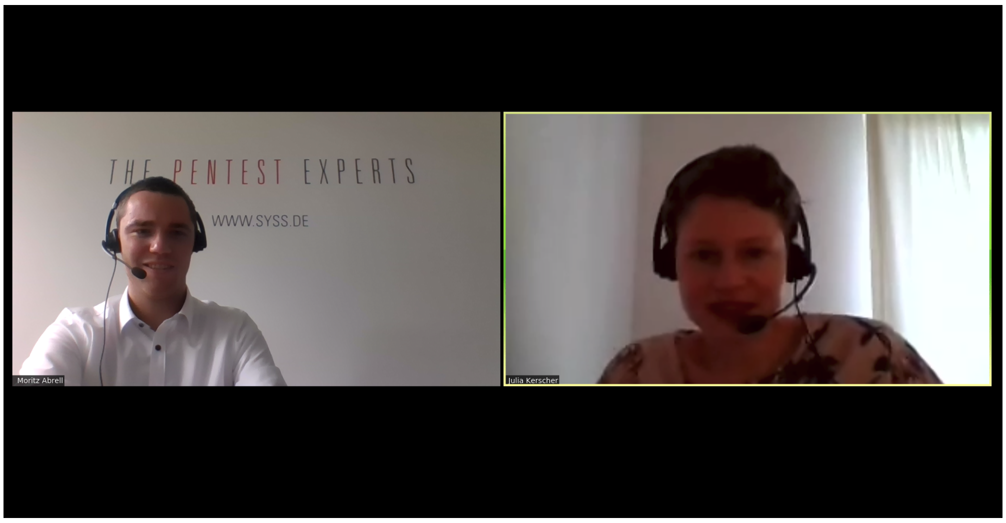


Figure 1: Online conference

A conference server is required in order to hold an online conference. This conference server can either be self-hosted or operated as a cloud service. The conference server represents the technological back end, performs the control function and guarantees the conferences' functions.

Once the moderator opens a virtual conference room, the participants can be connected to the conference server with their clients. This mostly takes place via a shared web link. Mobile apps, own PC software or supported browsers are used predominantly as clients.

In order to ensure audio and video transmission, a large number of solutions implement the open "WebRTC" [1] Standard. How WebRTC functions:

WebRTC – in technical terms

Web Real-Time communication (WebRTC) is an open standard which enables real-time communication and media transmission by means of a compatible browser. Communication in WebRTC is controlled directly by a web server through a JavaScript API, thus facilitating audio and video communication even without a plug-in. As an open-source project, WebRTC is being developed further primarily by the organizations World Wide Web Consortium (W3C) [2], Google [3], Mozilla [4] and Opera [5].

RTCPeerConnection

RTCPeerConnection is a JavaScript API within WebRTC that creates a connection to a remote destination. For this purpose, session-related data are sent to the destination in the form of a Session Description Protocol (SDP). Since the actual endpoints cannot still be directly reached at this time, these session data are transmitted in a session channel to the web server which then reroutes them to the destination. The recipient uses this so-called SDP offer to create a RTCPeerConnection object in the browser. The purpose of this is to be able to receive downstream user data of the sender. If this mechanism takes place bidirectionally, a direct bidirectional connection is also possible between the participants' browsers.

The following example shows an excerpt from a created SDP object:

```
v=0
o=mozilla...THIS_IS_SDPARTA-68.7.0 2362286823462031303 0 IN IP4 0.0.0.0
s=-
t=0 0
a=sendrecv
a=fingerprint:sha-256 FB:A6:A0:E1:C0:25:D1:02:6F:55:87:2D:ED:A9:58:83:5D:60:BA:BF:C1:B↩
    9:EB:DD:D5:2E:A4:D1:3E:88:4C:4B
a=group:BUNDLE 0 1
a=ice-options:trickle
a=msid-semantic:WMS *
m=audio 35673 UDP/TLS/RTP/SAVPF 109 101
c=IN IP4 192.168.178.22
a=candidate:0 1 UDP 2122252543 192.168.10.22 35673 typ host
a=candidate:1 1 UDP 2122187007 192.168.120.87 41127 typ host
a=candidate:2 1 UDP 2122121471 172.16.32.6 34890 typ host
a=recvonly
a=end-of-candidates
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level
a=extmap:3 urn:ietf:params:rtp-hdrext:sdes:mid
a=fmtp:109 maxplaybackrate=48000;stereo=1;useinbandfec=1
a=fmtp:101 0-15
a=ice-pwd:a0fb79579bc8b0c1a9da8985ccba3b0e
a=ice-ufrag:6d35217f
```

```
a=mid:0
a=rtcp-mux
a=rtpmap:109 opus/48000/2
a=rtpmap:101 telephone-event/8000
a=setup:active
a=ssrc:718341354 cname:{d54b1906-932a-4c13-bf35-d03699a1ce4b}
```

In addition to media attributes such as `a=rtpmap:109 opus/48000/2`, which indicate the Opus [6] codec with a clock frequency of 48 kHz as an audio codec, IP address information, so-called "candidates", or the fingerprint of the presented browser certificate are also generated and transmitted to the destination.

RTCDataChannel

A RTCDataChannel is used to receive and transmit data (audio/video streams and files) by means of a generated RTCPeerConnection object. Just like the RTCPeerConnection, the RTCDataChannel is mapped in a JavaScript API within WebRTC; similar to the WebSocket API [7], it uses event handlers. Received data can be transmitted to a generated RTCPeerConnection object in order, for example, to decode and play back an audio stream.

Transmission may take place reliably or unreliably. The two transmission protocols – Transmission Control Protocol (TCP) [8] and User Datagram Protocol (UDP) [9] – are used in this case.

Files are primarily transferred by means of TCP, and audio and video streams by means of UDP and the Real-Time Transport Protocol (RTP) [10]. This will ensure real-time communication with the lowest possible latency.

Interactive Connectivity Establishment

Interactive Connectivity Establishment (ICE) is a method for determining and administering nodes/candidates in order to establish a connection between the endpoints even if they are located behind a Network Address Translation (NAT) mechanism.The ICE method is not a new protocol, but uses already established protocols such as Session Traversal Utilities for the NAT (STUN) protocol [11] and the Traversal Using Relay NAT (TURN) protocol [12].

Figure 2 shows a connection between two endpoints which can reach each other directly as is the case, for example, in a common Local Area Network (LAN).

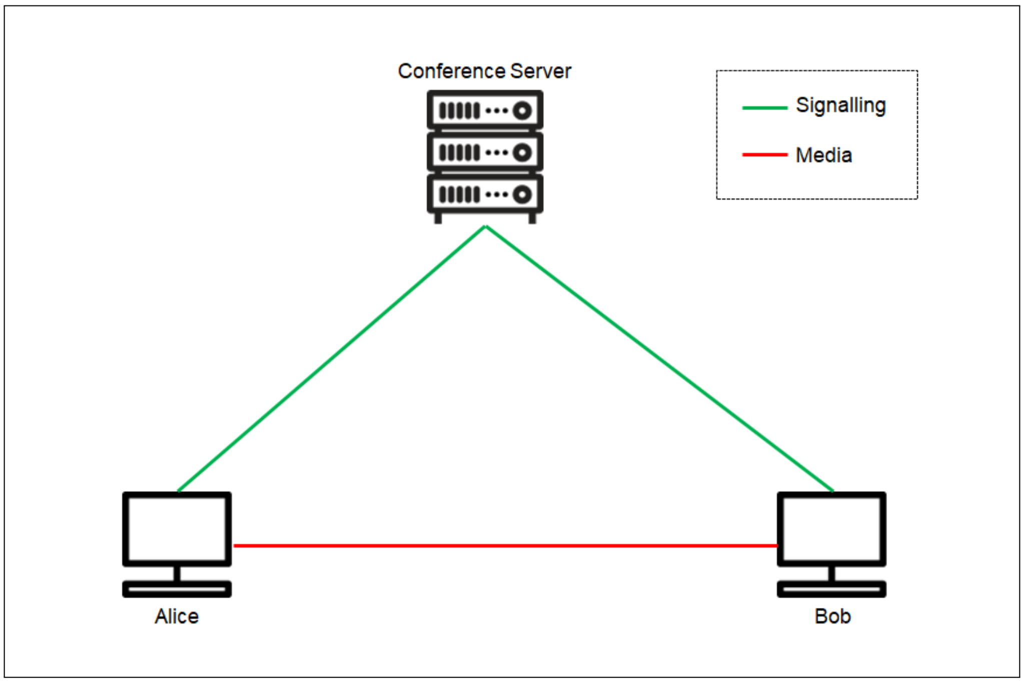


Figure 2: Direct connection between two endpoints

Figure 3 shows a connection between two endpoints which can reach each other by means of the previously determined external IP address information.

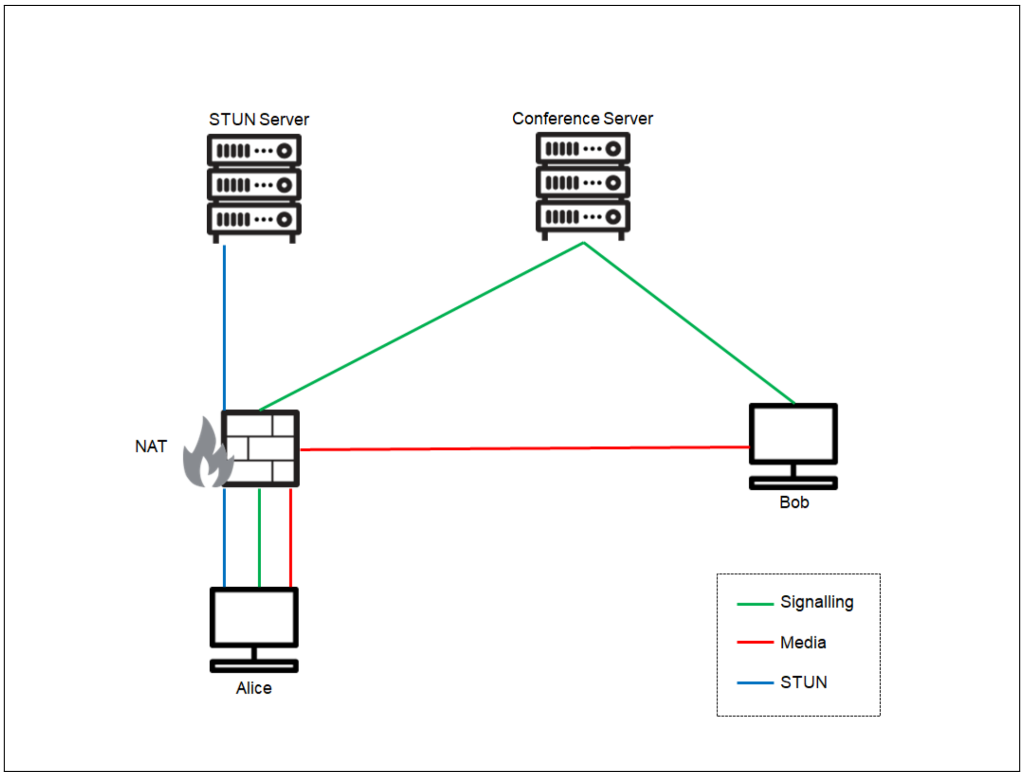


Figure 3: Direct connection between two endpoints in which the public IP/port pair was determined by means of STUN

For every outgoing connection, a "symmetric" NAT uses a separate IP/port pair even if the packet comes from the same client port. The client is therefore unable to predict the utilized IP/port pair for new connections even with the aid of STUN. If one of the involved participants is therefore behind a symmetric NAT, this prevents an incoming connection of another endpoint to the IP/port pair determined with the aid of STUN.

A TURN server is used as a media relay to solve this problem (see Figure 4).

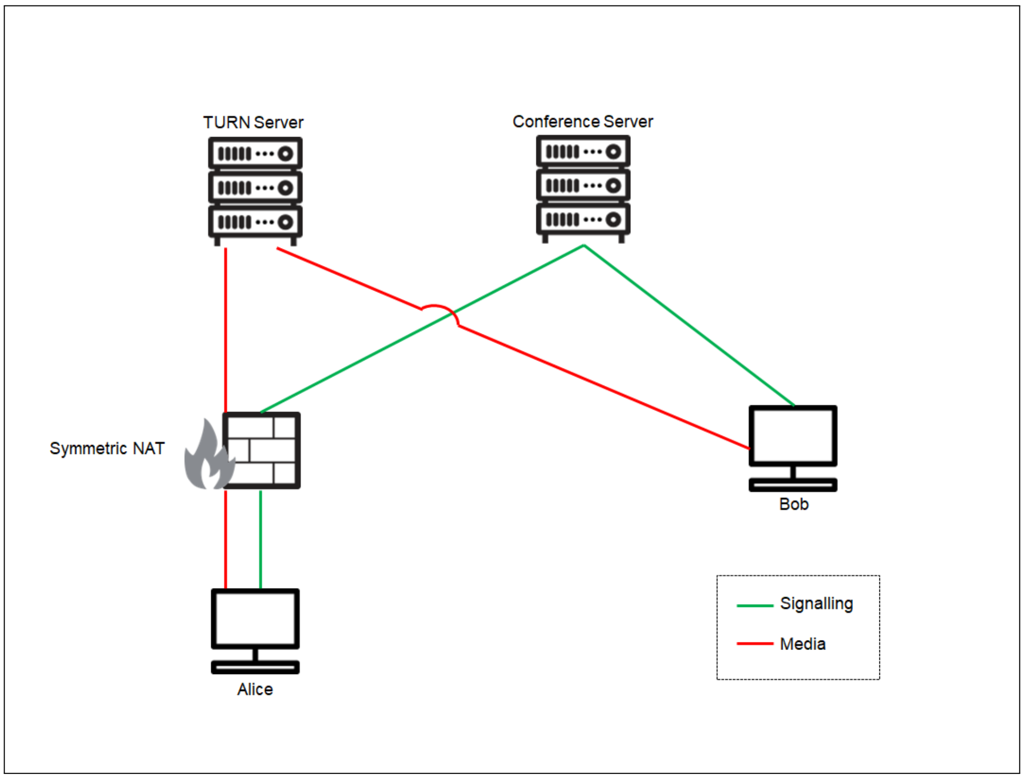


Figure 4: Connection between two endpoints via a TURN media relay server

All options are shown in the form of candidates in the SDP and are transmitted to the server.

```
Local IP/port candidate:
a=candidate:1 1 UDP 2122252543 192.168.10.98 59697 typ host

Public IP/port candidate determined by means of STUN
a=candidate:2 1 UDP 1694498815 77.47.111.245 16724 typ srflx

Configured TURN media relay server
a=candidate:3 1 UDP 8265727 3.127.240.102 50440 typ relay
```

The respective endpoints then determine the best possible connection using the received information.

Encryption of media data

Media data are encrypted in order to ensure the confidentiality of media streams. In order to prevent third parties from seeing the data, the data are primarily protected by means of the Secure Real-Time Transport Protocol (SRTP) [13] and the AES [14] encryption method. Two variants are used for key exchange when WebRTC technology is utilized: the SRTP-SDES method and the SRTP-DTLS method.

SRTP-SDES

During a Security Descriptions for Media Streams (SDES) key exchange, the AES key to be used is shown as a media attribute in the SDP and is transmitted to the server.

```
a=crypto:1 AES_CM_128_HMAC_SHA1_32 inline:CbTBosdVUZqEb7Htqhn+m3z7cUh4RJVR8nA15GbN
```

Figure 5 shows the topology of a session in which the AES key was negotiated for the SRTP data using the SDES method.

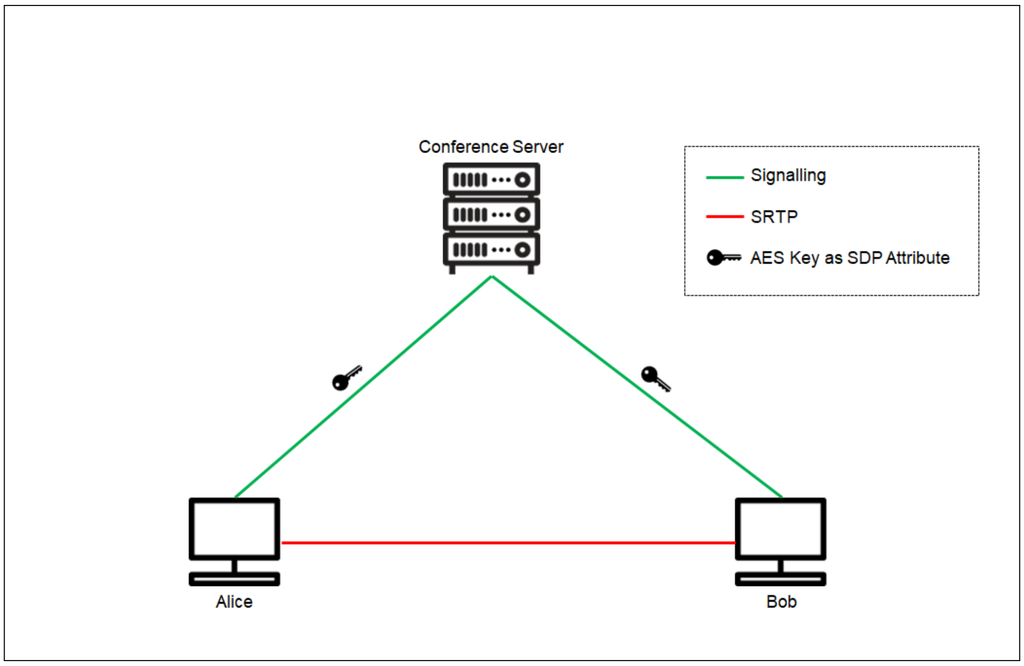


Figure 5: Topology of a SRTP-SDES session

Since the AES key in the SDED method is transferred as a media attribute in the SDP, every connection point in a session recognizes the key and the media stream is therefore not end-to-end encrypted (E2EE) [15].

This leads to two security-related problems:

- If an operator of the conference server has access to the SRTP data, he can decrypt and intercept data using the AES key.
- If an attacker gains access to the content of the SDP – e.g. using a TLS proxy –, he can also decrypt and intercept the SRTP stream.

Since end-to-end encryption is not possible with this method, it is only used to a very limited extent. In individual cases, certain implementations may offer SRTP-SDES to ensure downwards compatibility.

SRTP-DTLS

Unlike the SDES method, the key in Datagram Transport Layer Security (DTLS) [16] key exchange is not exchanged in the SDP across several points, but in a secured channel directly between the endpoints.

DTLS is similar to the asynchronous encryption method Transport Layer Security (TLS) [17] and is implemented in every modern browser.

In order to verify the identity of the opposite endpoint – e.g of the browser –, the fingerprint transmitted as the SDP attribute is used as the basis for verification.

Figure 6 shows the topology of a session in which the AES key was negotiated for the SRTP data using the DTLS method.

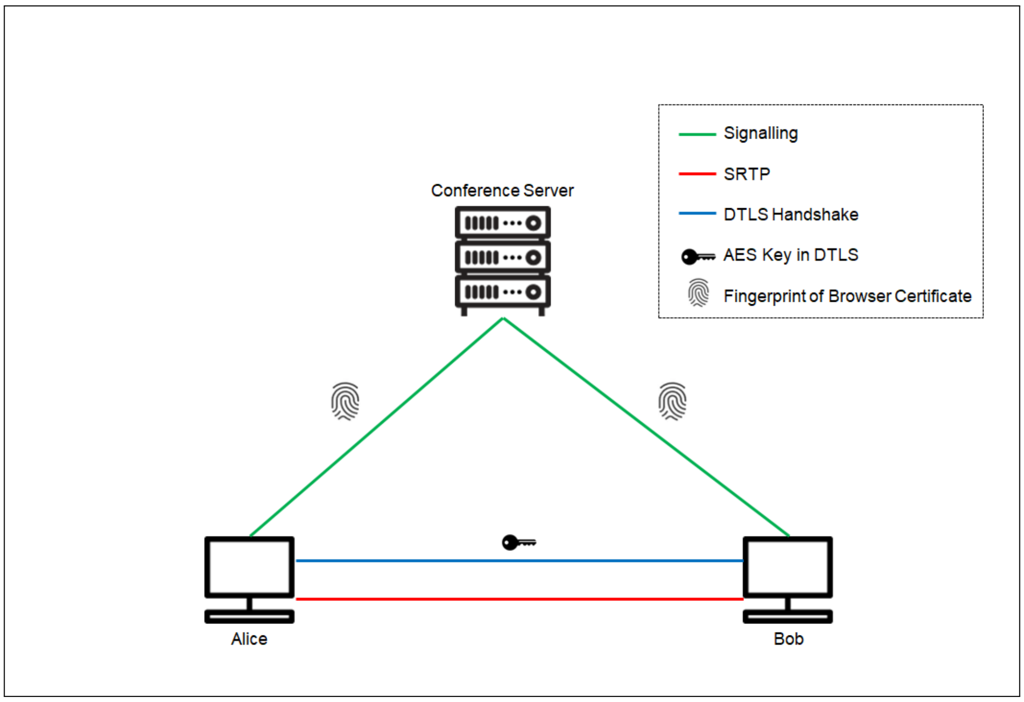


Figure 6: Topology of a SRTP-DTLS session

Since the AES key in this method is known only to the respective media endpoints, this constitutes end-to-end encryption.

Although the use of SRTP-DTLS offers greater protection compared with SRTP-SDES, this method is also not protected against active man-in-the-middle attacks. An attacker, who has access to the transport-encrypted signaling stream, could therefore actively manipulate the fingerprint of the certificates, for example, in order to attack the media stream using a DTLS proxy.

The WebRTC specification [18] stipulates that SRTP-DTLS is required as a key management procedure for WebRTC implementations. It is generally recognized that DTLS-SRTP should be the obligatory and standard option for encrypting WebRTC media.

Peer-to-Peer

A conference with two participants can also be described as a Peer-to-Peer (P2P) connection. In an ideal scenario, the conference server in a P2P conference acts as a signaling unit and media are transmitted directly between the two endpoints.

In the best case, it can be assumed that end-to-end encryption is actually implemented for the media stream since this exists directly between the two endpoints with the aid of ICE, and key exchange also takes place between the actual conference participants through SRTP-DTLS.

Peer-to-Multipeer

As soon as a conference exceeds the limit of two participants, this is called a Peer-to-Multipeer (P2MP) connection. Different media transmission methods are used so that all participants can communicate with one another.

Mesh

In mesh architecture, every participant sends the media stream to all other participants and receives it from all other participants. Since the total amount of bandwidth required for meshing increases quadratically with every additional participant, this architecture is limited to a few participants and is therefore not widespread.

Figure 7 shows the structure of mesh architecture.

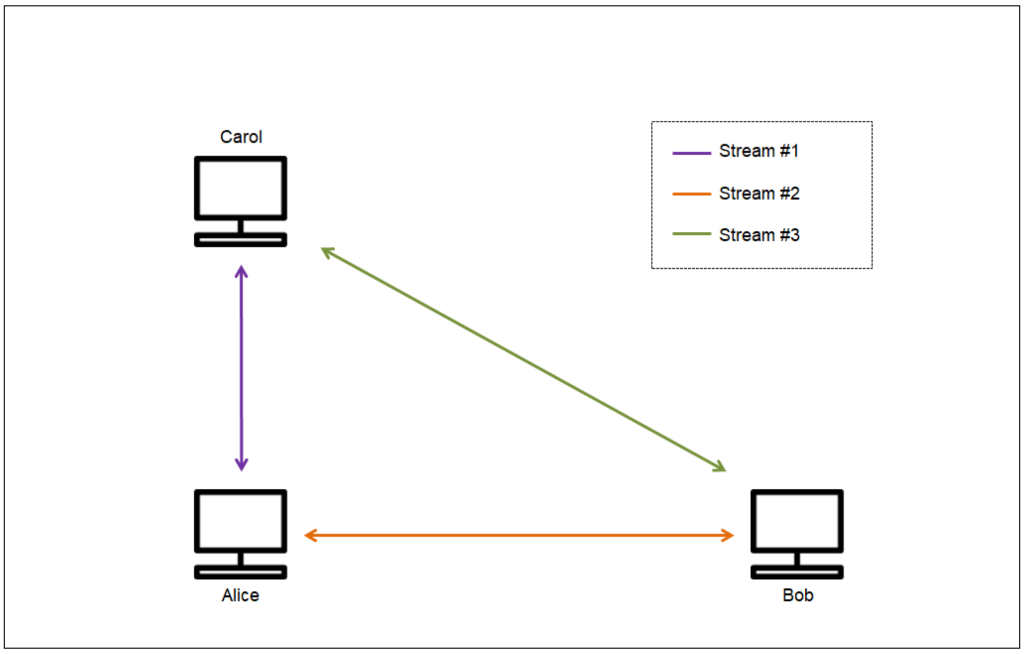


Figure 7: Mesh architecture

The advantage of meshing lies in the possibility of end-to-end encryption of the media stream between the participants.

Multipoint Control Unit

A Multipoint Control Unit (MCU) is a service which mixes the media streams of all participants into a single stream so that all participants can communicate with one another. Although a MCU requires high computing power on account of decoding, mixing and coding of the streams, the bandwidth needed for the participants is reduced substantially compared with meshing. In this method, the maximum size of a conference primarily depends on the utilized MCU and its resources.

Figure 7 shows the media streams in a WebRTC conference using a MCU integrated in the conference server.

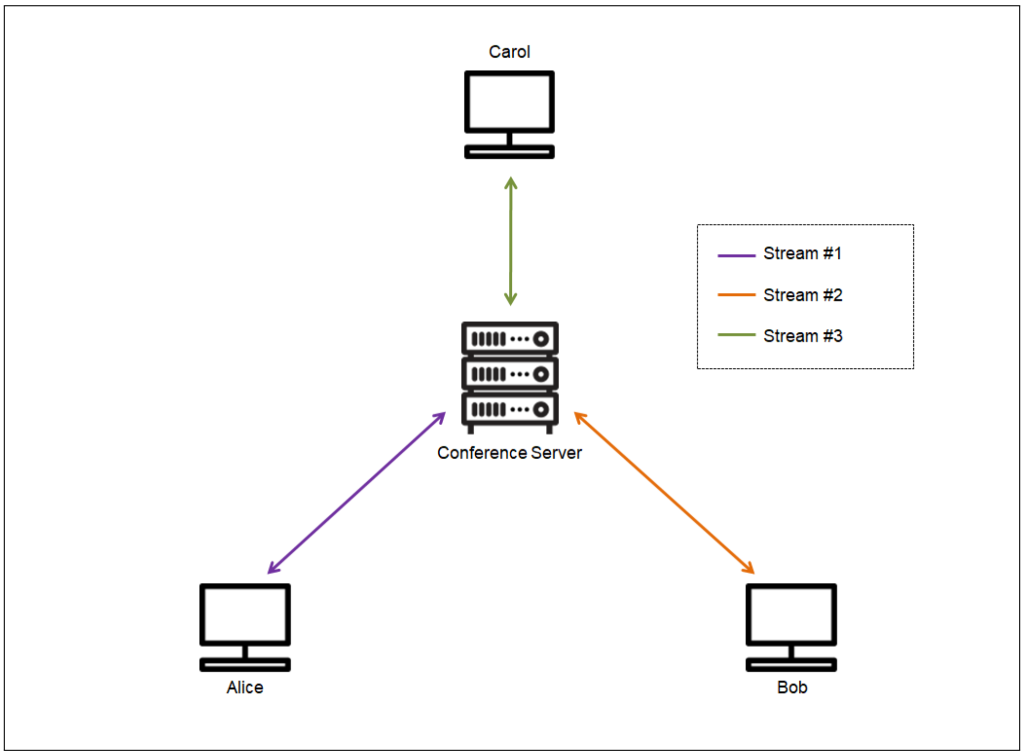


Figure 8: MCU architecture

The MCU is regarded as the respective media endpoint, i.e. key exchange also takes place between the participants and the MCU. Encryption is therefore only possible between the participants and the MCU. This means that the media streams are only hop-by-hop (HbH) encrypted rather than fully end-to-end encrypted between the participants.

Selective Forwarding Unit

A Selective Forwarding Unit (SFU) can be used to reduce the load on the server. A SFU is capable of receiving several media streams and then deciding which of these media streams should be sent to which participants. Media data can therefore be routed without having to decode, mix and recode them beforehand.

Figure 9 shows the media streams in a WebRTC conference which is routed using a SFU.

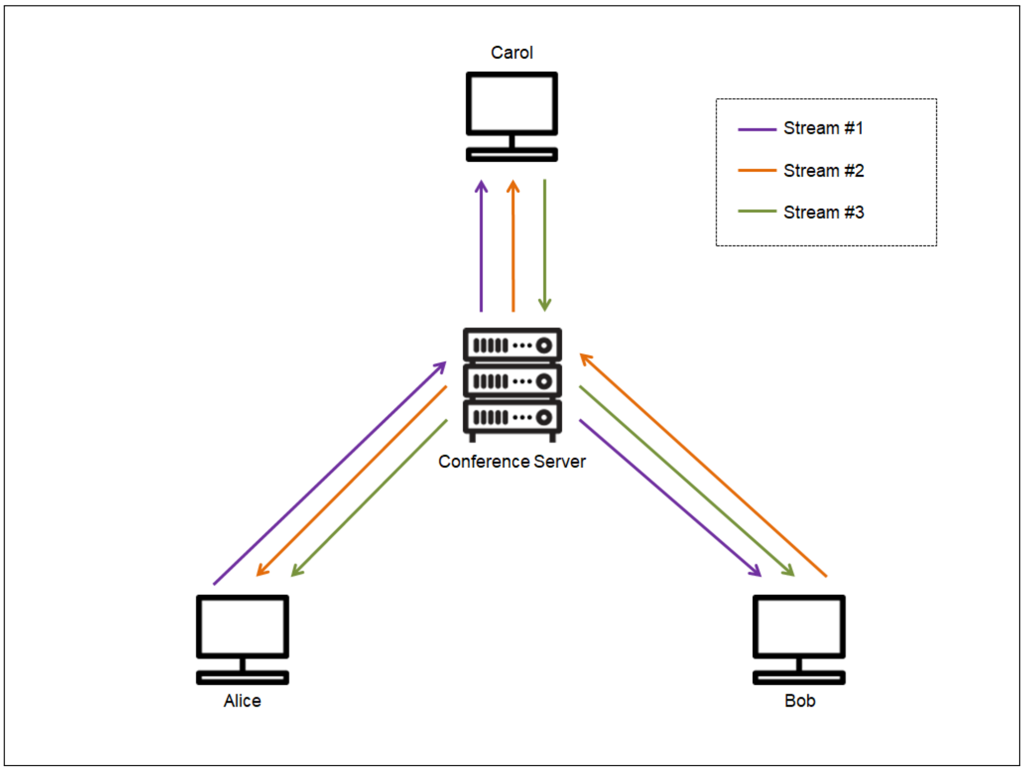


Figure 9: SFU architecture

Just like when a MCU is used, the SFU is the media endpoint in this application. According to the latest state of the art, the media data are hop-to-hop encrypted instead of fully end-to-end encrypted.

Security problems with online conferences and WebRTC

Some vulnerabilities and security problems occur very frequently with online conferences and WebRTC. The problems described in this section do not relate to implementation vulnerabilities, but rather to security problems which are due to architectures or specifications.

Overhearing of media streams

As soon as a conference is staged via a MCU or a SFU, this enables the operator of the conference server or of the MCU or the SFU to overhear the audio and video data. Since the particular MCU or SFU acts as the media endpoint, it also knows all AES keys of the media streams and the SRTP packets contained therein.

In addition to the operator, a potentially compromised server may also represent a corresponding risk. Since conference systems are often exposed on the public Internet, the attack area is increased and it is much more likely that a vulnerability or a configuration fault will be exploited by an attacker.

Exposure of IP addresses

Since all candidates are collected and transmitted to the server when ICE is used, the opposite endpoint receives information about public and private IP addresses.

This means, for example, that the local IP addresses of a participant are made known. For example, information relating to the local Wi-Fi network, the client network or the VPN network is divulged.

If the media endpoint is, for example, a participant in a P2P conference or a meshed P2MP conference, the information is transmitted to this participant via the IP addresses. In addition to the data protection aspect, the information also contains lucrative knowledge about a potential victim and his corresponding networks.

Inadequate data protection in encrypted media streams

The SRTP protocol encrypts the user data of a media stream, but not the header and the associated metadata of a RTP packet. The header contains information whose confidentiality may be desirable.

Everyone able to see SRTP packets can recognize whether or not a participant is speaking at a specific point in time. It is possible, for example, to find out how often a participant speaks during a conference in order to draw conclusions regarding potentially discussed topics.

An attacker in a suitable man-in-the-middle (MitM) position need only record the SRTP packets and then analyze them. This problem arises both with P2P conferences and P2MP conferences.

Comparison of conference systems

This section describes the results of an analysis of different free and commercial conference systems. The conference systems were examined according to the following aspects:

- **Different clients**: In order to analyze possible differences between various clients and browsers, the tests were mainly conducted using a 64-bit Firefox browser (version 68.7.0) and a 64-bit Google Chrome browser (version 81.0.4044.129) from the official build. In addition to the browsers, the respective officially available mobile applications on an Android device and an iOS device were also used.
- **P2P audio/video encryption**: After considering different client combinations, an analysis was carried out to determine whether a Peer-to-Peer audio/video (AV) connection is end-to-end encrypted or hop-by-hop encrypted.
- **P2MP audio/video encryption**: In addition to an analysis of a P2P AV connection, encryption with a Peer-to-Multipeer (P2MP) connection was also examined each time.
- **Private chat and group chat encryption**: Since a large number of conference systems enable chat messages to be transmitted, tests were carried out each time to determine whether group chats and private chats are end-to-end encrypted.
- **IP exposure**: The problem of IP exposure in the direction of the respective server and the other participants was analyzed.
- **STUN server**: A STUN server could collect utilization data. The STUN servers used for the respective systems are therefore shown.
- **Encryption symbol**: We discussed the question of whether an end user with little technical know-how can determine whether and especially how the conference and the related media data are protected.
- **DTLS cipher/SRTP cipher**: The negotiated DTLS ciphers and SRTP ciphers were determined.
- **Active media stream without any other participants**: If an AV stream is already transmitted to the server without any other participants when entering a conference, the server can overhear the participant while he is still waiting, for example, for other participants. This potential behavior was examined during the tests.

- **Active media stream after participants leave**: The question of whether a media stream remains active after all other participants (apart from one) have left the conference was also examined during the tests.

Overview of the test results

The test results are shown in the following table:

	Open Rainbow	Jitsi	Microsoft Teams	Zoom
P2P Firefox <> Firefox AV	E2EE	HbH	not supported	HbH
P2P Firefox <> Chrome AV	E2EE	HbH	not supported	HbH
P2P Firefox <> Mobile App AV	E2EE	HbH	not supported	HbH
P2P Chrome <> Chrome AV	E2EE	E2EE	HbH	HbH
P2P Chrome <> Mobile App AV	E2EE	E2EE	HbH	HbH
P2P Mobile App <> Mobile App AV	E2EE	E2EE	HbH	HbH
P2MP AV	HbH	HbH	HbH	HbH
P2MP > P2P > back to E2EE	No	Yes	*N.A.*	*N.A.*
Private Chat E2EE	No	No	No	Not in Meeting
Group Chat E2EE	No	No	No	Not in Meeting
IP Leak to Server	Yes	Yes	Yes	No
IP Leak to Participants	Yes - at P2P	Yes - at P2P	No	No
STUN Server	France OpenRainbow	Configurable / AWS	AWS	not implemented
Encryption Symbol	No	Yes	No	Yes
DTLS Cipher	TLS_ECDHE_ECDSA-AES_128_GCM_SHA256	TLS_ECDHE_ECDSA-AES_128_GCM_SHA256	SRTP-SDES	TLS_ECDHE_RSA-AES_128_GCM_SHA256
SRTP Cipher	AES_CM_128_HMAC_SHA1_80	AES_CM_128_HMAC_SHA1_80	AES_CM_128_HMAC_SHA1_80	AES_ECB_256
Stream without Participants	Yes - in Bubble	No	Yes	Yes
Stream after leaving Participants	Yes - in Bubble	Yes 20 seconds	Yes	Yes

Table 1: Comparison of conference systems

Open Rainbow

Open Rainbow [19] is a commercial conference solution from the company Alcatel-Lucent Enterprise. In order to use Open Rainbow, it is necessary to create an account to access the Open Rainbow platform used by the operator.

P2MP conferences and entertainment take place in so-called "bubbles" to which participants can be invited and added.

Negative test results:

– Open Rainbow cannot be operated in its own infrastructure.

– If a conference takes place from a bubble, the media streams are hop-by-hop encrypted. This applies both to P2P bubble conferences and P2MP bubble conferences.

– If a bubble conference starts with more than two participants and all of them except two leave the conference, the media stream of the two remaining participants is hop-by-hop encrypted.

– Chat messages are transmitted to the server in a WebSocket connection.

– Private IP addresses are sent to the server and in a P2P connection to the conference participants. Since Open Rainbow contains a call function, it is possible to call a contact. If this contact answers the call, all private IP addresses are transmitted.

– There is an active media stream to the server in a bubble without other participants being dialed up.

Jitsi

Jitsi [20], a conference solution compiled from several open-source projects [21], contains all current functions of an online conference. Jitsi is very popular and can be fully operated in your own infrastructure.

There are also various freely available and externally hosted Jitsi instances on the Internet. One of the most popular freely available instances is the instance provided by the developer team itself – accessible under the URL `https://meet.jit.si`.

In order to analyze Jitsi, we examined both the public Jitsi instance of the developer and a self-hosted instance which was applied by means of the recommended "Quick Installation" [22] method.

Negative test results:

– If a participant in a P2P conferences uses a Firefox browser, the media streams are hop-by-hop encrypted.

– A P2MP conference is always hop-by-hop encrypted.

– Chat messages are sent to the server as a HTTP POST request.

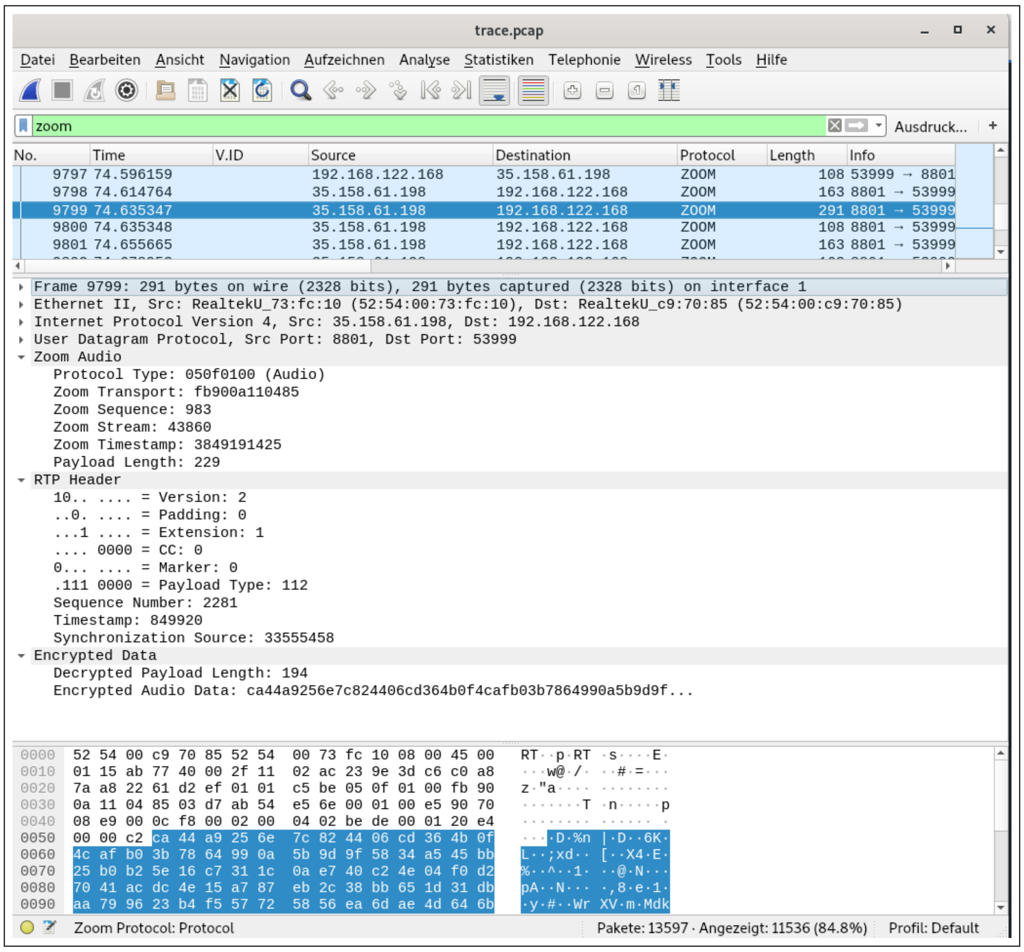
– Private IP addresses are made known to the server. With a P2P conference, private IP addresses are made known to the participants.

– If all other participants leave the conference, the media stream to the server is maintained for around 20 seconds.

Zoom

Zoom is a commercial conference solution which enjoyed a real user boom over night, especially on account of the COVID 19 pandemic. The platform is developed and provided by the company of the same name Zoom Inc. [23]. In addition to browser integration for all current operating systems, there are native clients which contain extended functions, e.g. a virtual background.

Zoom uses its own proprietary protocol. This protocol was analyzed in more detail. In order to make the analysis easier, a Wireshark Lua dissector was created. Figure 10 shows a Zoom audio packet.


trace.pcap
Datei Bearbeiten Ansicht Navigation Aufzeichnen Analyse Statistiken Telephonie Wireless Tools Hilfe
zoom
Ausdruck...

```
No.   Time          V.ID   Source            Destination       Protocol  Length Info
 9797 74.596159            192.168.122.168   35.158.61.198     ZOOM         108 53999 → 8801
 9798 74.614764            35.158.61.198     192.168.122.168   ZOOM         163 8801 → 53999
 9799 74.635347            35.158.61.198     192.168.122.168   ZOOM         291 8801 → 53999
 9800 74.635348            35.158.61.198     192.168.122.168   ZOOM         108 8801 → 53999
 9801 74.655665            35.158.61.198     192.168.122.168   ZOOM         163 8801 → 53999
```

```
▸ Frame 9799: 291 bytes on wire (2328 bits), 291 bytes captured (2328 bits) on interface 1
▸ Ethernet II, Src: RealtekU_73:fc:10 (52:54:00:73:fc:10), Dst: RealtekU_c9:70:85 (52:54:00:c9:70:85)
▸ Internet Protocol Version 4, Src: 35.158.61.198, Dst: 192.168.122.168
▸ User Datagram Protocol, Src Port: 8801, Dst Port: 53999
▾ Zoom Audio
    Protocol Type: 050f0100 (Audio)
    Zoom Transport: fb900a110485
    Zoom Sequence: 983
    Zoom Stream: 43860
    Zoom Timestamp: 3849191425
    Payload Length: 229
▾ RTP Header
    10.. .... = Version: 2
    ..0. .... = Padding: 0
    ...1 .... = Extension: 1
    .... 0000 = CC: 0
    0... .... = Marker: 0
    .111 0000 = Payload Type: 112
    Sequence Number: 2281
    Timestamp: 849920
    Synchronization Source: 33555458
▾ Encrypted Data
    Decrypted Payload Length: 194
    Encrypted Audio Data: ca44a9256e7c824406cd364b0f4cafb03b7864990a5b9d9f...
```

```
0000  52 54 00 c9 70 85 52 54  00 73 fc 10 08 00 45 00   RT··p·RT ·s····E·
0010  01 15 ab 77 40 00 2f 11  02 ac 23 9e 3d c6 c0 a8   ···w@·/· ··#·=···
0020  7a a8 22 61 d2 ef 01 01  c5 be 05 0f 01 00 fb 90   z·"a···· ········
0030  0a 11 04 85 03 d7 ab 54  e5 6e 00 01 00 e5 90 70   ·······T ·n·····p
0040  08 e9 00 0c f8 00 02 00  04 02 be de 00 01 20 e4   ········ ······ ·
0050  00 00 c2 ca 44 a9 25 6e  7c 82 44 06 cd 36 4b 0f   ····D·%n |·D··6K·
0060  4c af b0 3b 78 64 99 0a  5b 9d 9f 58 34 a5 45 bb   L··;xd·· [··X4·E·
0070  25 b0 b2 5e 16 c7 31 1c  0a e7 40 c2 4e 04 f0 d2   %··^··1· ··@·N···
0080  70 41 ac dc 4e 15 a7 87  eb 2c 38 bb 65 1d 31 db   pA··N··· ·,8·e·1·
0090  aa 79 96 23 b4 f5 57 72  58 56 ea 6d ae 4d 64 6b   ·y·#··Wr XV·m·Mdk
```

```
Zoom Protocol: Protocol          Pakete: 13597 · Angezeigt: 11536 (84.8%)   Profil: Default
```

Figure 10: Zoom audio packet with a Wireshark dissector

As also determined by Bill Marczak and John Scott-Railton in their article dated April 3, 2020 [24], the encrypted audio data can be separated since these data are applied as an additional frame to Zoom's own protocol. Tests of the encryption utilized for the user data revealed that AES is used in the ECB mode with a key length of 256 bit. This is confirmed in Zoom's Encryption White Paper [25]; a change has now been made to AES in the GCM mode with a key length of 256 bit.

In addition to the Wireshark dissector, a suitable script was created that permits decryption of the encrypted RTP data using the AES key. A suitable SILK decoder [26] can then be used to convert the SILK data into a MP3 format.

Negative test result

- When Zoom is used in the browser, no end-to-end encryption takes place.
- Chat messages during a conference are transferred to the server using base64 encoding.
- User data are encrypted in the ECB mode.
- There is a media stream to the server if the conference only (still) comprises one participant.

Microsoft Teams

Microsoft Teams, the direct successor of Skype for Business, is probably the best known conference solution for companies. In addition to conferencing, Teams offers many more collaboration functions and a wide range of upgradable plug-ins. As the product name implies, Teams was developed and is provided by Microsoft [28].

The main point to note here is the use of the SRTP-SDES method through which the AES key for encryption of media data is contained in plain text in the SDP object. With Microsoft Teams, the SDP object is transmitted to the Microsoft web server in a HTTP POST request. The AES key is therefore known not only to the respective media endpoints, but also additional servers which are independent of the media session.

Negative test result:

– No end-to-end encryption takes place.

– Chat messages are sent to the server as a HTTP POST request in plain text.

– The SRTP-SDES method is used.

– The AES key is sent to the web server as a HTTP POST request. The AES key is therefore known to several servers.

– There is an active media stream to the server if the conference only (still) comprises one participant.

– Private IP addresses are made known to the server.

Recommendation

Since, as things stand, it is not practical when using a SFU or MCU to also encrypt media data end-to-end in a P2MP conference, it is recommended that confidential data be exchanged solely via systems on your own command.

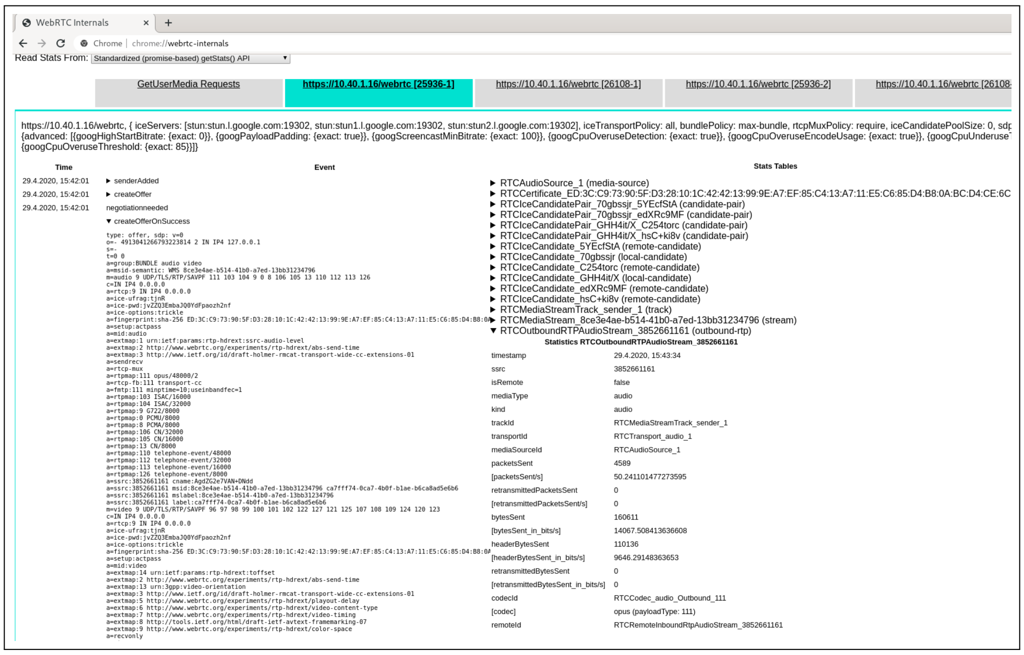
Check which external systems are used to transfer confidential data.

Ensure that your P2P conference is also actually end-to-end encrypted by examining the RTCPeerConnection objects in the browser.

The internal WebRTC tools in the browser are the first contact point for controlling a WebRTC conference:

- Firefox: `about:webrtc`
- Google Chrome: `chrome://webrtc-internals`

The provided browser tools contain detailed information regarding the respective RTCPeerConnection objects. Figure 11 shows a WebRTC session which is described in detail by means of Google Chrome WebRTC tools.


WebRTC Internals
Chrome | chrome://webrtc-internals
Read Stats From: Standardized (promise-based) getStats() API
GetUserMedia Requests
https://10.40.1.16/webrtc [25936-1]
https://10.40.1.16/webrtc [26108-1]
https://10.40.1.16/webrtc [25936-2]
https://10.40.1.16/webrtc [26108-

https://10.40.1.16/webrtc, { iceServers: [stun:stun.l.google.com:19302, stun:stun1.l.google.com:19302, stun:stun2.l.google.com:19302], iceTransportPolicy: all, bundlePolicy: max-bundle, rtcpMuxPolicy: require, iceCandidatePoolSize: 0, sdp
{advanced: [{googHighStartBitrate: {exact: 0}}, {googPayloadPadding: {exact: true}}, {googScreencastMinBitrate: {exact: 100}}, {googCpuOveruseDetection: {exact: true}}, {googCpuOveruseEncodeUsage: {exact: true}}, {googCpuUnderuse
{googCpuOveruseThreshold: {exact: 85}}]}

Time	Event
29.4.2020, 15:42:01	▶ senderAdded
29.4.2020, 15:42:01	▶ createOffer
29.4.2020, 15:42:01	negotiationneeded
	▼ createOfferOnSuccess

```
type: offer, sdp: v=0
o=- 4913041266793223814 2 IN IP4 127.0.0.1
s=-
t=0 0
a=group:BUNDLE audio video
a=msid-semantic: WMS 8ce3e4ae-b514-41b0-a7ed-13bb31234796
m=audio 9 UDP/TLS/RTP/SAVPF 111 103 104 9 0 8 106 105 13 110 112 113 126
c=IN IP4 0.0.0.0
a=rtcp:9 IN IP4 0.0.0.0
a=ice-ufrag:tjnR
a=ice-pwd:jvZZQ3EmbaJQ0YdFpaozh2nf
a=ice-options:trickle
a=fingerprint:sha-256 ED:3C:C9:73:90:5F:D3:28:10:1C:42:42:13:99:9E:A7:EF:85:C4:13:A7:11:E5:C6:85:D4:B8:0A
a=setup:actpass
a=mid:audio
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level
a=extmap:2 http://www.webrtc.org/experiments/rtp-hdrext/abs-send-time
a=extmap:3 http://www.ietf.org/id/draft-holmer-rmcat-transport-wide-cc-extensions-01
a=sendrecv
a=rtcp-mux
a=rtpmap:111 opus/48000/2
a=rtcp-fb:111 transport-cc
a=fmtp:111 minptime=10;useinbandfec=1
a=rtpmap:103 ISAC/16000
a=rtpmap:104 ISAC/32000
a=rtpmap:9 G722/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:106 CN/32000
a=rtpmap:105 CN/16000
a=rtpmap:13 CN/8000
a=rtpmap:110 telephone-event/48000
a=rtpmap:112 telephone-event/32000
a=rtpmap:113 telephone-event/16000
a=rtpmap:126 telephone-event/8000
a=ssrc:3852661161 cname:AgdZG2e7VAN+DNdd
a=ssrc:3852661161 msid:8ce3e4ae-b514-41b0-a7ed-13bb31234796 ca7fff74-0ca7-4b0f-b1ae-b6ca8ad5e6b6
a=ssrc:3852661161 mslabel:8ce3e4ae-b514-41b0-a7ed-13bb31234796
a=ssrc:3852661161 label:ca7fff74-0ca7-4b0f-b1ae-b6ca8ad5e6b6
m=video 9 UDP/TLS/RTP/SAVPF 96 97 98 99 100 101 102 122 127 121 125 107 108 109 124 120 123
c=IN IP4 0.0.0.0
a=rtcp:9 IN IP4 0.0.0.0
a=ice-ufrag:tjnR
a=ice-pwd:jvZZQ3EmbaJQ0YdFpaozh2nf
a=ice-options:trickle
a=fingerprint:sha-256 ED:3C:C9:73:90:5F:D3:28:10:1C:42:42:13:99:9E:A7:EF:85:C4:13:A7:11:E5:C6:85:D4:B8:0A
a=setup:actpass
a=mid:video
a=extmap:14 urn:ietf:params:rtp-hdrext:toffset
a=extmap:2 http://www.webrtc.org/experiments/rtp-hdrext/abs-send-time
a=extmap:13 urn:3gpp:video-orientation
a=extmap:3 http://www.ietf.org/id/draft-holmer-rmcat-transport-wide-cc-extensions-01
a=extmap:5 http://www.webrtc.org/experiments/rtp-hdrext/playout-delay
a=extmap:6 http://www.webrtc.org/experiments/rtp-hdrext/video-content-type
a=extmap:7 http://www.webrtc.org/experiments/rtp-hdrext/video-timing
a=extmap:8 http://tools.ietf.org/html/draft-ietf-avtext-framemarking-07
a=extmap:9 http://www.webrtc.org/experiments/rtp-hdrext/color-space
a=recvonly
```

Stats Tables

- ▶ RTCAudioSource_1 (media-source)
- ▶ RTCCertificate_ED:3C:C9:73:90:5F:D3:28:10:1C:42:42:13:99:9E:A7:EF:85:C4:13:A7:11:E5:C6:85:D4:B8:0A:BC:D4:CE:6C
- ▶ RTCIceCandidatePair_70gbssjr_5YEcfStA (candidate-pair)
- ▶ RTCIceCandidatePair_70gbssjr_edXRc9MF (candidate-pair)
- ▶ RTCIceCandidatePair_GHH4it/X_C254torc (candidate-pair)
- ▶ RTCIceCandidatePair_GHH4it/X_hsC+ki8v (candidate-pair)
- ▶ RTCIceCandidate_5YEcfStA (remote-candidate)
- ▶ RTCIceCandidate_70gbssjr (local-candidate)
- ▶ RTCIceCandidate_C254torc (remote-candidate)
- ▶ RTCIceCandidate_GHH4it/X (local-candidate)
- ▶ RTCIceCandidate_edXRc9MF (remote-candidate)
- ▶ RTCIceCandidate_hsC+ki8v (remote-candidate)
- ▶ RTCMediaStreamTrack_sender_1 (track)
- ▶ RTCMediaStream_8ce3e4ae-b514-41b0-a7ed-13bb31234796 (stream)
- ▼ RTCOutboundRTPAudioStream_3852661161 (outbound-rtp)

Statistics RTCOutboundRTPAudioStream_3852661161

timestamp	29.4.2020, 15:43:34
ssrc	3852661161
isRemote	false
mediaType	audio
kind	audio
trackId	RTCMediaStreamTrack_sender_1
transportId	RTCTransport_audio_1
mediaSourceId	RTCAudioSource_1
packetsSent	4589
[packetsSent/s]	50.241101477273595
retransmittedPacketsSent	0
[retransmittedPacketsSent/s]	0
bytesSent	160611
[bytesSent_in_bits/s]	14067.508413636608
headerBytesSent	110136
[headerBytesSent_in_bits/s]	9646.29148363653
retransmittedBytesSent	0
[retransmittedBytesSent_in_bits/s]	0
codecId	RTCCodec_audio_Outbound_111
[codec]	opus (payloadType: 111)
remoteId	RTCRemoteInboundRtpAudioStream_3852661161

Figure 11: Display of a WebRTC session by means of Google Chrome WebRTC tools

In addition to WebRTC tools, network sniffers such as Wireshark can also be used to analyze the systems and servers on which data are sent.

Conclusion

The described analysis shows that none of the examined conference systems offers complete end-to-end encryption. In some cases, key exchange depends on the utilized client. It therefore makes a real difference whether a user joins a conference with the native mobile application, a Firefox browser or a Chrome browser.

Due to the lack of end-to-end encryption, there is a danger at present that third parties can eavesdrop on the media streams. During a conference, messages are exchanged using the frequently integrated chat function. Since these messages are also not end-to-end encrypted in the analyzed systems, they do not have adequate protection against prying eyes.

In addition to data confidentiality, the handling of sensitive data – e.g. a local VPN IP address – is unsatisfactory. Attackers can therefore also collect information at times about the local IP address configuration by simply calling a contact. The operators of the server instances can also collect data relating to the users. Users could therefore be partially profiled by analyzing the public and local IP addresses at certain periods of time.

The analysis also shows that it is difficult for end users to recognize how the media data in a conference are protected and whether these data are end-to-end encrypted.

Outlook

The Google Chrome developer team is currently working on the implementation of so-called "insertable streams" [29]. Using this API, WebRTC applications can access already coded media data before they enter the network. SRTP double encryption procedures [30] can therefore be utilized to re-encrypt the user data of the media streams in order to also ensure confidentiality beyond Selective Forwarding Units.

For this purpose, the developers at Jitsi have already integrated a proof-of-concept function [31] which enables end-to-end encryption of data. The key for the second encryption level is still currently distributed via a third communication channel in the demonstration function, e.g. via an encrypted e-mail. The developers have stated that they are working on suitable key administration.

Sources

[1] webrtc.org, https://webrtc.org/ 2

[2] w3.org, https://www.w3.org/ 2

[3] google.de, https://about.google/intl/ 2

[4] mozilla.org, https://www.mozilla.org/ 2

[5] opera.com, https://www.opera.com/ 2

[6] JM. Valin, RFC 6716, tools.ietf.org, https://tools.ietf.org/html/rfc6716, 2012-09 3

[7] I. Fette, RFC 6455, tools.ietf.org, https://tools.ietf.org/html/rfc6455, 2011-12 3

[8] D. Borman, RFC 7323, tools.ietf.org, https://tools.ietf.org/html/rfc7323, 2014-09 3

[9] J. Postel, RFC 768, tools.ietf.org, https://tools.ietf.org/html/rfc768, 1980-08-28 3

[10] H. Schulzrinne, RFC 3550, tools.ietf.org, https://tools.ietf.org/html/rfc3550, 2003-07 3

[11] J. Rosenberg, RFC 5389, tools.ietf.org, https://tools.ietf.org/html/rfc5389, 2008-10 3

[12] R. Mahy, RFC 5766, tools.ietf.org, https://tools.ietf.org/html/rfc5766, 2010-04 3

[13] M. Baugher, RFC 3711, tools.ietf.org, https://tools.ietf.org/html/rfc3711, 2004-03 5

[14] wikipedia.org, https://de.wikipedia.org/wiki/Advanced_Encryption_Standard 5

[15] wikipedia.org, https://en.wikipedia.org/wiki/End-to-end_encryption 6

[16] E. Rescorla, RFC 6347, tools.ietf.org, https://tools.ietf.org/html/rfc6347, 2012-01 6

[17] E. Rescorla, RFC 8446, tools.ietf.org, https://tools.ietf.org/html/rfc8446, 2018-08 6

[18] tools.ietf.org, https://tools.ietf.org/html/draft-ietf-rtcweb-rtp-usage-25, 2015-07-28 7

[19] openrainbow.com, https://www.openrainbow.com 14

[20] jitsi.org, https://jitsi.org 14

[21] github.com/jitsi, https://github.com/jitsi/ 14

[22] github.com/jitsi, https://github.com/jitsi/jitsi-meet/blob/master/doc/quick-install.md, 2020-04-10 14

[23] zoom.us, https://zoom.us/ 14

[24] Bill Marczak and John Scott-Railton, Move Fast and Roll Your Own Crypto, citizenlab.ca, https://citizenlab.ca/2020/04/move-fast-roll-your-own-crypto-a-quick-look-at-the-confidentiality-of-zoom-meetings/ 2020-04-03 15

[25] Zoom Encyrption Whitepaper, zoom.us, https://zoom.us/docs/doc/Zoom Encryption Whitepaper.pdf, 2020-04 15

[26] kn007, silk-v3-decoder, github.com, https://github.com/kn007/silk-v3-decoder 15

[27] teams.micorsoft.com, https://teams.microsoft.com/

[28] microsoft.com `https://www.microsoft.com/de-de/` 15

[29] chromestatus.com, `https://www.chromestatus.com/feature/6321945865879552`, 2020-04-26 17

[30] tools.ietf.org, `https://tools.ietf.org/id/draft-ietf-perc-double-10.html`, 2018-10-17 17

[31] Emil Ivov, jitsi.org, `https://jitsi.org/blog/e2ee/`, 2020-04-12 17

THE PENTEST EXPERTS

SySS GmbH   72072 Tübingen   Germany   +49 (0)7071 - 40 78 56-0   info@syss.de

WWW.SYSS.DE