TLS mit Wireshark entschlüsseln

TLS mit Wireshark entschlüsseln

Was es beim kriminellen Man in the Middle zu verhindern gilt, gehört bei legal agierenden Systemadmins zum notwendigen Handwerkszeug: der Zugriff auf verschlüsselte Datenströme zwecks Fehlersuche.
Was es beim kriminellen Man in the Middle zu verhindern gilt, gehört bei legal agierenden Systemadmins zum notwendigen Handwerkszeug: der Zugriff auf verschlüsselte Datenströme zwecks Fehlersuche.

Von Benjamin Pfister

Der Anteil des verschlüsselten Datenverkehrs nimmt ständig zu. Fast alle Webdienste nutzen Transport Layer Security (TLS) und aktuelle Browser warnen bei unverschlüsselten HTTP-Verbindungen ausdrücklich vor dem damit verbundenen Risiko. Das ist aus Sicht der Sicherheit und des Datenschutzes sehr zu

begrüßen – doch die Verschlüsselung verhindert auch eine legale Analyse des Datenstroms, etwa seitens berechtigter Admins. Es gibt jedoch Möglichkeiten der Fehlersuche trotz TLS-Verschlüsselung, zum Beispiel mit dem im Folgenden beschriebenen Paketanalysewerkzeug Wireshark.

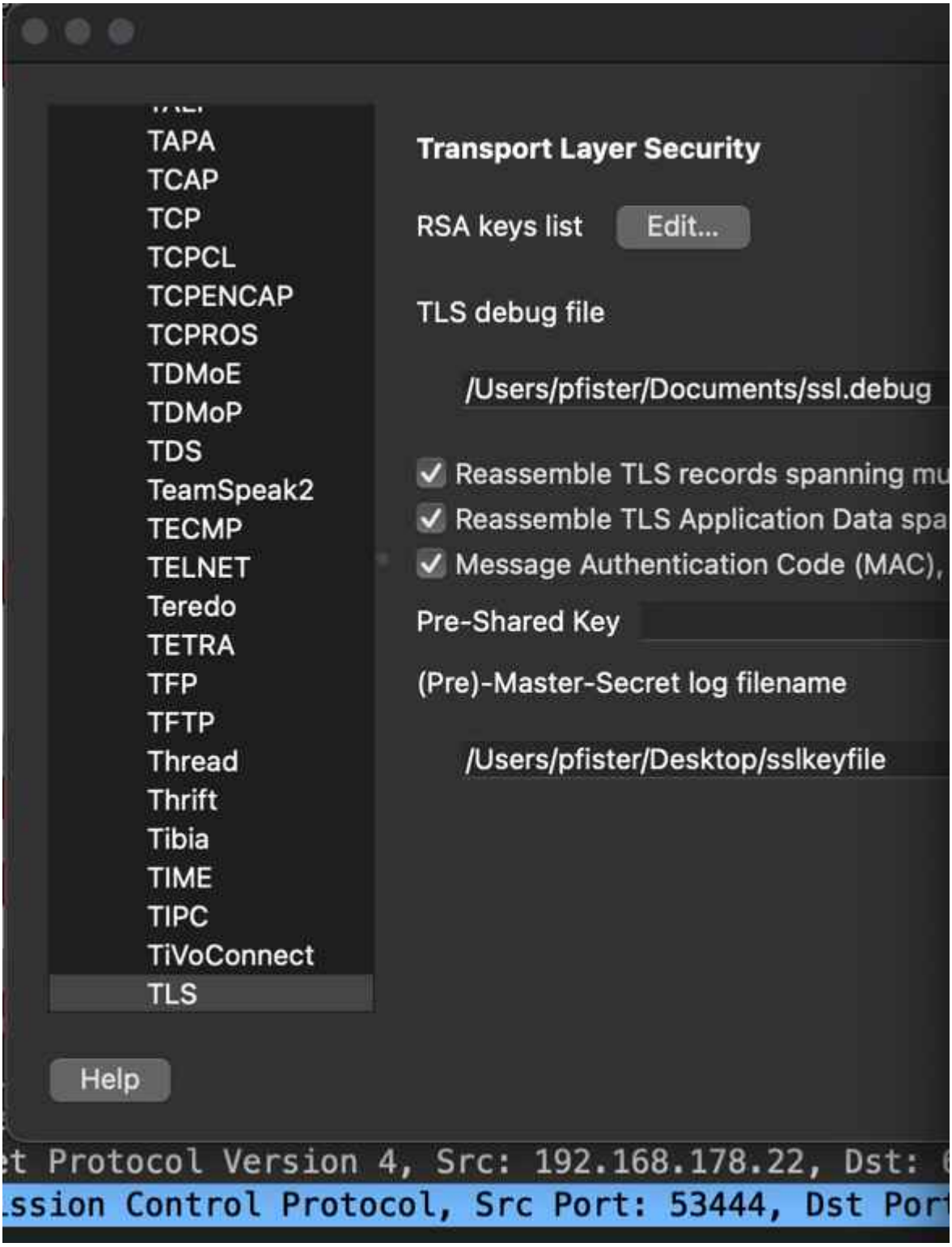
Wireshark bringt einen eigenen Dissector (wörtlich übersetzt Sezierer) für TLS mit. Er ermöglicht neben der Aufteilung und Darstellung der Protokolle auch die Entschlüsselung der Nutzdaten. Dazu bedarf es der passenden Schlüssel. Je nach eingesetzter Cipher Suite kommen unterschiedliche Entschlüsselungsmethoden zum Einsatz: auf Basis eines Session- (Pre-Master Secret) oder eines privaten RSA-Schlüssels.

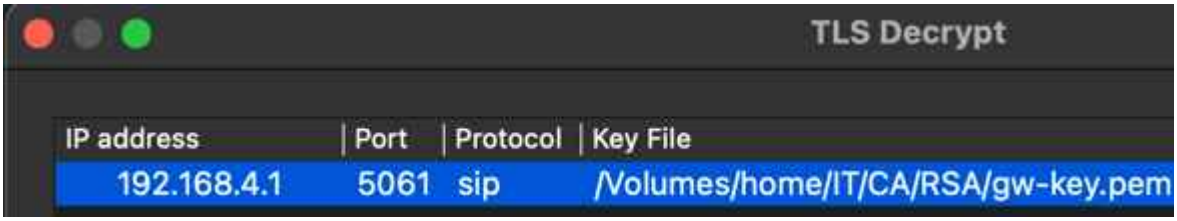
Welche der beiden zur Anwendung kommt, hängt von der Cipher Suite ab: mit Perfect Forward Secrecy (PFS) oder ohne. Falls für die Übertragung keine PFS Cipher Suites vorgesehen sind, kann die Entschlüsselung auf Basis des privaten Schlüssels des Serverauthentifizierungszertifkats stattfinden. In diesem Fall kann Wireshark jedoch auch die Methode Pre-Master Secret nutzen. Dies ist beispielsweise dann interessant, wenn man – wie bei öffentlichen Webdiensten – nicht im Besitz der privaten Schlüssel ist.

Bei Nutzung von Perfect Forward Secrecy (PFS) lässt sich der Datenstrom selbst bei Kenntnis des privaten Schlüssels nicht nachträglich entschlüsseln. Daher empfiehlt das BSI zum Schutz personenbezogener oder anderer sensibler Daten diese Cipher Suites. Darunter fallen die Cipher Suites mit Diffie-Hellman Ephemeral (DHE) und Elliptic Curve Diffie-Hellman Ephemeral (ECDHE). Um diese Varianten zu entschlüsseln, muss man die Methode Pre-Master Secret einsetzen.

Der Besitz des privaten Schlüssels nützt also nur dann etwas, wenn keine (EC)DHE Cipher Suites zum Einsatz kommen. Zudem funktioniert diese erste Variante nicht mit TLS 1.3. Einen weiteren Fallstrick birgt der TLS Session Resume, bei dessen Anwendung das Entschlüsseln fehlschlägt. Es bedarf der

Aufzeichnung eines ClientKeyExchange im TLS Handshake. Zum Entschlüsseln der Daten benötigt man das Serverauthentifizierungszertifikat – genauer dessen privaten Schlüssel. In den TLS-Protokolleinstellungen von Wireshark und dem Menüpunkt „RSA keys list“ referenziert man die Datei mit dem privaten Schlüssel und verknüpft ihn mit der IP-Adresse, dem Port und dem Protokoll des Servers. Abbildung 1 zeigt eine solche Hinterlegung für die IP-Adresse 192.168.4.1 mit dem Port 5061 und dem Protokoll SIP. Der Private Key liegt im Beispiel unter /Volumes/Home/IT/CA/RSA/gw-key.pem. Daran erkennt man, dass nicht nur HTTPS als Applikationsprotokoll zur Verfügung steht. Die Referenzen liegen im Beispiel von macOS unter /Users/<username>/.config/wireshark/ssl_keys.





Hinterlegung des Private Key aus dem Serverauthentifizierungszertifikat (Abb. 1).
Nach der korrekten Hinterlegung beginnt der Dissector mit der Entschlüsselung. Bei eventuellen Fehlern lohnt ein Blick in

die TLS-Debug-Datei, die beispielsweise fehlerhafte Private-Key-Zuweisungen oder Probleme beim Laden der Private Keys aufzeigt. Deren Zielverzeichnis und Namen kann man selbst wählen (siehe Abbildung 1).

Auf der Kommandozeile kann man das in Wireshark enthaltene CLI-Tool tshark nutzen. Für die RSA-Methode lautet der Befehl

```
tshark -o "ssl.keys_list: 192.168.4.1,5061,sip,/Volumes/Home/IT/CA/RSA/gw-key.pem" -r siptls.pcapng -Y sip
```

Über die Option

```
-o "ssl.keys_list: 192.168.4.1,5061,sip,/Volumes/Home/IT/CA/RSA/gw-key.pem"
```

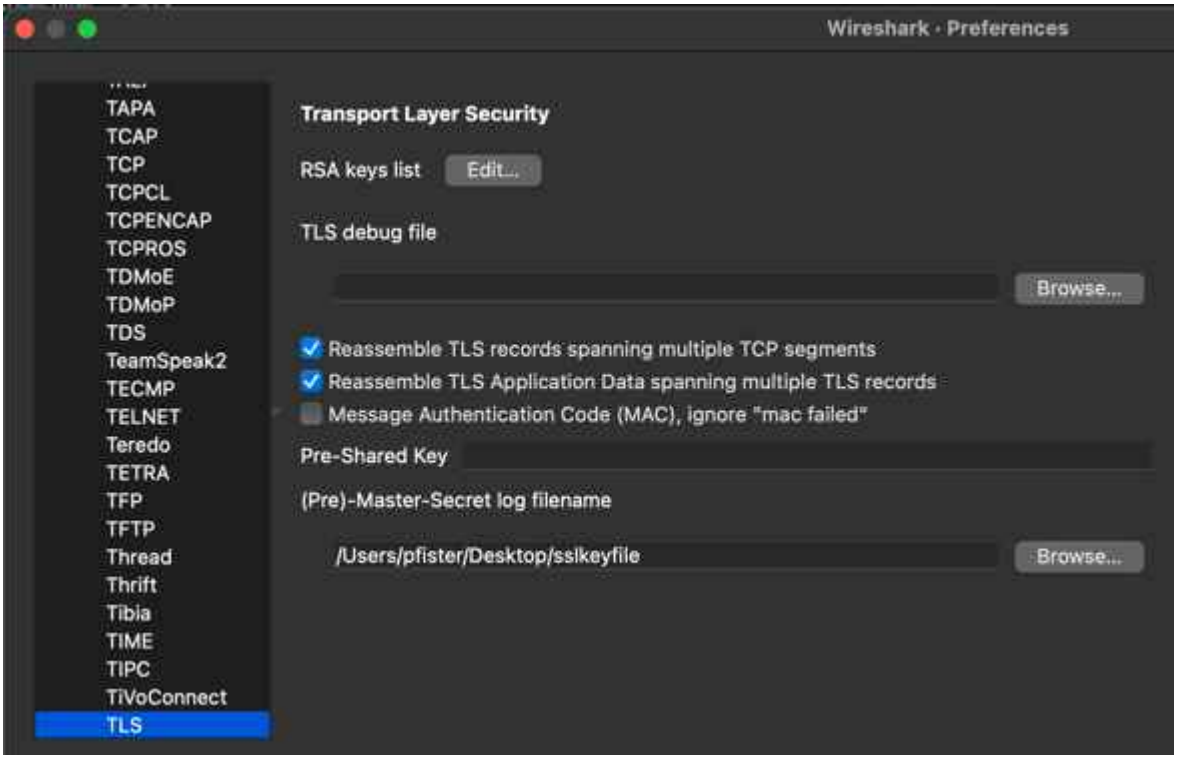
verknüpft man die in Abbildung 1 dargestellten Einstellungen – ähnlich wie mit der GUI-Variante. Das Argument -r siptls.pcapng liest dabei lediglich die PCAPNG-Datei. Das Argument -Y sip setzt einen Display-Filter auf das VoIP-Signalisierungsprotokoll SIP, sodass keine Pakete anderer Protokolle die Ausgabe fluten.

Die zweite Variante – keine Kenntnis des privaten Schlüssels und der Einsatz von (EC)DHE – setzt eine Keylog-Datei voraus, also eine Textdatei, die von unterschiedlichen Kryptobibliotheken bereitgestellt wird, beispielsweise OpenSSL oder NSS. Darauf aufbauende Applikationen wie Chrome, Firefox oder Curl generieren diese Datei, wenn die Umgebungsvariable SSLKEYLOGFILE gesetzt ist. Unter macOS kann man diese beispielsweise wie folgt anlegen: export SSLKEYLOGFILE="/Users/<username>/Desktop/sslkeyfile".

Die Bibliotheken schreiben den Pre-Master Key dann in die in der Umgebungsvariablen referenzierte Datei. Der Client generiert diesen in der Client Exchange Phase des TLS Handshake. Der Export kann auf Client- oder Serverseite stattfinden. Ein Mitlesen auf dem Transportweg ist somit nicht möglich. Wireshark kann den Pre-Master Key aus dem Handshake

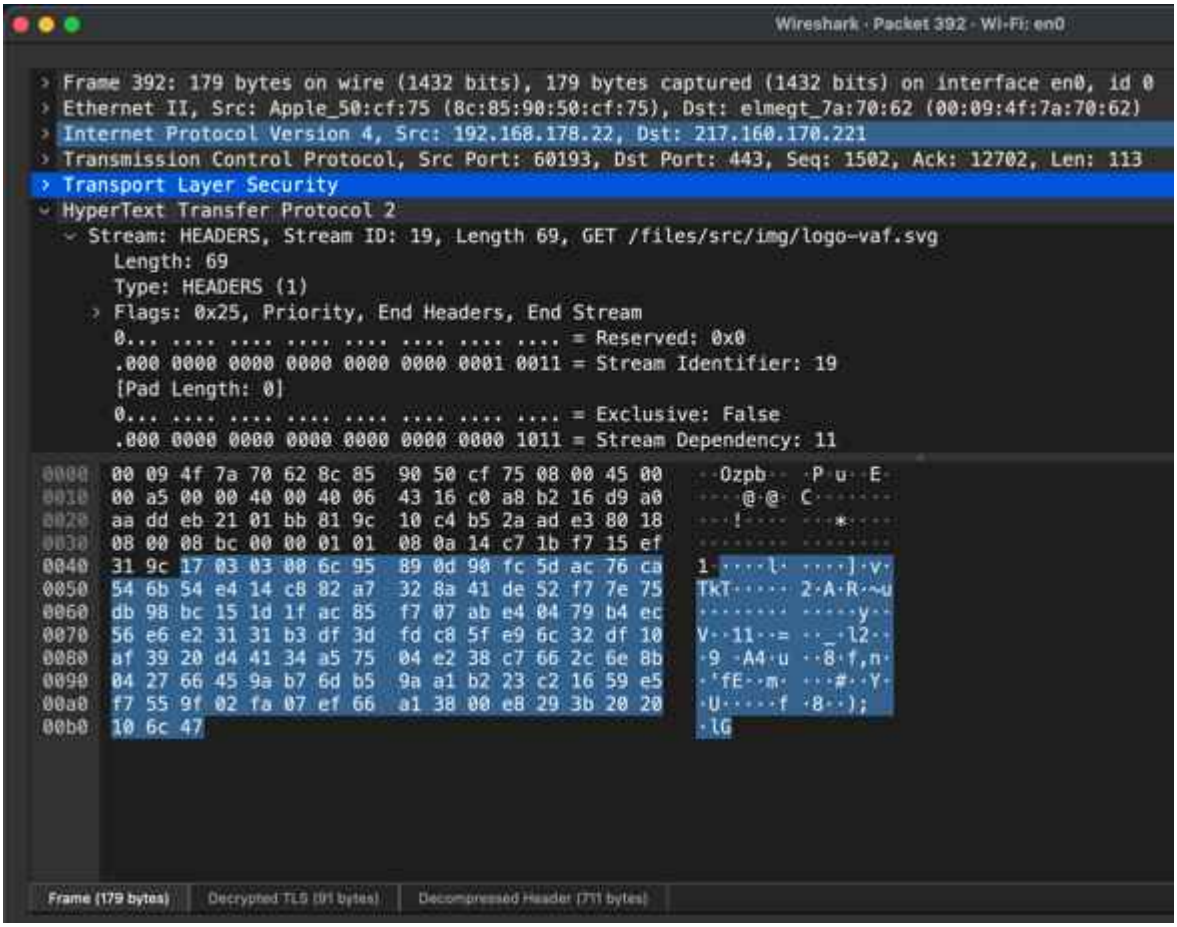
dafür nutzen, den Master Key abzuleiten und damit den Datenverkehr zu entschlüsseln. Im Anschluss an die Konfiguration der Umgebungsvariablen startet man den Mitschnitt in Wireshark und öffnet dann über die Konsole beispielsweise Firefox mittels open /Applications/Firefox.app unter macOS. Nachdem die erste TLS-verschlüsselte Seite aufgerufen wurde, zeigt sich, ob die Schlüsseldatei korrekt gespeichert wurde. In der ersten Zeile der Datei erkennt man auch, dass sie Bibliothek NSS für den Schreibvorgang zuständig war (siehe Abbildung 2).

Nach dem Laden der ersten TLS-verschlüsselten Daten zeigt sich am SSLKEYLOG, ob die Schlüsseldatei korrekt gespeichert wurde (Abb. 2).
Damit Wireshark die Datei mit den passenden Schlüsseln zum Entschlüsseln heranzieht, ist sie als „(Pre)-Master-Secret log filename“ unter „Preferences/Protocols/TLS“ zu referenzieren (siehe Abbildung 3).



Referenzierung der PMK-Datei in den TLS-Protokolleinstellungen in Wireshark – in diesem Fall /Users/pfister/Desktop/sslkeyfile (Abb. 3).

Sobald der TLS Dissector in Wireshark den Traffic entschlüsselt hat, wird der HTTP2-GET-Request im Klartext lesbar (siehe Abbildung 4). Dass eine Entschlüsselung stattgefunden hat, zeigen die Angabe „HyperText Transport Protocol 2“ unterhalb der Zeile „Transport Layer Security“ und der Hinweis „Decrypted TLS“ im unteren Bereich.



Entschlüsselter HTTP2-GET-Request (Abb. 4).

Wer die Kommandozeile bevorzugt, kann mit tshark arbeiten – es folgt ein Beispiel einer Aufzeichnung und Entschlüsselung per tshark. Zunächst wird wieder die Umgebungsvariable angelegt, gefolgt vom Öffnen des Browsers Mozilla Firefox. Anschließend startet tshark für 60 Sekunden (-a duration:60) ohne direkte Ausgabe (-Q) und schreibt die aufgezeichneten Daten in eine PCAPNG-Datei (-w /Users/pfister/Desktop/tls_decrypt.pcapng). In der letzten Zeile liest tshark die PCAPNG-Datei (-r) mit dem Argument für die Referenz zur Keylog-Datei ein (-o tls.keylog_file:$SSLKEYLOGFILE) und filtert die Ausgabe über einen Displayfilter auf HTTP (-Y http):

```
export SSLKEYLOGFILE="/Users/<username>/Desktop/sslkeyfile"
open /Applications/Firefox.app
tshark -Q -a duration:60 -w /Users/pfister/Desktop/tls_decrypt.pcapng &
tshark -r /Users/pfister/Desktop/tls_decrypt.pcapng -o tls.keylog_file:$SSLKEYLOGFILE -Y http
```

Fazit

Mit der Session-Key-Methode lassen sich selbst aktuelle Protokolle wie TLS 1.3 entschlüsseln. Dafür bedarf es jedoch einer Applikation, die den Sessionschlüssel in eine Logdatei schreibt. Falls dies nicht der Fall ist und Server und Client keine (EC)DHE Cipher Suite nutzen, kann der Analyst als Fallback die RSA-Methode anwenden. Grundsätzlich kann die Möglichkeit einer Entschlüsselung ein Troubleshooting jedenfalls immens erleichtern. Wireshark bietet dafür einen recht einfach zu nutzenden Ansatz. (un@ix.de)

1. Quellen
2. Weiterführende Informationen finden sich unter ix.de/ztmc.