VAF | BUNDESVERBAND TELEKOMMUNIKATION

SIP- und RTP-Troubleshooting Teil 2

Teil 2: Werkzeuge – Open Source, Third-Party Appliances, Betriebssystem- und Herstellertools

Bild: unsplash.com / @bermixstudio

www.vaf.de

Impressum

SIP- und RTP-Troubleshooting
Teil 2: Werkzeuge – Open Source, Third-Party Appliances, Betriebssystem- und Herstellertools

Übersicht

Autor
Benjamin Pfister, Rotenburg an der Fulda

Redaktion
Martin Bürstenbinder (V. i. S. d. P.), Dr. Christian Jerger (Schlussredaktion und Lektorat), Uwe Klenner (Layout, Titelseite und Bildbearbeitung). Beratende Unterstützung: VAF-Fachkreis Technik mit den Mentoren Bernd Rücker und Jens Wilkens.

Bilder
Benjamin Pfister; Titelseite: unsplash.com/@bermixstudio; Bildbearbeitungen: VAF

Herausgeber
VAF Bundesverband Telekommunikation e. V.
Schulstraße 2
40721 Hilden
www.vaf.de

Daniel Brosend (1. Vorsitzender), Martin Bürstenbinder (Geschäftsführer)

1. Auflage, August 2024

Inhaltsverzeichnis

Editorial

Der hier vorliegende Teil 2 setzt die dreiteilige Fachpapierreihe des VAF fort. Er behandelt die in der Praxis relevanten Werkzeuge des SIP/RTP-Troubleshootings. Zahlreiche Hinweise für den Praktiker runden den Text ab. Der kommende und abschließende Teil 3 wird neben den Methoden des Troubleshootings einen umfangreichen Part mit Praxisfällen beisteuern. Im Teil 1 wurden bereits die technischen Grundlagen behandelt.

Der Autor ist aktiver Troubleshooting-Experte im ITK-Bereich und wirkt für den VAF als technischer Berater und Trainer. Die Idee zu der Heftreihe entstand im VAF-Fachkreis Technik, und Vertreter aus Mitgliedsunternehmen haben an der Entwicklung von Konzept und Inhalt beratend mitgewirkt.

So vermitteln die Publikationen Know-how aus der praktischen Erfahrung heraus und grundlegendes Wissen zu der Fehleranalyse im besonderen Anwendungsbereich der Kommunikationstechnik. Sie unterstützen ITK-Integratoren und -Administratoren beim SIP/RTP-Troubleshooting und tragen so dazu bei, dessen Effektivität und Effizienz zu steigern.

Die Heftreihe darf jedoch nicht als Ersatz für das allgemeine Grundlagenwissen zur Netzwerktechnik, zu den einschlägigen Protokollen, zu SIP-Trunks usw. verstanden werden. Sie setzt vielmehr darauf auf. Für interessierte Einsteiger sei darum auf die Fachschulungen des VAF hingewiesen, zu denen auch diese Publikationsreihe als begleitendes Lehrmaterial dienen kann.

Das ITK-Systemhaus als Problemlöser für den Kunden positioniert sich erfolgsorientiert am Markt. Letztlich zielt die Heftreihe darauf, den Mitgliedern in dem relevanten Kompetenzbereich des VoIP-spezifischen Troubleshootings eine kompakte Wissenssammlung für die Fortbildung an die Hand zu geben.

Hilden, Juni 2024

Martin Bürstenbinder
Geschäftsführer
VAF Bundesverband Telekommunikation e. V.

1. Einleitung

In Teil 1 der Fachpapierreihe zum SIP- und RTP-Troubleshooting haben wir zunächst die technischen Grundlagen mit dem Protokollaufbau erläutert sowie Dialoge und Servertypen dargestellt. Ohne Protokollkenntnisse hilft jedoch das beste Tool nichts, da die theoretischen Grundlagen fehlen.

Ziel des Teils 2 ist es, Ihnen einen Werkzeugkoffer für das Troubleshooting an die Hand zu geben, den Sie nach Bedarf für die Aufzeichnung von Paketmitschnitten und Systemverhalten, deren Analyse sowie die Behebung von Fehlern verwenden können.

Wir beginnen mit einigen Open-Source-Tools, wie dem beliebten grafischen Protokollanalysewerkzeug Wireshark und seinem Kommandozeilen-Pendant TShark. Zudem stellen wir SIP- und RTP-Testtools vor, mit denen Sie applikationsspezifisches Verhalten prüfen können.

In einigen Fällen reichen die Open-Source-Tools jedoch nicht aus oder es ist eine höhere Analyseeffizienz mit Third-Party Tools und Hardware Appliances erzielbar. Daher stellen wir Ihnen einige Beispiele dieser Kategorien vor.

In anderen Fällen möchte man jedoch ohne zusätzliche Installationen etwas aufzeichnen oder analysieren. Daher stellen wir sowohl für unixoide als auch Windows-Betriebssysteme einige Werkzeuge vor, die »out of the box« nutzbar sind.

Die Hersteller von ITK-Systemen bieten zudem meist auch selbst einige Funktionen an, die folglich speziell auf die Eigenheiten und Anforderungen des jeweiligen Systems zugeschnitten sind. Über diese Varianten geben wir einen Überblick.

2. Open-Source-Tools

2.1 Wireshark

Eines der beliebtesten Aufzeichnungs- und Analysetools für das SIP- und RTP-Troubleshooting ist das Open-Source-Tool Wireshark. Dieses verfügt über eine grafische Oberfläche. Das Tool steht unter einer GNU General Public License (GNU GPL) 2.0 und ist für eine Vielzahl von Betriebssystemen verfügbar. Dazu gehören die aktuellen Windows-Client- und Serversysteme, wie Windows 10/11 und Windows Server 2016/2022, aber auch verschiedene Linux- und BSD-Varianten sowie macOS.

2.1.1 Aufzeichnung mit Wireshark

Zur Aufzeichnung von Live-Daten nutzt Wireshark die libpcap-Bibliothek. Somit sind auch die Art und Anwendung von Aufzeichnungsfiltern von den Möglichkeiten von libpcap abhängig. Unter Windows wird aktuell nur noch Npcap weiterentwickelt.

Um die Menge an aufgezeichneten Daten einzugrenzen, kann der Analyst sogenannte Aufzeichnungsfilter (Capture-Filter) anwenden, beispielsweise auf Basis von Quell- und Ziel-IP-Adressen oder entsprechender Ports. Dies sollte jedoch mit Vorsicht geschehen. Daten im Nachgang zu filtern ist meist einfacher, als beim Endkunden neue Aufzeichnungen zu erstellen, weil die Daten zu restriktiv gefiltert wurden.

Für das SIP- und RTP-Troubleshooting wäre ein Capture-Filter auf den SIP-Proxy, den Signalisierungsport oder die RTP-Portbereiche sinnvoll. Im Beispiel des unten stehenden Screenshots **(Bild 1)** filtern wir in den eingehenden Aufzeichnungsoptionen auf die IP-

Adresse 10.10.10.10 über den Filter »ip host« sowie den Port 5060 oder den RTP-Portbereich 16384 bis 32767. Damit werden sowohl UDP (User Datagram Protocol) als auch TCP (Transmission Control Protocol) zur Aufzeichnung herangezogen. Die logische »Und«-Verknüpfung definieren wir mit dem Wort »and«, die logische »Oder«-Verknüpfung entsprechend mit zwei Pipe-Symbolen: ||.

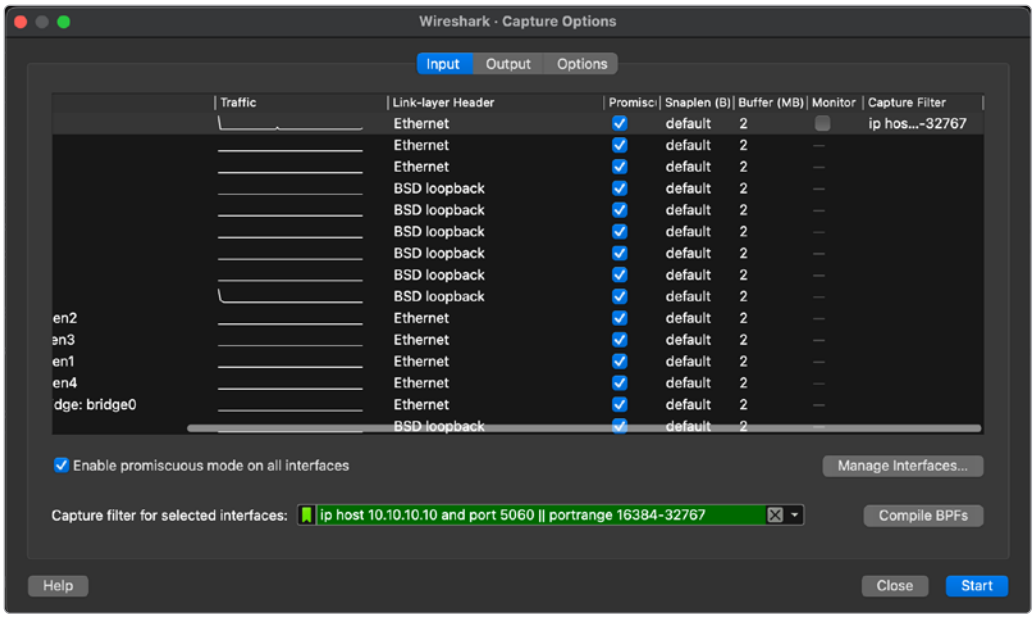

Bild 1: Hinterlegung eines Capture-Filters für einen SIP-Proxy auf Port 5060 und den RTP-Portbereich 16384 und 32767

Neben den Filtern bietet Wireshark jedoch noch viel mehr, um die Aufzeichnung zu optimieren. So lassen sich Traces, die größer als 100 MB sind, nur schwer filtern. In vielen Fällen empfiehlt sich auch, den Capture-Buffer zu vergrößern. Dieser dient als Puffer des Capture-Treibers, bevor Daten auf die Festplatte/SSD geschrieben werden können, und ist im Standard 2 MB groß (siehe **Bild 1**). Sollten beim Mitschnitt Pakete verloren gehen, bietet sich entsprechend die genannte Vergrößerung an. Geht es um große Capture-Dateien, verhält sich Wireshark sehr träge bei der Analyse und Anzeigefilterung. Daher empfiehlt es sich, bei längeren Aufzeichnungen oder falls große Datenmengen anfallen, ein Größenlimit zu setzen. Dieses können wir in den Output-Optionen hinterlegen. Um bei einem begrenzten lokalen Speicher einen zeitweiligen Fehler zu ermitteln, bietet sich ein Ringspeicher an. In dem in **Bild 2** dargestellten Screenshot haben wir den Ringspeicher auf zehnmal 100 MB große Dateien eingestellt. Nachdem der Fehler aufgetaucht ist, kann der Analyst die Aufzeichnung stoppen und die Dateien in dem Zeitslot analysieren, in dem der Fehler auftrat.

2.1.2 Auswertung mit Wireshark

Nach der Aufzeichnung geht es an die Analyse. Um aus den Rohdaten hierarchische Darstellungen für die strukturierte Auswertung zu generieren, wendet Wireshark sogenannte Dissektoren an. Diese definieren, wie das jeweilige Protokoll erkannt wird. Im Nachgang entsteht ein Protokollbaum, den Wireshark wie im unten stehenden Screen-

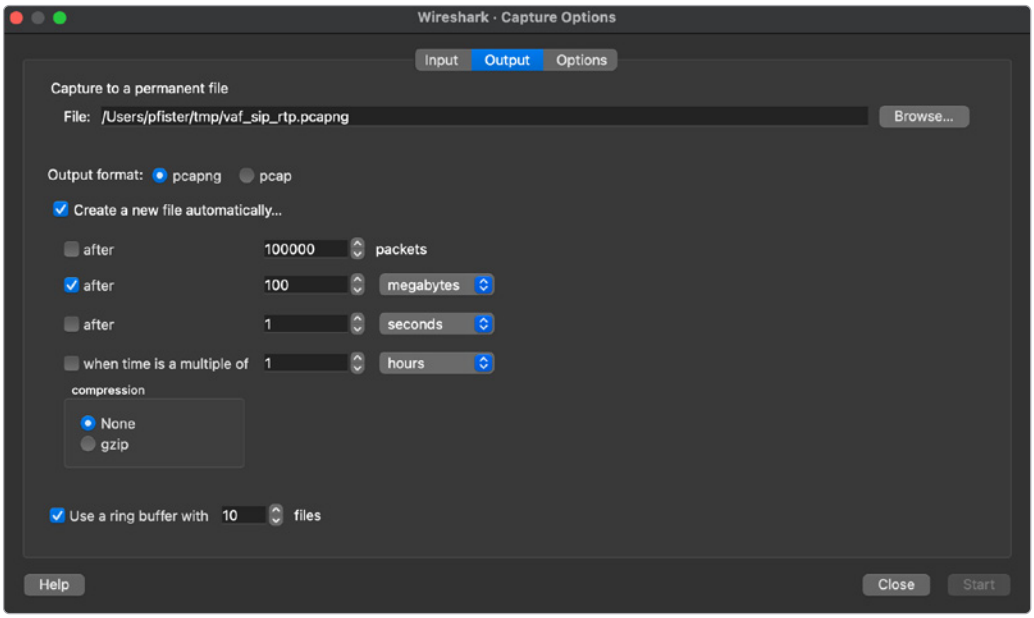

Bild 2: Hinterlegung eines Größenfilters der pcapng-Datei auf 100 MB und eines Ringspeichers auf zehn Dateien

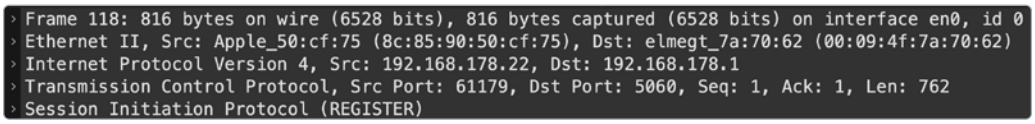

Bild 3: Darstellung der Anwendung von Ethernet-II-, IPv4-, TCP- und SIP-Dissektoren

shot **(Bild 3)** darstellt. Im linken Bereich erkennen wir Pfeile zum Aufklappen. Nach einem Klick darauf erscheinen wiederum Attribut/Wert-Paare innerhalb des Protokolls, die sich aufklappen lassen. So entsteht der oben genannte Baum der Protokollhierarchie, der wiederum eine strukturierte Analyse im Troubleshooting zulässt.

Um die zuvor aufgezeichneten Daten gezielter auszuwerten und folglich näher an den Root Cause eines Fehlers zu gelangen, gibt es sogenannte Anzeige- bzw. Display-Filter. Mit ihnen kann der Analyst die Anzeige der aufgezeichneten Daten entsprechend den Präferenzen und dem Fehlerbild eingrenzen. Die Display-Filter-Zeile befindet sich prominent im oberen Bereich der Wireshark-GUI unterhalb der Toolbar.

Die integrierten Dissektoren erkennen die Protokolle an deren Aufbau, sofern die Übertragung nicht verschlüsselt stattfindet.

Im folgenden Screenshot **(Bild 4)** haben wir beispielsweise einen Display-Filter auf das SIP- und RTP-Protokoll gesetzt. Dabei erkennen wir die beiden entsprechenden Protokollnamen »sip« und »rtp«. Sie werden über einen logischen Operator – in diesem Fall »or« – verknüpft. Die farbliche Hinterlegung zeigt an, dass eine Syntax-, jedoch keine Inhaltsprüfung des anzuwendenden Filters erfolgt. Ist sie rot, akzeptiert Wireshark den Filter nicht. Grün bedeutet, dass Wireshark ihn akzeptiert und er korrekt funktionieren sollte. Eine gelbe Hinterlegung bedeutet hingegen, dass der Filter akzeptiert wurde, aber vermutlich nicht die gewünschten Werte zutage fördert.

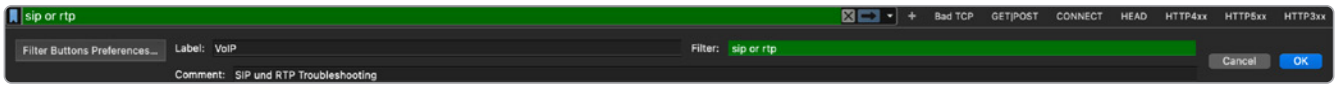

Bild 4: Anwendung eines Display-Filters auf SIP und RTP

Um wiederkehrend anzuwendende Display-Filter nicht mehrfach eintippen zu müssen, gibt es die Möglichkeit, einen Filter-Button zu hinterlegen. Im in **Bild 4** dargestellten Beispielfall würden wir über einen Klick auf das +-Symbol neben der Display-Filter-Zeile beispielsweise einen Filter-Button mit dem Namen »VoIP« anlegen und dahinter den Display-Filter »sip or rtp« verbergen. Somit muss der Systemverantwortliche oder Analyst nur noch den Button »VoIP« drücken, um den oben genannten Display-Filter anzuwenden. In **Tabelle 1** sind häufig verwendete Display-Filter für die SIP- und RTP-Analyse in Wireshark enthalten. Die zugehörigen Vergleichsoperatoren stellt **Tabelle 2** dar.

Display-Filter	Protokoll
sip	Session Initiation Protocol
rtp	Real-Time Transport Protocol
ip.addr	Quell- oder Ziel-IPv4-Adresse
ipv6.addr	Quell- oder Ziel-IPv6-Adresse
udp.port	User Datagram Protocol Port
tcp.port	Transmission Control Protocol Port
mac	Media Access Control Address

Tabelle 1: Häufig genutzte Display-Filter für Protokolle in Wireshark

Vergleichsoperatoren in Display-Filtern (unterschiedliche Schreibweisen mit Schrägstrich getrennt)	Funktion
eq / ==	Gleich
neq / !=	Nicht gleich
gt / >	Größer als
lt / <	Kleiner als
contains	Beinhaltet
matches / ~	Gleich regulärer Perl-kompatibler Ausdruck

Tabelle 2: Vergleichsoperatoren in Display-Filtern in Wireshark

Kombinationsausdrücke in Display-Filtern (unterschiedliche Schreibweisen mit Schrägstrich getrennt)	Funktion
and /]]	Logische »Und«-Verbindung
o / ||	Logische »Oder«-Verbindung
xor | ^^	Logische Verknüpfung mit exklusivem »oder«
not / !	Logisches Nicht
[…]	Anteilsoperator (Beschreibung in eckigen Klammern)
in	Beinhaltet Operator mit mehreren Bestandteilen (Beschreibung dahinter in geschwungenen Klammern)

Tabelle 3: Vergleichsoperatoren in Display-Filtern in Wireshark

In **Tabelle 4** haben wir einige Beispiele von Display-Filtern mit entsprechender Beschreibung aufgelistet.

Display-Filter	Funktion
sip or rtp	Stellt alle SIP- oder RTP-Pakete dar
sip.Method == »INVITE«	Stellt alle SIP-INVITE-Nachrichten dar
sip.Status-Code > 499	Zeigt alle Antworten an, deren Antwortcode ab 500 beginnt
sip.from.user == »+4915114948991«	Alle SIP-Pakete, die im User-Anteil der SIP-URI die Rufnummer +4915114948991 enthalten
rtp.ssrc == 0x3965e77f	Filtert auf die SSRC 0x3965e77f, die den RTP-Medienstrom identifiziert
rtp.p_type == 8	Filtert auf den Payload Type 8 für G.711a-Law
rtpevent	Filtert auf RFC2833&4733-DTMF-Töne

Tabelle 4: Beispiele für Display-Filter im Zusammenhang mit der SIP- und RTP-Analyse in Wireshark

Wireshark bietet auch die Möglichkeit einer einfachen Erkennung, Filterung und Auswertung einzelner Anrufe. Hierzu können wir uns im Menü »Telephony/VoIP Calls« zunächst die Anrufe auflisten lassen. Bei einer Signalisierungsverschlüsselung müssten wir zunächst den TLS-Datenstrom entschlüsseln.

Im Screenshot in **Bild 5** erkennen wir Quell- und Zielrufnummer sowie das Signalisierungsprotokoll. Hier kann der Analyst also neben SIP- auch H.323- oder MGCP-Anrufe erkennen.

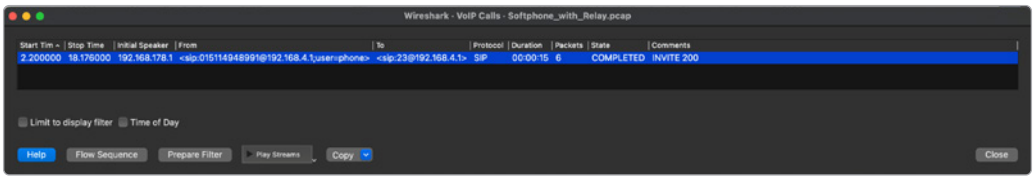

Bild 5: Auflistung erkannter VoIP-Anrufe