THE PENTEST EXPERTS.

IT SECURITY KNOW-HOW

Moritz Abrell

DIE KOMMUNIKATION GEHT NEUE WEGE

Ende-zu-Ende-Verschlüsselung erhält eine neue Bedeutung

Juni 2020

Schaffhausenstraße 77, 72072 Tübingen, Deutschland

+49 (0)7071 - 40 78 56-0

info@syss.de

www.syss.de

Hinführung

Millionen Menschen nutzen täglich verschiedene Konferenzlösungen. Es werden Vorlesungen, Arbeitsmeetings, Workshops oder auch private Audio/Video-Chats durchgeführt. Die rasante Ausbreitung von COVID 19 erhöhte diese Nutzung von Konferenzsoftware nochmals enorm und zwang viele Unternehmen zu handeln. Mitarbeiter wurden, wenn möglich, ins Homeoffice geschickt und die Kommunikation wurde nahezu vollständig digitalisiert. Der Zwang, schnell zu agieren, stellte viele Unternehmen vor neue technologische Herausforderungen. Neben Fernzugriffen auf Firmenressourcen ist vor allem die fortan digitale Kommunikation ein sehr komplexes Themengebiet und die Entscheidung für eine entsprechende Lösung fällt schwer. Auf dem scheinbar unendlichen Markt an Softwarelösungen und -Funktionen ist es nicht leicht, den Überblick zu behalten. Und nicht zuletzt sollte neben Audio/Video- und Screencastfunktionalitäten auch der Sicherheits- und Datenschutzaspekt Beachtung finden. Die Hersteller verschiedener Konferenzlösungen werben dabei bevorzugt mit Aussagen wie „Ende-zu-Ende-verschlüsselt" und „100% verschlüsselt".

Solche Versprechen werfen folgende Fragen auf: Sind die Daten wirklich vor Einblicken Dritter geschützt? Inwieweit ist den Marketingaussagen zu trauen? Welche Schutzmaßnahmen existieren?

Auf der Basis dieser Fragen analysierte die SySS GmbH im Kontext eines Forschungsprojekts vier kommerzielle bzw. freie Konferenzlösungen und deren Implementierungen zur Audio- und Videoübertragung. Bei den in diesem Artikel präsentierten Ergebnissen handelt es sich nicht um neue Schwachstellen, sondern um eine Einführung in die Technologien, eine Untersuchung zur Verschlüsselungsmethodik bei Online-Konferenzen sowie einen Vergleich der unterschiedlichen Lösungen.

Was ist eine Online-Konferenz?

Unter einer Online- oder auch Webkonferenz versteht man eine Zusammenkunft von Teilnehmern in einem virtuellen Konferenzraum, der meist von einem Moderator organisiert ist. Audio- und Videoübertragung sowie Desktop-Sharing sind die Hauptfunktionalitäten einer Online-Konferenz. Neben diesen Hauptfunktionalitäten gibt es je nach Produkt auch weitere Funktionen wie z.B. Chats, Filesharing, Umfrageoptionen, etc.

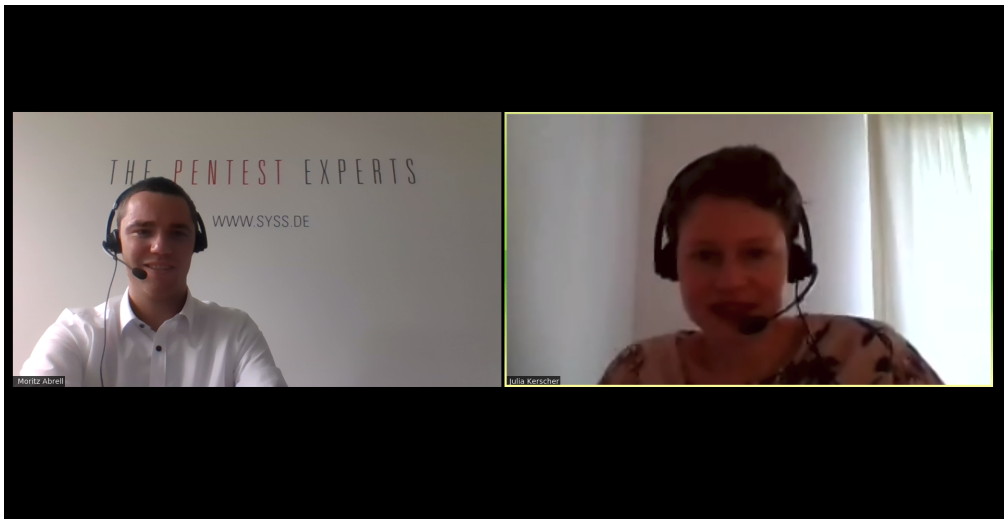

Abbildung 1: Online-Konferenz

Um eine Online-Konferenz durchzuführen, wird ein Konferenzserver benötigt. Dieser kann selbst gehosted oder als Cloud-Service betrieben werden. Er stellt das technologische Back-End dar, übernimmt die Steuerung und gewährleistet die Funktionen der Konferenzen.

Eröffnet der Moderator einen virtuellen Konferenzraum, so können sich die Teilnehmer mit ihren Clients zum Konferenzserver verbinden. Meist erfolgt dies über einen geteilten Weblink. Als Clients kommen überwiegend Mobile Apps, eigene PC-Software oder auch unterstützte Browser zum Einsatz.

Um die Audio- und Videoübertragung zu gewährleisten, implementieren viele Lösungen den offenen Standard „WebRTC" [1]. Und so funktioniert WebRTC:

WebRTC – technisch betrachtet

Web Real-Time Communication (WebRTC) ist ein offener Standard, der eine Echtzeitkommunikation und Medienübertragung mithilfe eines kompatiblen Browsers ermöglicht. Die Kommunikation wird bei WebRTC direkt über eine JavaScript-API von einem Webserver gesteuert und ermöglicht so eine Audio- und Videokommunikation auch ohne Plug-in. WebRTC ist ein Open Source-Projekt und wird vor allem von den Organisationen World Wide Web Consortium (W3C) [2], Google [3], Mozilla [4] und Opera [5] weiterentwickelt.

RTCPeerConnection

RTCPeerConnection ist eine JavaScript-API innerhalb von WebRTC, die es ermöglicht, eine Verbindung zu einem entfernten Ziel aufzubauen. Hierfür werden Session-bezogene Daten in Form des Session Description Protocol (SDP) an das Ziel gesendet. Da sich zu diesem Zeitpunkt die tatsächlichen Endpunkte noch nicht direkt erreichen können, werden diese Sessiondaten in einem Sessionchannel an den Webserver gesendet, der diese wiederum an das Ziel weiterleitet. Der Empfänger nutzt dieses sogenannte SDP Offer, um ein RTCPeerConnection-Objekt innerhalb des Browsers zu erzeugen. Dies geschieht zu dem Zweck, nachfolgende Nutzdaten des Senders empfangen zu können. Erfolgt dieser Mechanismus bidirektional, so ist auch eine direkte bidirektionale Verbindung zwischen den Browsern der Teilnehmer möglich.

Das nachfolgende Beispiel zeigt einen Auszug eines erzeugten SDP-Objekts:

```
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}
```

Neben Medienattributen wie `a=rtpmap:109 opus/48000/2`, die den Opus [6]-Codec mit einer Taktfrequenz von 48kHz als Audiocodec angeben, werden auch IP-Adress-Informationen, sogenannte „Kandidaten", oder auch der Fingerprint des präsentierten Browserzertifikats generiert und an das Ziel übermittelt.

RTCDataChannel

Ein RTCDataChannel wird verwendet, um Daten (Audio- sowie Videostreams und Dateien) mithilfe eines erstellten RTCPeerConnection-Objekts empfangen und senden zu können. Wie die RTCPeerConnection ist der RTCDataChannel in einer JavaScript-API innerhalb von WebRTC abgebildet; ähnlich wie die WebSocket-API [7] verwendet er Eventhandler. Empfangene Daten können an ein erstelltes RTCPeerConenction-Objekt übergeben werden, um z. B. einen Audiostream zu decodieren und abzuspielen.

Die Übertragung kann zuverlässig oder auch unzuverlässig stattfinden. Dabei kommen die beiden Übertragungsprotokolle Transmission Control Protocol (TCP) [8] und User Datagram Protocol (UDP) [9] zum Einsatz.

Dateien werden meist mittels TCP übertragen, Audio- und Videostreams hingegen mittels UDP und dem Real-Time Transport Protocol (RTP) [10]. Dies soll eine Echtzeitkommunikation mit möglichst geringer Latenz ermöglichen.

Interactive Connectivity Establishment

Interactive Connectivity Establishment (ICE) ist ein Verfahren zur Ermittlung und Verwaltung von Knoten/Kandidaten, um so eine Verbindung zwischen den Endpunkten zu ermöglichen, auch wenn sich diese hinter einem Network Address Translation (NAT)-Mechanismus befinden. Das ICE-Verfahren ist kein neues Protokoll, sondern nutzt bereits etablierte Protokolle wie das Session Traversal Utilities for NAT (STUN)-Protokoll [11] und das Traversal Using Relay NAT (TURN)-Protokoll [12].

Abbildung 2 zeigt eine Verbindung zwischen zwei Endpunkten, die sich direkt erreichen können, wie es z. B. in einem gemeinsamen Local Area Network (LAN) möglich ist.

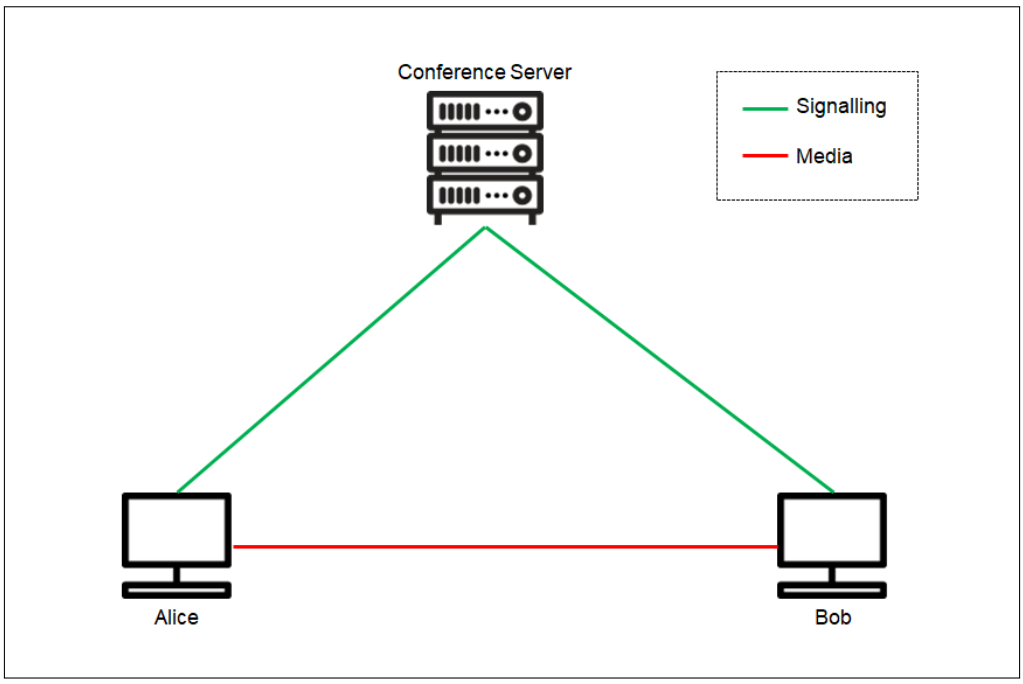

Abbildung 2: Direkte Verbindung zwischen zwei Endpunkten

Abbildung 3 zeigt eine Verbindung zwischen zwei Endpunkten, die sich mithilfe der zuvor ermittelten externen IP-Adress-Informationen erreichen können.

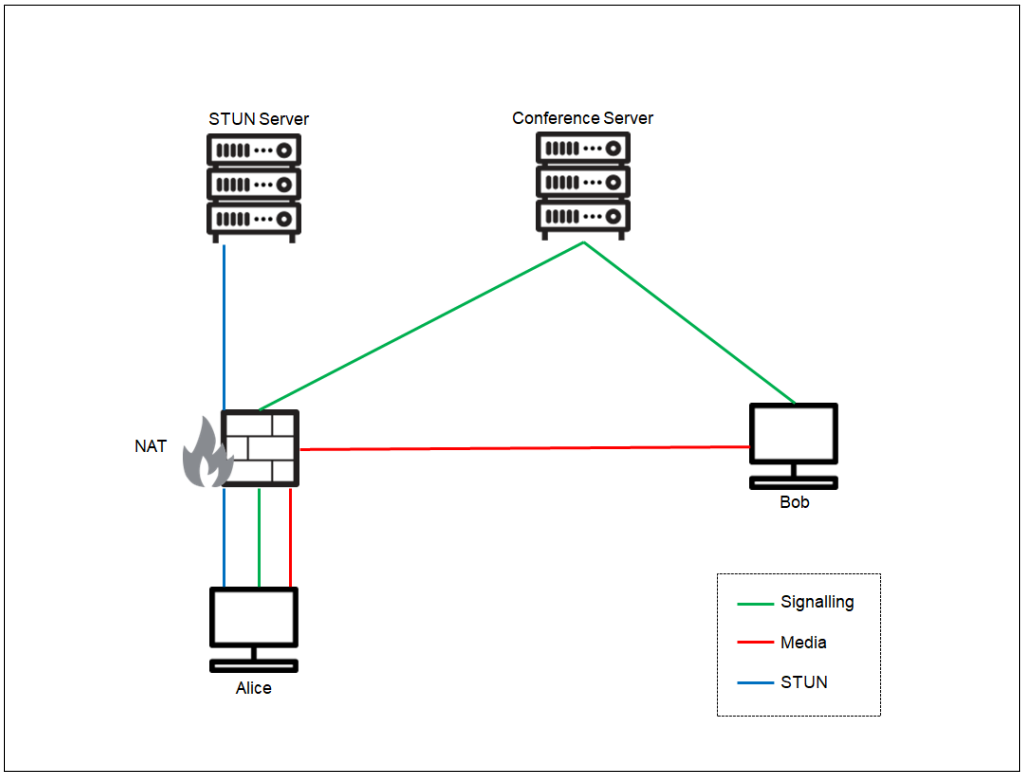

Abbildung 3: Direkte Verbindung zwischen zwei Endpunkten, bei der mithilfe von STUN das öffentliche IP/Port-Paar ermittelt wurde

Ein NAT vom Typ „Symmetric“ verwendet für jede neue ausgehende Verbindung ein separates IP/Port-Paar, auch wenn das Paket von demselben Client-Port stammt. Somit kann der Client das verwendete IP/Port-Paar für neue Verbindungen auch unter Zuhilfenahme von STUN nicht vorhersagen. Befindet sich einer der beteiligten Teilnehmer also hinter einem symmetrischen NAT, so verhindert dieses eine eingehende Verbindung eines anderen Endpunktes auf das mithilfe von STUN ermittelte IP/Port-Paar.

Um dieses Problem zu umgehen, wird ein TURN-Server als Mediarelay verwendet (s. Abbildung 4).

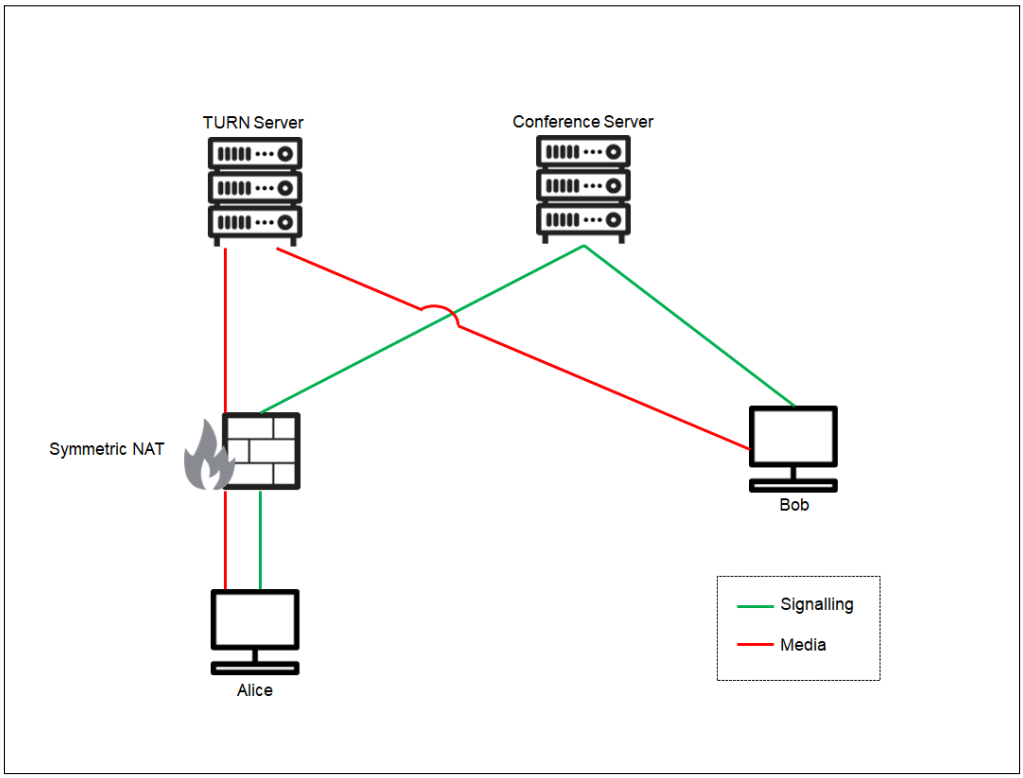

Abbildung 4: Verbindung zwischen zwei Endpunkten über einen TURN Mediarelay-Server

Alle Möglichkeiten werden in Form von Kandidaten im SDP aufgeführt und an den Server gesendet.

```
Lokaler IP/Port Kandidat:
a=candidate:1 1 UDP 2122252543 192.168.10.98 59697 typ host

Durch STUN ermittelter öffentlicher IP/Port Kandidat
a=candidate:2 1 UDP 1694498815 77.47.111.245 16724 typ srflx

Konfigurierter TURN Mediarelay-Server
a=candidate:3 1 UDP 8265727 3.127.240.102 50440 typ relay
```

Die jeweiligen Endpunkte ermitteln mit den empfangenen Informationen anschließend die bestmögliche Verbindung.

Verschlüsselung von Mediendaten

Um die Vertraulichkeit der Medienstreams zu gewährleisten, werden diese verschlüsselt. Um die Daten vor dem Einblick Dritter abzusichern, werden sie meist mit dem Secure Real-Time Transport Protocol (SRTP) [13] und dem AES [14]-

Verschlüsselungsverfahren geschützt. Für den Schlüsselaustausch kommen unter Verwendung der WebRTC-Technologie zwei Varianten zum Einsatz: das SRTP-SDES- und das SRTP-DTLS-Verfahren.

SRTP-SDES

Bei einem Security Descriptions for Media Streams (SDES)-Schlüsselaustausch wird der zu verwendende AES-Schlüssel als Medienattribut im SDP aufgeführt und an den Server gesendet.

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

Abbildung 5 zeigt die Topologie einer Session, in der der AES-Schlüssel für die SRTP-Daten mit der SDES-Methode ausgehandelt wurde.

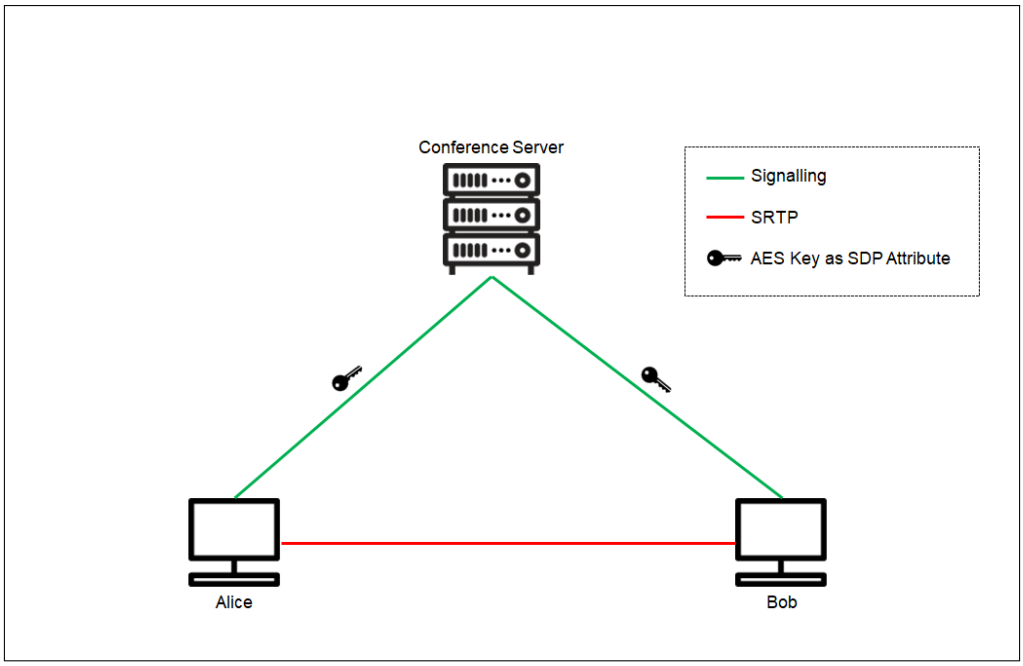

Abbildung 5: Topologie einer SRTP-SDES-Session

Da bei der SDES-Methode der AES-Schlüssel im SDP als Medienattribut übertragen wird, kennen alle Verbindungspunkte in einer Session den Schlüssel und der Medienstream ist damit nicht Ende-zu-Ende-verschlüsselt (E2EE) [15].

Dies führt zu zwei sicherheitsrelevanten Problematiken:

- Hat ein Betreiber des Konferenzservers Zugriff auf die SRTP-Daten, so kann er mithilfe des AES-Schlüssels die Daten entschlüsseln und abhören.
- Kommt ein Angreifer an den Inhalt des SDP – z. B. mithilfe eines TLS-Proxy –, so kann er den SRTP-Stream ebenfalls entschlüsseln und abhören.

Da bei dieser Methode keine Ende-zu-Ende-Verschlüsselung möglich ist, wird sie nur noch sehr selten verwendet. In einzelnen Fällen kann es sein, dass bestimmte Implementierungen SRTP-SDES anbieten, um eine Abwärtskompatibilität zu gewährleisten.

SRTP-DTLS

Im Gegensatz zur SDES-Methode wird bei einem Datagram Transport Layer Security (DTLS) [16]-Schlüsselaustausch der Schlüssel nicht im SDP über mehrere Punkte hinweg, sondern in einem gesicherten Kanal direkt zwischen den Endpunkten ausgetauscht.

DTLS ist dem asynchronen Verschlüsselungsverfahren Transport Layer Security (TLS) [17] nachempfunden und in allen modernen Browsern implementiert.

Um die Identität des gegenüberliegenden Endpunkts – z. B. des Browsers – zu überprüfen, wird der Fingerprint, der als SDP-Attribut übermittelt wird, als Verifizierungsgrundlage benutzt.

Abbildung 6 zeigt die Topologie einer Session, in der der AES-Schlüssel für die SRTP-Daten mit der DTLS-Methode ausgehandelt wurde.

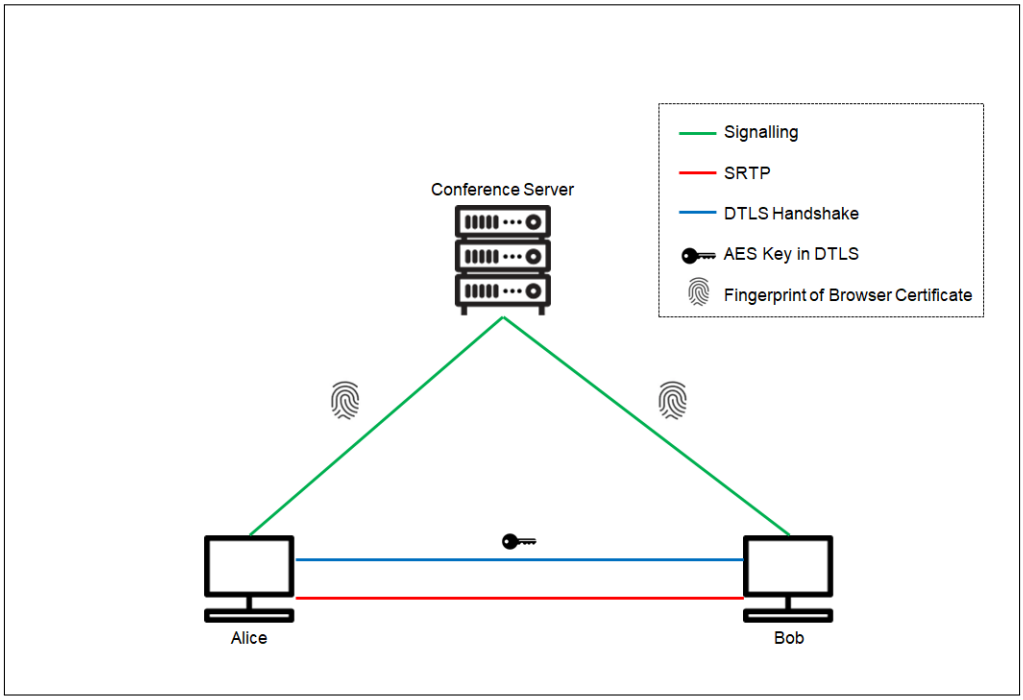

Abbildung 6: Topologie einer SRTP-DTLS-Session

Da bei dieser Methode der AES-Schlüssel nur den jeweiligen Medienendpunkten bekannt ist, kann hier von einer Ende-zu-Ende-Verschlüsselung gesprochen werden.

Der Einsatz von SRTP-DTLS bietet einen höheren Schutz im Vergleich zu SRTP-SDES, dennoch ist auch diese Methode nicht vor aktiven Man-in-the-Middle-Angriffen geschützt. So könnte ein Angreifer, der Zugriff auf den Transport-verschlüsselten Signalisierungsstream hat, den Fingerprint der Zertifikate aktiv manipulieren, um so mithilfe eines DTLS-Proxy den Medienstream anzugreifen.

Die WebRTC-Spezifikation [18] gibt vor, dass SRTP-DTLS als Schlüsselverwaltungsverfahren von WebRTC-Implementierungen erforderlich ist. Es ist allgemein anerkannt, dass DTLS-SRTP die obligatorische und standardmäßige Option für die Verschlüsselung von WebRTC-Medien sein sollte.

Peer-to-Peer

Eine Konferenz mit zwei Teilnehmern kann man auch als Peer-to-Peer (P2P)-Verbindung bezeichnen. Optimalerweise agiert der Konferenzserver bei einer P2P-Konferenz als Signalisierungseinheit und die Medienübertragung findet direkt zwischen den beiden Endpunkten statt.

Im Optimalfall kann man von einer tatsächlichen Ende-zu-Ende-Verschlüsselung für den Medienstream ausgehen, da dieser unter Zuhilfenahme von ICE direkt zwischen den beiden Endpunkten besteht und auch der Schlüsselaustausch durch SRTP-DTLS zwischen den tatsächlichen Konferenzteilnehmern stattfindet.

Peer-to-Multipeer

Sobald eine Konferenz die Grenze von zwei Teilnehmern überschreitet, spricht man von einer Peer-to-Multipeer (P2MP)-Verbindung. Damit alle Teilnehmer untereinander kommunizieren können, werden verschiedene Verfahren der Medienübertragung eingesetzt.

Mesh

Bei der Mesh-Architektur sendet und empfängt jeder Teilnehmer den Medienstream an alle anderen und von allen anderen Teilnehmern. Da beim Meshing die insgesamt benötigte Bandbreite bei jedem weiteren Teilnehmer quadratisch ansteigt, ist diese Architektur auf wenige Teilnehmer limitiert und daher nicht weit verbreitet.

Abbildung 7 zeigt den Aufbau der Mesh-Architektur.

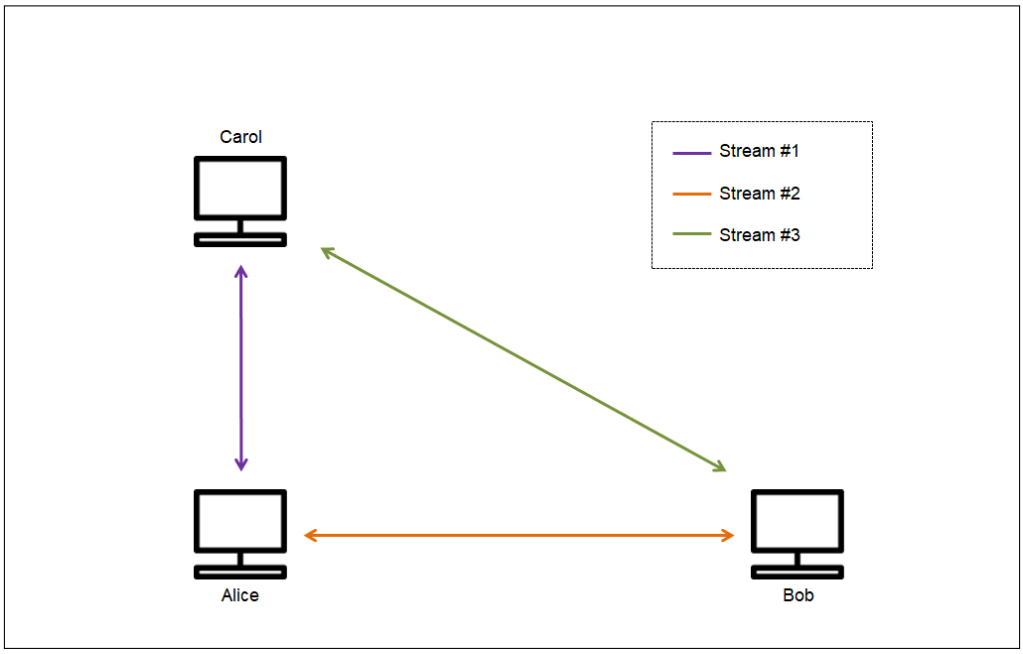

Abbildung 7: Mesh-Architektur

Der Vorteil beim Meshing ist die Möglichkeit einer Ende-zu-Ende-Verschlüsselung des Medienstreams zwischen den Teilnehmern.

Multipoint Control Unit

Eine Multipoint Control Unit (MCU) ist ein Dienst, der die Medienstreams aller Teilnehmer zu einem einzigen Stream mischt, um auf diese Weise eine Kommunikation zwischen allen Teilnehmern zu ermöglichen. Die MCU benötigt aufgrund von Decodierung, Mischung und Codierung der Streams eine hohe Rechenleistung, jedoch wird gegenüber Meshing die benötigte Bandbreite für die Teilnehmer drastisch reduziert. Die Maximalgröße einer Konferenz hängt bei diesem Verfahren vor allem von der eingesetzten MCU und deren Ressourcen ab.

Abbildung 7 zeigt die Medienstreams einer WebRTC-Konferenz mithilfe einer MCU, die im Konferenzserver integriert ist.

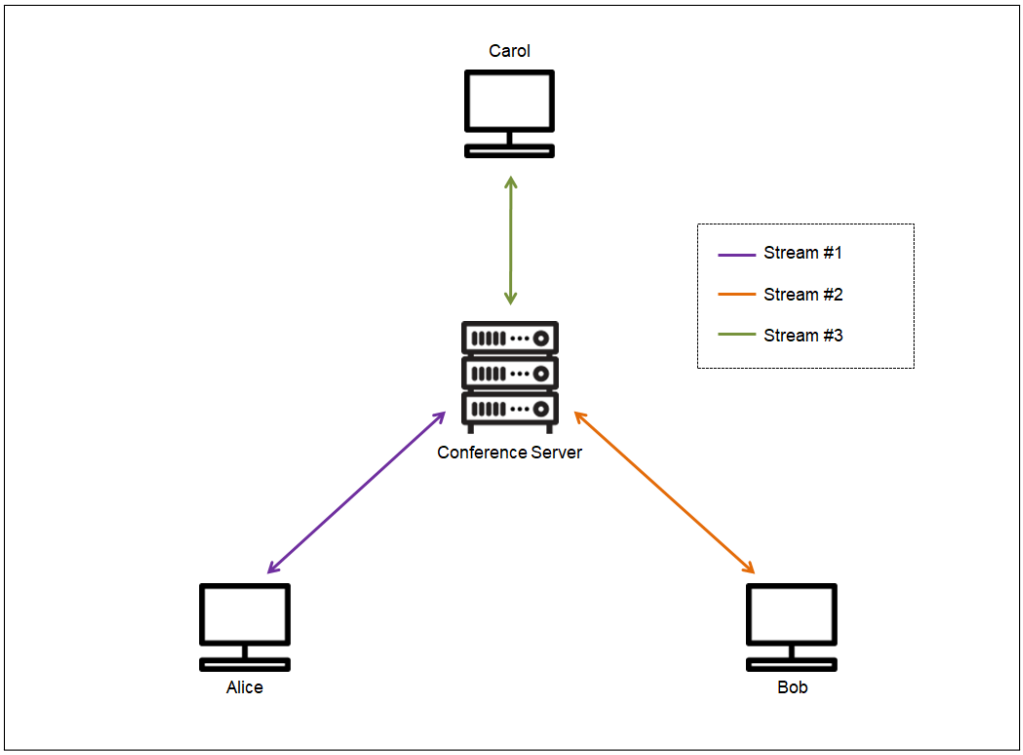

Abbildung 8: MCU-Architektur

Die MCU gilt als jeweiliger Medienendpunkt, d. h. auch der Schlüsselaustausch findet zwischen Teilnehmer und MCU statt. Eine Verschlüsselung ist daher nur zwischen Teilnehmer und MCU möglich und die Medienstreams sind somit nur Hop-by-Hop (HbH)- und nicht vollständig zwischen den Teilnehmern Ende-zu-Ende-verschlüsselt.

Selective Forwarding Unit

Um die Last auf dem Server zu verringern, besteht die Möglichkeit des Einsatzes einer Selective Forwarding Unit (SFU). Eine SFU ist in der Lage, mehrere Medienstreams zu empfangen und dann zu entscheiden, welcher dieser Streams an welche Teilnehmer gesendet werden soll. So ist es möglich, Mediendaten zu routen, ohne diese vorher decodieren, mischen und wieder codieren zu müssen.

Abbildung 9 zeigt die Medienstreams einer WebRTC-Konferenz, die mithilfe einer SFU geroutet werden.

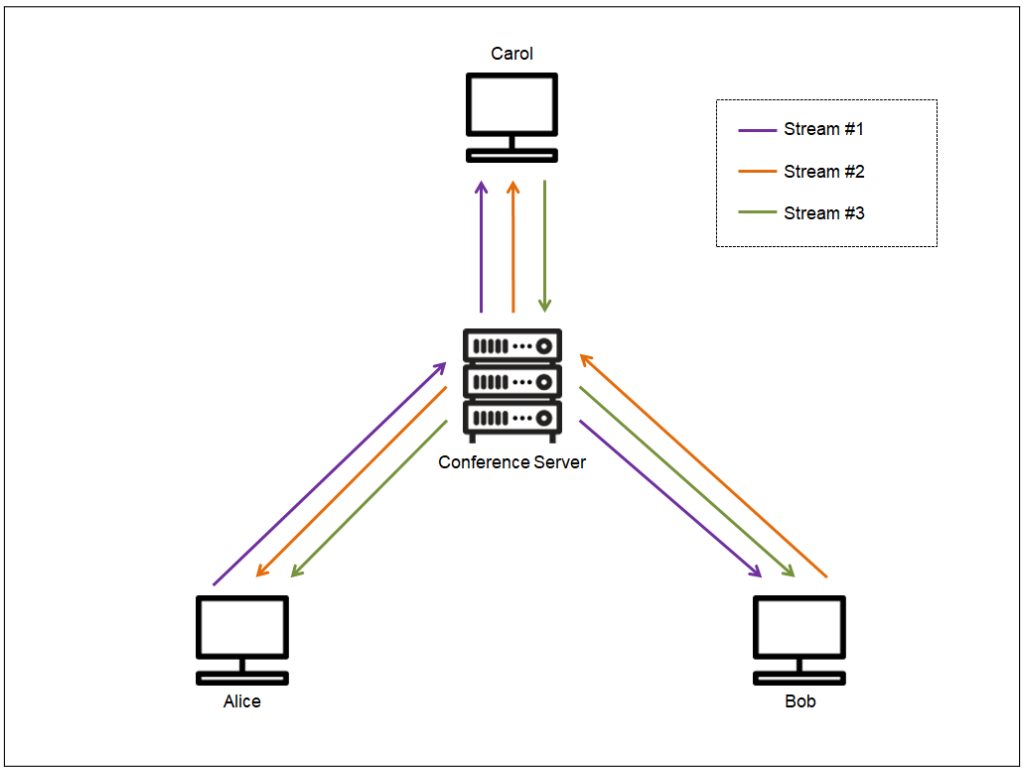

Abbildung 9: SFU-Architektur

Wie auch schon beim Einsatz der MCU ist in diesem Anwendungsfall die SFU der Medienendpunkt. Die Mediendaten sind zum derzeitigen Stand der Technik Hop-by-Hop- und nicht vollständig Ende-zu-Ende-verschlüsselt.

Sicherheitsprobleme bei Online-Konferenzen und WebRTC

Im Kontext von Online-Konferenzen und WebRTC sind einige Schwachstellen und Sicherheitsprobleme besonders häufig anzutreffen. Bei den in diesem Abschnitt beschriebenen Problematiken handelt es sich nicht um Implementierungsschwachstellen, sondern um Sicherheitsprobleme, die den Architekturen oder Spezifikationen geschuldet sind.

Belauschen von Medienstreams

Sobald eine Konferenz über eine MCU oder SFU geführt wird, ermöglicht dies dem Betreiber des Konferenzservers bzw. der MCU oder SFU das Belauschen der Audio- und Videodaten. Da die jeweilige MCU oder SFU als Medienendpunkt agiert, kennt sie auch alle AES-Schlüssel der Medienstreams und der darin enthaltenen SRTP-Pakete.

Neben dem Betreiber kann auch ein potentiell kompromittierter Server ein entsprechendes Risiko darstellen. Da Konferenzsysteme oftmals ins öffentliche Internet exponiert sind, vergrößert sich die Angriffsfläche, und die Wahrscheinlichkeit, dass eine Schwachstelle oder ein Konfigurationsfehler von einem Angreifer ausgenutzt wird, nimmt drastisch zu.

Preisgabe von IP-Adressen

Da mit der Verwendung von ICE alle Kandidaten gesammelt und an den Server gesendet werden, erhält der gegenüberliegende Endpunkt Informationen über öffentliche und private IP-Adressen.

Dies hat zur Folge, dass unter anderem die lokalen IP-Adressen eines Teilnehmers bekannt gemacht werden. Beispielsweise werden Informationen über das lokale WiFi-, Client- oder VPN-Netzwerk preisgegeben.

Ist der Medienendpunkt z. B. ein Teilnehmer einer P2P- oder gemeshten P2MP-Konferenz, so werden ihm die Informationen über die IP-Adressen übermittelt. Neben dem datenschutzrechtlichen Aspekt liefern die Informationen auch lukrative Erkenntnisse über ein potentielles Opfer und dessen korrespondierende Netzwerke.

Unzureichender Datenschutz in verschlüsselten Medienstreams

Das SRTP-Protokoll verschlüsselt die Nutzdaten eines Medienstreams, nicht aber den Header und die damit verbundenen Metadaten eines RTP-Pakets. Der Header enthält Informationen, deren Geheimhaltung wünschenswert sein kann.

Jeder, dem es möglich ist, SRTP-Pakete zu sehen, kann erkennen, ob ein Teilnehmer zu einem bestimmten Zeitpunkt spricht oder nicht. Es ist z. B. möglich, herauszufinden, wie viele Sprachanteile ein Teilnehmer einer Konferenz hat, um so Rückschlüsse auf potentiell besprochene Themen zu ziehen.

Ein Angreifer in einer passenden Man-in-the-Middle (MitM)-Position muss lediglich die SRTP-Pakete mitschneiden und anschließend analysieren. Dieses Problem besteht sowohl bei P2P- als auch bei P2MP-Konferenzen.

Vergleich der Konferenzsysteme

In diesem Abschnitt werden die Ergebnisse der Analyse von verschiedenen freien und kommerziellen Konferenzsystemen aufgeführt. Die Konferenzsysteme wurden unter den folgenden Aspekten untersucht:

- **Verschiedene Clients**: Um mögliche Unterschiede zwischen verschiedenen Clients und Browsern zu analysieren, wurden die Tests überwiegend mit einem 64-Bit Firefox-Browser in der Version 68.7.0 sowie einem 64-Bit Google Chrome-Browser in der Version 81.0.4044.129 aus dem offiziellen Build durchgeführt. Neben den Browsern wurden zusätzlich die jeweiligen offiziell verfügbaren mobilen Applikationen auf einem Android- und einem iOS-Gerät verwendet.
- **P2P Audio/Video-Verschlüsselung**: Es wurde unter Berücksichtigung der verschiedenen Client-Kombinationen analysiert, ob eine Peer-to-Peer Audio/Video (AV)-Verbindung Ende-zu-Ende- oder Hop-by-Hop-verschlüsselt ist.
- **P2MP Audio/Video-Verschlüsselung**: Neben der Analyse einer P2P AV-Verbindung wurde auch jeweils die Verschlüsselung bei einer Peer-to-Multipeer (P2MP)-Verbindung untersucht.
- **Privatchat- und Gruppenchat-Verschlüsselung**: Da viele Konferenzsysteme die Möglichkeit bieten, Chatnachrichten zu übermitteln, wurde jeweils geprüft, ob Gruppen- sowie Privatchats Ende-zu-Ende-verschlüsselt sind.
- **IP-Preisgabe**: Die Problematik der IP-Preisgabe in Richtung des jeweiligen Servers sowie der anderen Teilnehmer wurde untersucht.
- **STUN-Server**: Ein STUN-Server könnte Nutzungsdaten sammeln. Aus diesem Grund sind die eingesetzten STUN-Server der jeweiligen Systeme aufgeführt.
- **Encryption Symbol**: Erörtert wurde die Frage, ob ein Endbenutzer, der wenig technisches Know-how besitzt, erkennen kann, ob und vor allem wie die Konferenz und die dazugehörigen Mediendaten abgesichert sind.
- **DTLS Cipher / SRTP Cipher**: Die ausgehandelten DTLS- sowie SRTP-Chiffren wurden ermittelt.
- **Aktiver Medienstream ohne weitere Teilnehmer**: Wird ein AV-Stream bereits beim Betreten einer Konferenz ohne weitere Teilnehmer an den Server gesendet, ist es dem Server möglich, den Teilnehmer zu belauschen, während er z. B. noch auf weitere Teilnehmer wartet. Dieses potentielle Verhalten wurde in den Tests untersucht.

- **Aktiver Medienstream nach Verlassen von Teilnehmern**: Die Frage, ob ein Medienstream aktiv bleibt, nachdem alle anderen Teilnehmer (bis auf einen) die Konferenz verlassen haben, wurde in den Tests ebenfalls berücksichtigt.

Übersicht der Testergebnisse

Die Testergebnisse sind im Nachfolgenden tabellarisch dargestellt:

	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

Tabelle 1: Vergleich der Konferenzsysteme

Open Rainbow

Open Rainbow [19] ist eine kommerzielle Konferenzlösung des Unternehmens Alcatel Lucent Enterprise. Um Open Rainbow nutzen zu können, muss ein Account erstellt werden, mit dem dann die vom Betreiber verwendete Open Rainbow-Plattform zugänglich ist.

P2MP-Konferenzen und -Unterhaltungen finden in sogenannten „Bubbles" statt, zu denen man Teilnehmer einladen und hinzufügen kann.

Negative Testergebnisse:

– Open Rainbow kann nicht in einer eigenen Infrastruktur betrieben werden.

– Findet eine Konferenz aus einer Bubble heraus statt, so sind die Medienstreams Hop-by-Hop-verschlüsselt. Dies gilt sowohl für P2P- als auch P2MP-Bubble-Konferenzen.

– Beginnt eine Bubble-Konferenz mit mehr als zwei Teilnehmern und es verlassen anschließend alle bis auf zwei Teilnehmer die Konferenz, so ist der Medienstream der beiden verbliebenen Teilnehmer Hop-by-Hop-verschlüsselt.

– Chatnachrichten werden in einer WebSocket-Verbindung an den Server gesendet.

– Private IP-Adressen werden an den Server und in einer P2P-Verbindung an die Teilnehmer der Konferenz gesendet. Da Open Rainbow eine Anruffunktion enthält, ist es möglich, einen Kontakt anzurufen. Nimmt dieser ab, so werden alle privaten IP-Adressen übermittelt.

– Es existiert ein aktiver Medienstream zum Server in einer Bubble, ohne dass andere Teilnehmer eingewählt sind.

Jitsi

Jitsi [20] ist eine aus mehreren Open-Source-Projekten [21] zusammengestellte Konferenzlösung, die alle gängigen Funktionen einer Online-Konferenz enthält. Jitsi erfreut sich großer Beliebtheit und lässt sich vollständig in der eigenen Infrastruktur betreiben.

Es gibt auch verschiedene frei verfügbare und fremd gehostete Jitsi-Instanzen im Netz. Eine der beliebtesten frei verfügbaren Instanzen ist die vom Entwickler-Team selbst bereitgestellte Instanz – erreichbar unter der URL `https://meet.jit.si`.

Für die Untersuchung von Jitsi wurde sowohl die öffentliche Jitsi-Instanz des Entwicklers als auch eine selbst gehostete Instanz untersucht, die mithilfe der empfohlenen „Quick Installation" [22]-Methode aufgesetzt worden ist.

Negative Testergebnisse:

– Verwendet ein Teilnehmer einer P2P-Konferenz einen Firefox-Browser, so werden die Medienstreams Hop-by-Hop verschlüsselt.

– Eine P2MP-Konferenz wird immer Hop-by-Hop verschlüsselt.

– Chatnachrichten werden als HTTP POST-Request an den Server gesendet.

– Private IP-Adressen werden dem Server bekannt gemacht. Bei einer P2P-Konferenz werden private IP-Adressen den Teilnehmern bekannt gemacht.

– Verlassen alle anderen Teilnehmer die Konferenz, so bleibt der Medienstream zum Server für rund 20 Sekunden erhalten.

Zoom

Zoom ist eine kommerzielle Konferenzlösung, welche vor allem angesichts der COVID 19-Pandemie über Nacht einen regelrechten Nutzerboom erlebt hat. Die Plattform wird vom gleichnamigen Unternehmen Zoom Inc. [23] entwickelt und bereitgestellt. Es gibt neben der Browserintegration für alle gängigen Betriebssysteme native Clients, die erweiterte Funktionalitäten beinhalten, z. B. einen virtuellen Hintergrund.

Zoom verwendet ein eigenes proprietäres Protokoll. Dieses wurde genauer untersucht. Um die Analyse zu erleichtern, wurde ein Wireshark Lua Dissector erstellt. Abbildung 10 zeigt ein Zoom-Audiopaket.

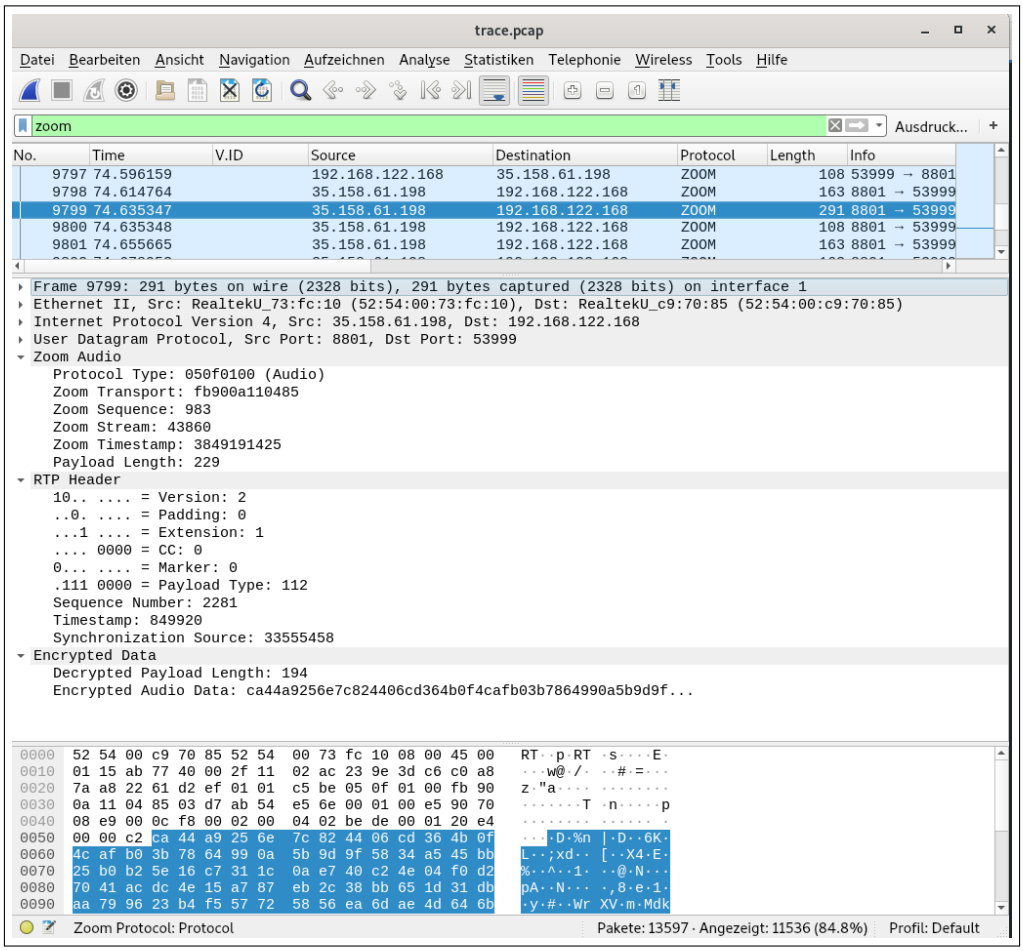

Abbildung 10: Zoom-Audiopaket mit Wireshark Dissector

Wie auch schon Bill Marczak und John Scott-Railton in ihrem Artikel vom 3. April 2020 [24] feststellen, können die verschlüsselten Audiodaten separiert werden, da diese als zusätzlicher Frame auf das Zoom-eigene Protokoll aufgesetzt sind. Die Untersuchungen der verwendeten Verschlüsselung für die Nutzdaten ergaben, dass AES im ECB-Modus mit einer Schlüssellänge von 256 Bit verwendet wird. Im Encryption Whitepaper [25] von Zoom wird dies bestätigt; eine Umstellung auf AES im GCM-Modus mit einer Schlüssellänge von 256 Bit ist mittlerweile erfolgt.

Neben dem Wireshark Dissector wurde auch ein passendes Skript erstellt, welches die Entschlüsselung der verschlüsselten RTP-Daten mithilfe des AES-Schlüssels erlaubt. Mit einem passenden SILK-Decoder [26], ist es dann möglich, die SILK-Daten in ein MP3-Format zu konvertieren.

Negatives Testergebnis

– Bei der Verwendung von Zoom im Browser findet keine Ende-zu-Ende-Verschlüsselung statt.

– Chatnachrichten innerhalb einer Konferenz werden Base64-codiert an den Server gesendet.

– Nutzdaten werden im ECB-Modus verschlüsselt.

– Es existiert ein Medienstream zum Server, wenn die Konferenz nur (noch) aus einem Teilnehmer besteht.

Microsoft Teams

Microsoft Teams, der direkte Nachfolger von Skype for Business, ist wohl die bekannteste Online-Konferenz-Lösung für Unternehmen. Neben Conferencing bietet Teams noch viele weitere Collaberation-Funktionalitäten sowie eine breite Palette an nachrüstbaren Plug-ins. Wie aus dem Produktnamen hervorgeht, wird Teams von Microsoft [28] entwickelt und bereitgestellt.

Besonders anzumerken ist die Verwendung der SRTP-SDES-Methode, wodurch der AES Key für die Verschlüsselung der Mediendaten im Klartext im SDP-Objekt enthalten ist. Im Fall von Microsoft Teams wird das SDP-Objekt in einem HTTP POST-Request an den Microsoft-Webserver gesendet. Dadurch ist der AES Key nicht nur den jeweiligen Medienendpunkten, sondern noch zusätzlichen Servern bekannt, die von der Mediensession unabhängig sind.

Negatives Testergebnis:

– Es findet keine Ende-zu-Ende-Verschlüsselung statt.

– Chatnachrichten werden als HTTP POST-Request im Klartext an den Server gesendet.

– Die SRTP-SDES-Methode wird verwendet.

– Der AES Key wird als HTTP POST-Request an den Webserver gesendet. Der AES Key ist damit mehreren Servern bekannt.

– Es existiert ein aktiver Medienstream zum Server, wenn die Konferenz nur (noch) aus einem Teilnehmer besteht.

– Private IP-Adressen werden dem Server bekannt gemacht.

Empfehlung

Da es nach aktuellem Stand unter Verwendung einer SFU oder MCU nicht praktikabel ist, die Mediendaten auch in einer P2MP-Konferenz Ende-zu-Ende zu verschlüsseln, wird empfohlen, vertrauliche Daten ausschließlich über Systeme unter eigener Hoheit auszutauschen.

Kontrollieren Sie, an welche externen Systeme schützenswerte Daten übermittelt werden.

Stellen Sie sicher, dass Ihre P2P-Konferenz auch tatsächlich Ende-zu-Ende verschlüsselt ist, indem Sie die RTCPeerConnection-Objekte im Browser untersuchen.

Die erste Anlaufstelle für die Kontrolle einer WebRTC-Konferenz sind die internen WebRTC-Tools der Browser:

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

In den bereitgestellten Browser-Tools werden detaillierte Informationen zu den jeweiligen RTCPeerConnection-Objekten aufgeführt. Abbildung 11 zeigt eine WebRTC-Session, die mithilfe der Google Chrome WebRTC-Tools detailliert dargestellt wird.

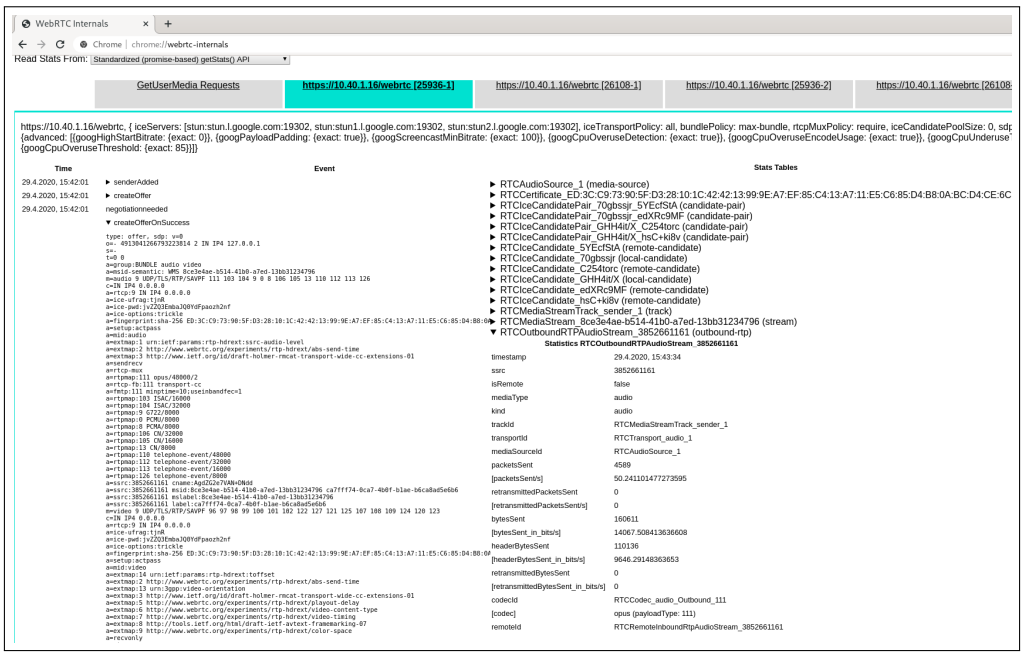

Abbildung 11: Anzeige einer WebRTC-Session mithilfe der Google Chrome WebRTC-Tools

Neben den WebRTC-Tools können auch Netzwerksniffer wie Wireshark verwendet werden, um zu analysieren, an welche Systeme und Server Daten gesendet werden.

Fazit

Die beschriebene Analyse zeigt, dass keines der untersuchten Konferenzsysteme eine vollständige Ende-zu-Ende-Verschlüsselung bietet. Teilweise ist der Schlüsselaustausch abhängig vom verwendeten Client. So macht es durchaus einen Unterschied, ob ein Benutzer mit der nativen mobilen Applikation, einem Firefox- oder dem Chrome-Browser einer Konferenz beitritt.

Aufgrund fehlender Ende-zu-Ende-Verschlüsselung besteht nach derzeitigem Stand die Gefahr, dass Dritte die Medienstreams belauschen können. Auch Nachrichten, die während einer Konferenz über die oftmals integrierte Chatfunktion ausgetauscht werden, sind in den untersuchten Systemen nicht Ende-zu-Ende-verschlüsselt und damit nicht ausreichend vor Einblicken Dritter geschützt.

Neben der Vertraulichkeit der Daten ist auch der Umgang mit schützenswerten Daten – wie z. B. eine lokale VPN-IP-Adresse – nicht zufriedenstellend. So können Angreifer teilweise durch einfaches Anrufen eines Kontakts bereits Informationen über die lokale IP-Adress-Konfiguration sammeln. Auch die Betreiber der Serverinstanzen können Daten über die Benutzer sammeln. So könnte man Benutzer teilweise profilieren, indem die öffentlichen und lokalen IP-Adressen zu bestimmten Zeiträumen analysiert werden.

Außerdem zeigt die Untersuchung, dass es für Endbenutzer schwierig ist, zu erkennen, wie die Mediendaten einer Konferenz geschützt sind, und ob diese Ende-zu-Ende-verschlüsselt sind.

Ausblick

Das Entwicklerteam von Google Chrome arbeitet aktuell an der Implementierung von sogenannten „Insertable Streams“ [29]. Über diese API ist es WebRTC-Anwendungen möglich, auf bereits codierte Mediendaten zuzugreifen, bevor sie ins

Netzwerk gelangen. Dies erlaubt, mittels SRTP Double Encryption Procedures [30] die Nutzdaten der Medienstreams erneut zu verschlüsseln, um so die Vertraulichkeit auch über Selective Forwarding Units hinweg zu gewährleisten.

Die Entwickler von Jitsi haben dafür bereits eine Proof-of-Concept-Funktion [31] eingebaut, die es ermöglicht, die Daten Ende-zu-Ende zu verschlüsseln. Der Schlüssel für die zweite Verschlüsselungsebene wird in der Demo-Funktion derzeit noch über einen dritten Kommunikationskanal verteilt, beispielsweise über eine verschlüsselte E-Mail. An einer geeigneten Schlüsselverwaltung wird nach eigenen Aussagen der Entwickler gearbeitet.

Quellen

[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 7

[17] E. Rescorla, RFC 8446, tools.ietf.org, https://tools.ietf.org/html/rfc8446, 2018-08 7

[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/ 15

[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/` 16

[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 18

[31] Emil Ivov, jitsi.org, `https://jitsi.org/blog/e2ee/`, 2020-04-12 18

THE PENTEST EXPERTS

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

WWW.SYSS.DE