Internet Security 2009W

Protokoll

WLAN Relay

Manuel Mausz, Matr. Nr. 0728348
manuel-tu@mausz.at
Aldin Rizvanovic, Matr. Nr. 0756024
e0756024@student.tuwien.ac.at

Wien, am 23. Dezember 2009

Inhaltsverzeichnis

Aufgabenstellung und Vorbereitung

Teil der Aufgabe ist es, ein „man in the middle“-Szenario aufzubauen, IP-Pakete diverser Protokolle aufzuzeichnen und diese zu analysieren. Fokus soll ins besonders auf Logindaten liegen.

Um einen „man in the middle“-Angriff zu ermöglichen, ist ein zusätzlicher Angreifer-PC notwendig, der als als Bridge (oder Router) agiert, der Pakete vom und zum Benutzer-PC empfängt und diese transparent an sein entsprechendes Ziel weiterleitet.

In der Aufgabenstellung ist der Angreifer-PC als Bridge zu konfigurieren. Dies funktionierte jedoch auf unserem Angreifer-PC, dessen Betriebssystem GNU/Linux ist, nicht. Die Verbindung zum Benutzer-PC via WLAN im AdHoc-Modus funktionierte zwar, doch sobald die 2 Endpunkt-Interfaces der Bridge hinzugefügt wurde, war kein Ping mehr möglich. Merkwürdigerweise funktionierte die IP-Zuweisung via DHCP, dessen Pakete ebenfalls durch die Bridge geroutet werden. Nach genauerer Recherche stellte sich heraus, das alle der Bridge zugewiesenen Interfaces, das dynamische Ändern der MAC-Adresse unterstützen müssen, was unsere Intel WLAN-Karte nicht unterstützte.

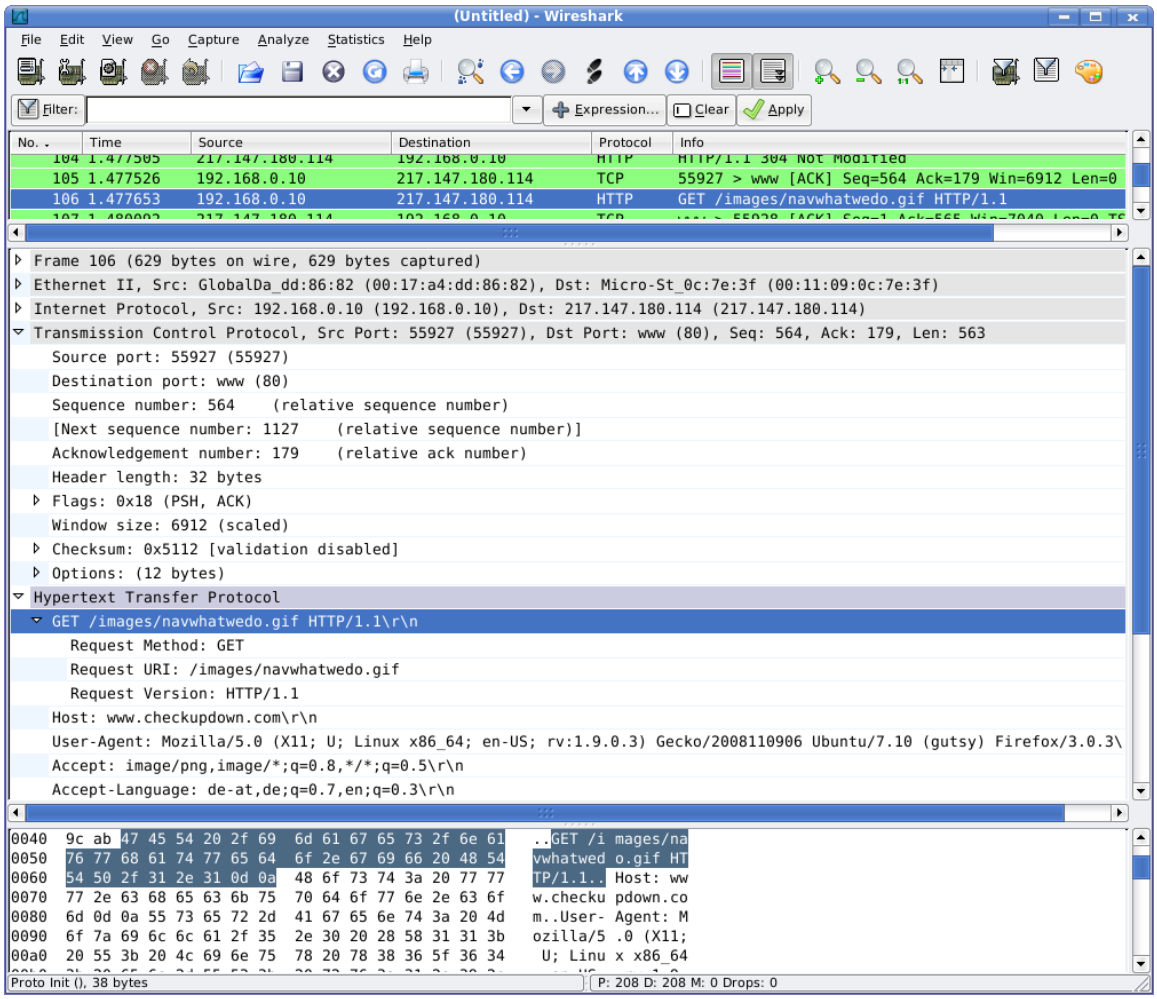
Daher haben wir uns für eine alternative Konfiguration entschieden, bei der der Angreifer-PC als Router dient und ein NAT (*„Network Adress Translation“)* aufbaut. Somit ist das Szenario, abgesehen von unterschiedlichen Subnetzen der 2 PCs, gegeben. Wie man ein NAT mit *iptables* unter Linux konfiguriert, ist auch Teil der Aufgabe „firewall“ und wird daher nicht näher behandelt.

Paketaufzeichnung und Tools

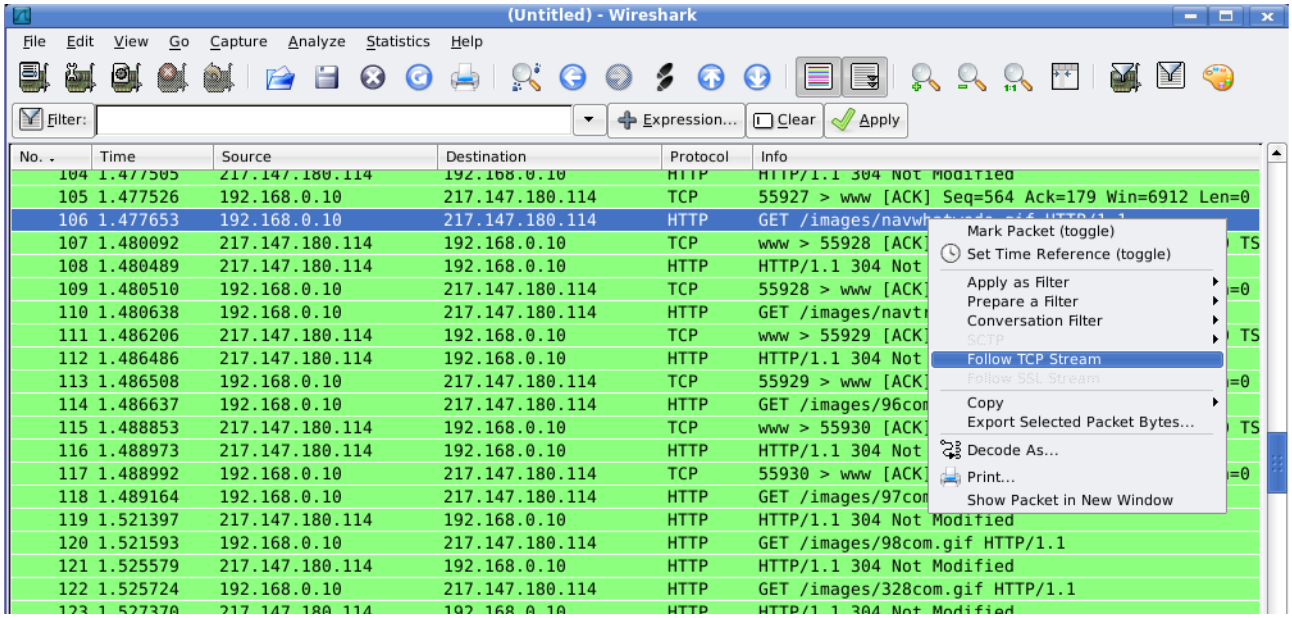
Zur Aufzeichnung der Datenpakete sowie zur Auswertung des TCP/UDP-Streams wurde das Opensource Tool Wireshark verwendet, das unter der Lizenz GPL entwickelt wird. Hierzu wird Wireshark gestartet, in den Options das entsprechend Ethernet Interface ausgewählt und die Aufzeichnung gestartet.

Alle ein und ausgehenden Pakete dieses Interfaces bekommen eine fortlaufende Nummer zugewiesen und werden direkt angezeigt in einer Tabelle angezeigt. Zudem wird üblicherweise die IP-Adresse der Quelle und des Ziels sowie der Port und einige Informationen aufgelistet. Genauere Details erhält man über das klicken auf das jeweilige Datenpaket. Zudem kann man sich, sofern für das jeweilige Protokoll unterstützt, alle Nutzdaten der einzelnen Pakete in einem eigenem Fenster anzeigen lassen.

Informationen zu dem ausgewählten Paket:



„*Follow TCP Stream*“ zum Anzeigen der Nutzdaten des Streams:



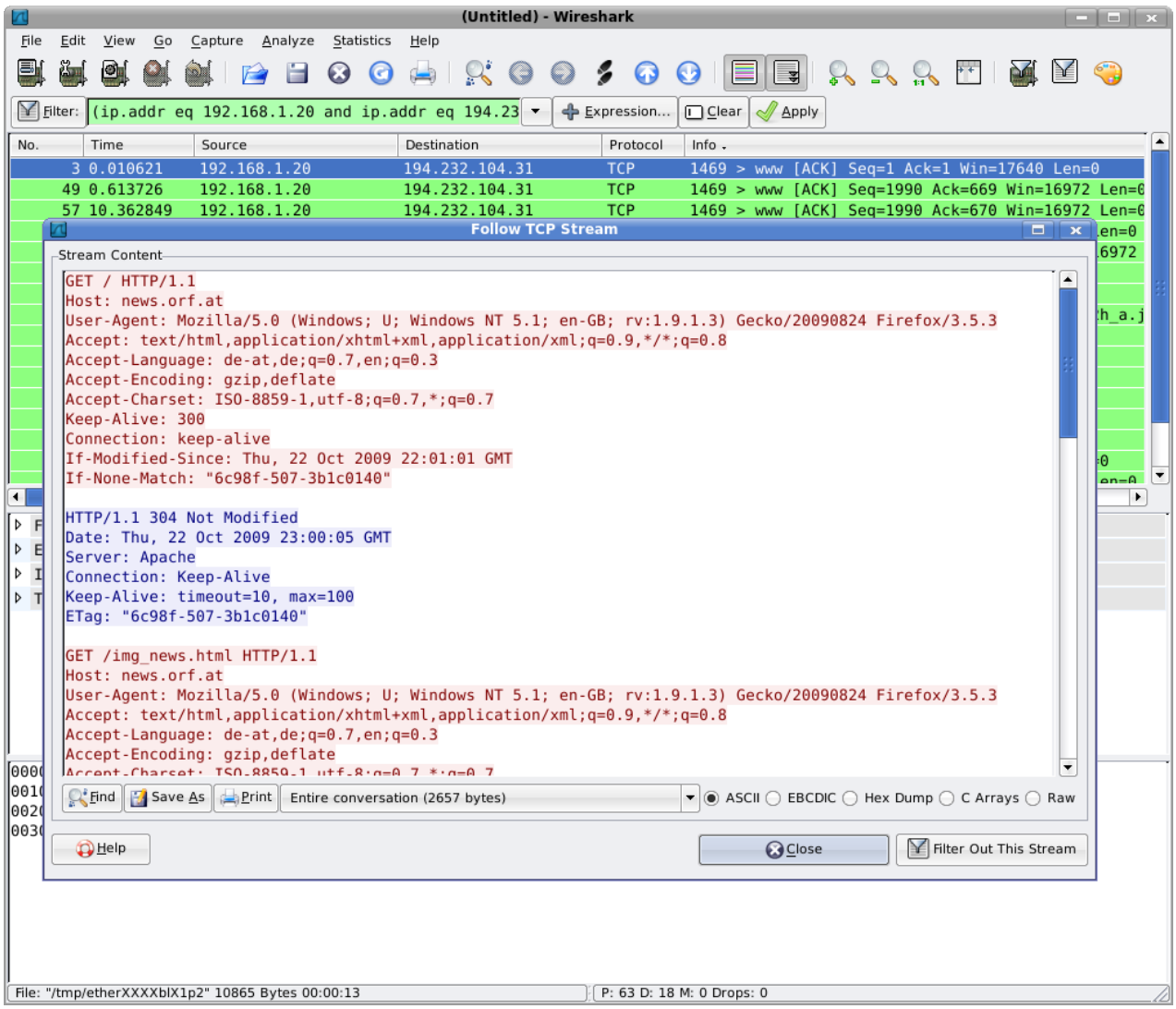
Protokoll: HTTP

Als erstes Protokoll wurde von uns HTTP ausgewählt. Hierbei wurde natürlich, wie bei allen folgenden Protokollen auch, hauptsächlich Pakete erzeugt, anhand deren man benutzerspezifische Daten auswerten kann. Bei einigen Protokollen wurde insbesondere die Verschlüsselung deaktiviert, damit die Pakete überhaupt gelesen werden können.

Das HTTP („*Hypertext Transfer Protocol*“) Protokoll dient zur Übertragung von Daten über ein Netzwerk, wobei es heutzutage hauptsächlich zur Übertragung von Webseiten eingesetzt wird.

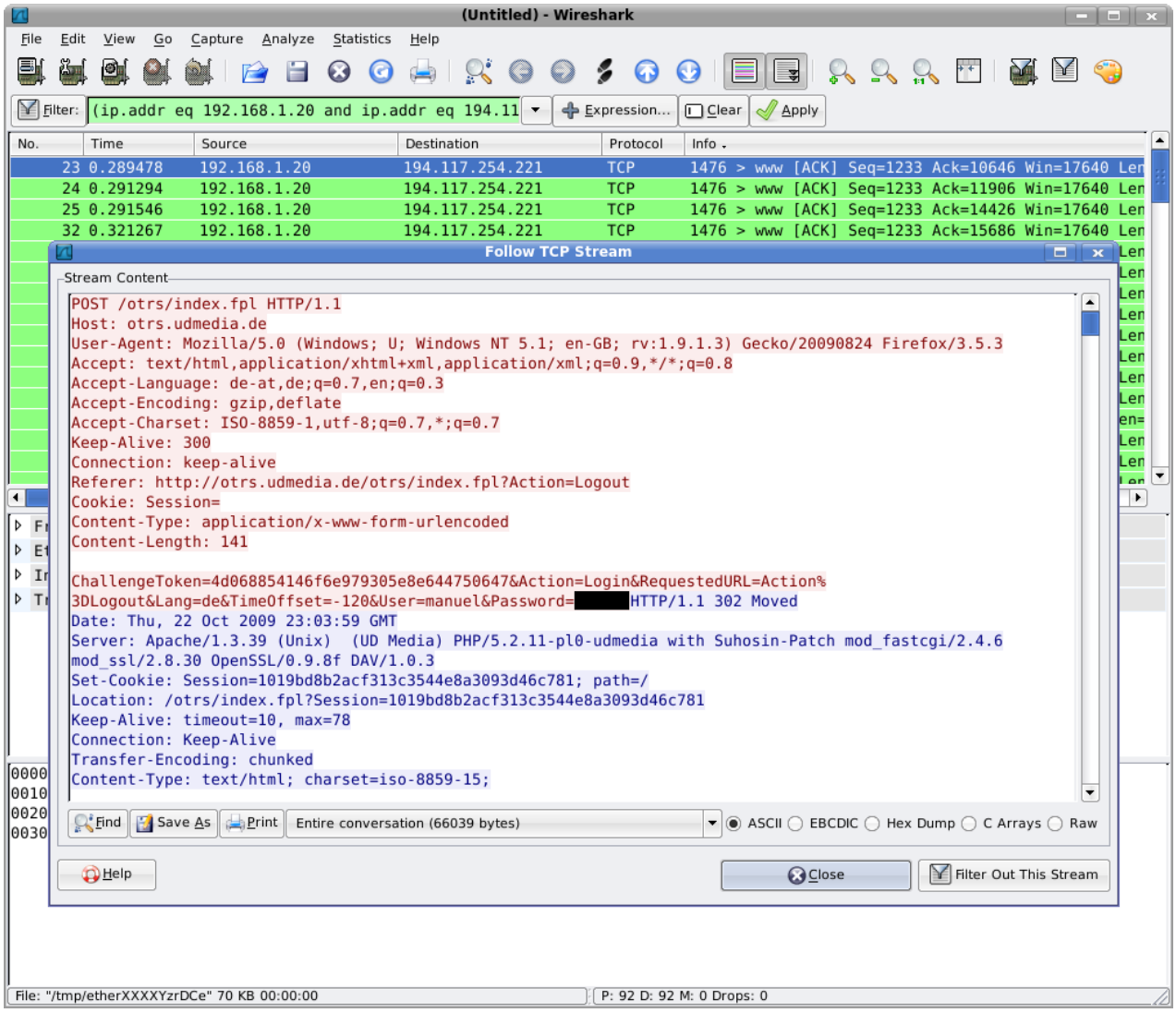
Als erstes Beispiel wurde die Webseite *http://news.orf.at* aufgerufen. Der rot hinterlegte Text kennzeichnet hierbei die Daten des Benutzer-PCs an den Webserver, der blaue die Daten des Webservers an den Benutzer-PC. Im ersten Teil sendet der Benutzer-PC einen „*HTTP GET*“-Request, worauf der Webserver mit einem „*HTTP 304 Not Modified*“ antwortet. Diese Antwort bedeutet, die angefragte Webseite enthält keine neuen Daten und muss somit nicht neu übertragen und kann aus dem Cache gelesen werden. Ob die Webseite neue Daten enthält, kann der Webserver durch das im ersten Request enthalten Datum im „*If-Modifiert-Since*“-Header feststellen.

Im dritten Teil wird ein „*HTTP GET*“-Request für ein Bild an den Webserver geschickt. Die weiteren Daten sind im Screenshot nicht ersichtlich.



Im zweiten Beispiel wurde ein „*HTTP POST*“-Request durchgeführt. Dies wird üblicherweise bei Formularen verwendet, um meist benutzerspezifische Daten an den Webserver zu übertragen, dabei aber diese Daten nicht in der URL und somit in der Browser History für andere sichtbar zu machen. Meist ist jeder Login ein solche Request, weshalb wir einen solchen auch als Beispiel gewählt haben.

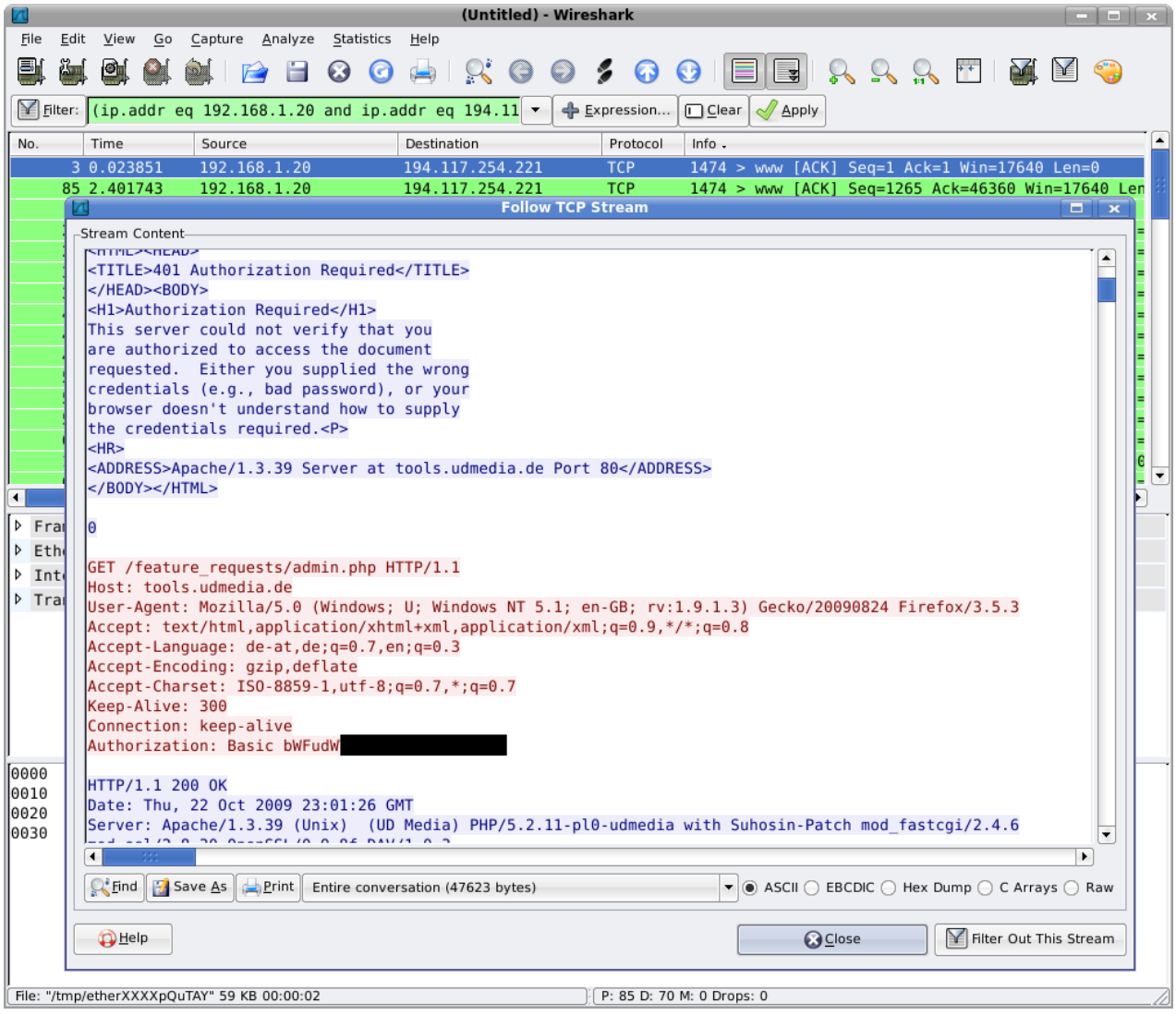
Die an den Webserver übertragenen Daten sieht man am Ende des Requests an den Webserver (roter Teil). Insbesondere Benutzername („*User*“) und Password („*Password*“) ist in der Paket-Aufzeichnung deutlich ersichtlich.



Als drittes Beispiel für das HTTP-Protokoll, haben wir uns für die Authentifizierung (*HTTP 401 Authorization Required*) entschlossen. Dies ist ein HTTP-Requesttyp um eine direkte Authentifizierung des Benutzers über den Webbrowser zu erzwingen. Üblicherweise wird dies durch ein schlichtes Popup des Webbrowsers mit Eingabefeldern für Benutzername und Passwort dargestellt.

Auch bei diesem Requesttyp liegen die Benutzerdaten im Klartext vor, sind jedoch zusätzlich noch im Syntax „<benutzername>:<password>“ Base64 kodiert. Die Base64-Kodierung dient hierbei zur Übertragung von Daten, die nicht im ASCII-Zeichensatz enthalten sind, wird also nicht zur Erhöhung der Sicherheit angewendet! - Zur Dekodierung muss also die Zeichenkette zuerst Base64 dekodiert werden und kann dann gelesen werden.

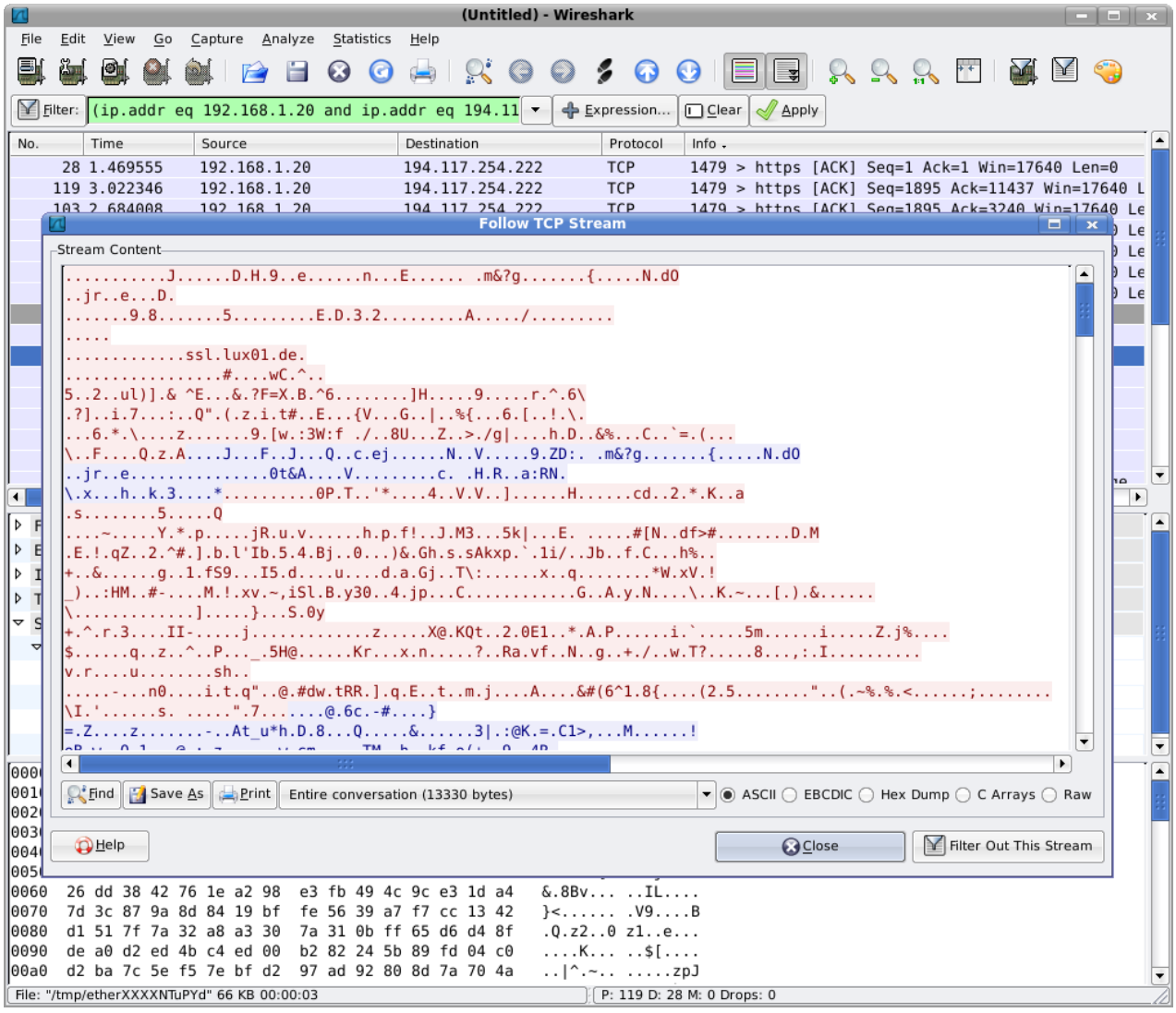
Zur Vollständigkeit sei erwähnt, das die HTTP-Authentifizierung auch Methoden unterstützt, bei dem nur ein Hash-Wert aus Benutzerdaten und Challenge (vom Webserver erstellt) übertragen wird. Hier sind die Benutzerdaten selbstverständlich nicht lesbar.



Protokoll: HTTPS

Als zweites Protokoll wurde *HTTPS* ausgewählt. Hierbei werden die Daten der HTTP-Übertragung zusätzlich noch mit *SSL/TLS* verschlüsselt. SSL bzw. TLS dient zur verschlüsselten Übertragung von Daten, basiert auf „a*symmetrischen Kryptoverfahren*“ und unterstützt mehrere Algorithmen.

Wie man im Screenshot erkennen kann, ist das Lesen der übertragenen Daten nicht möglich. Grundsätzlich ist aber auch SSL/TLS nicht vor „*man in the middle*“-Attacken geschützt, denn der Angreifer-PC könnte spezifische Pakete an beide Seiten senden, wodurch neue Schlüssel ausgehandelt werden. Damit wäre auch hier das lesen der Nutzdaten möglich.

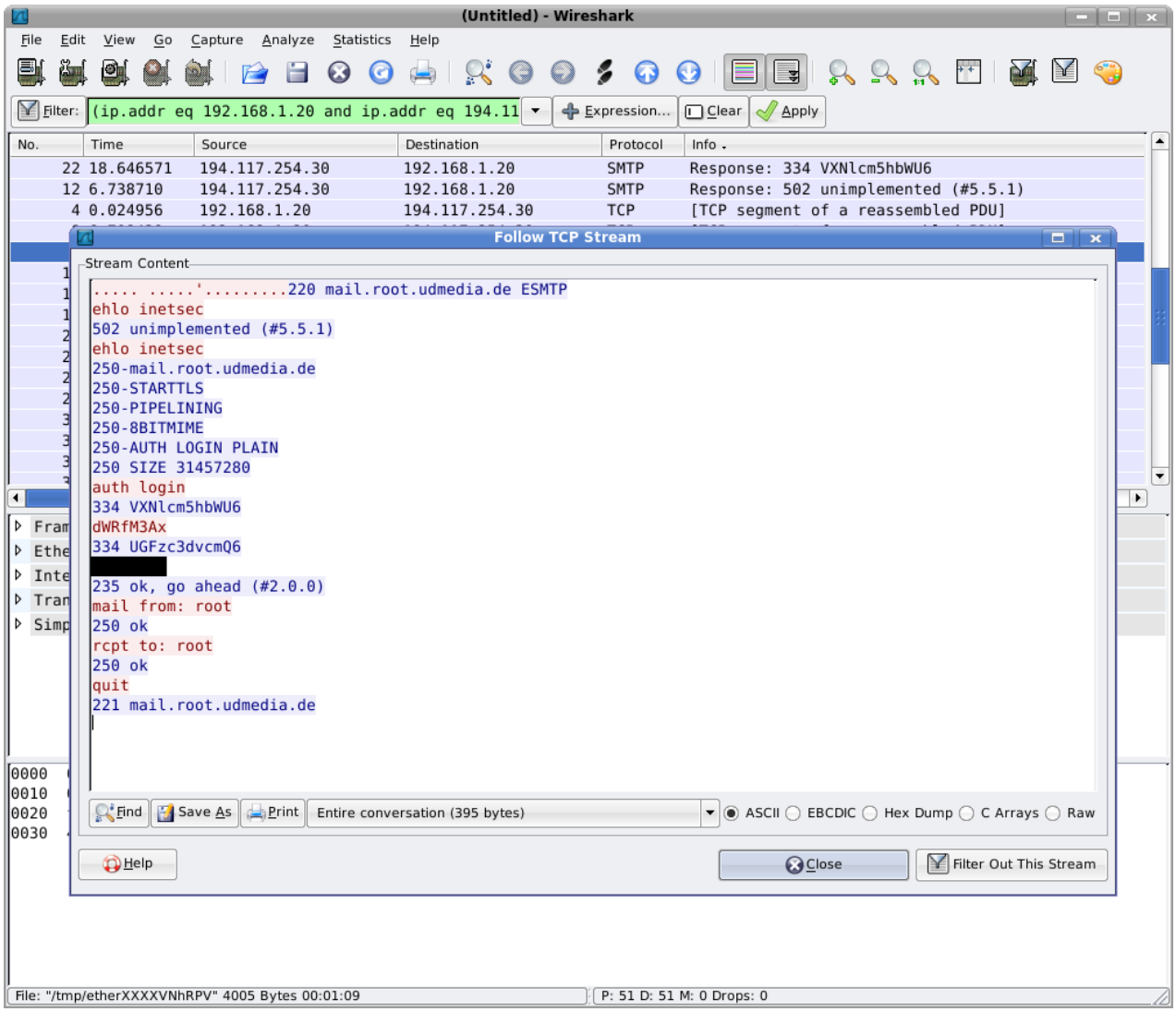


Protokoll: SMTP

Das SMTP „*Simple Mail Transfer Protocol*" Protokoll dient, wie der Name schon sagt, zum Versenden von E-Mails über einen E-Mail-Server. Damit das Versenden an für den E-Mail-Server fremde Benutzer (nicht lokale Benutzer, üblicherweise als „*Relaying*“ bezeichnet) möglich ist, muss sich der Benutzer zuerst authentifizieren. Dies ist beim SMTP-Protokoll, wie bei der HTTP-Authentifizierung, ebenfalls Base64 kodiert, jedoch in 2 einzelne Befehle ausgeteilt, wobei zuerst der Benutzername, anschließend das Passwort verschickt wird.

Auch hier sei zur Vollständigkeit erwähnt, dass SMTP eine Vielzahl von Authentifizierungen unterstützt. Neben dem, aus der HTTP Authentifizierung schon bekannten, Übertragung eines Hash-Wertes, ist auch die verschlüsselte Übertragung über SSL/TLS möglich.

Diese Aufzeichnung einer SMTP-Verbindung überträgt übrigens keine E-Mail, sondern beendet die Sitzung vorher. Üblicherweise werden die Daten nach einem „*DATA*“-Befehl übertragen.

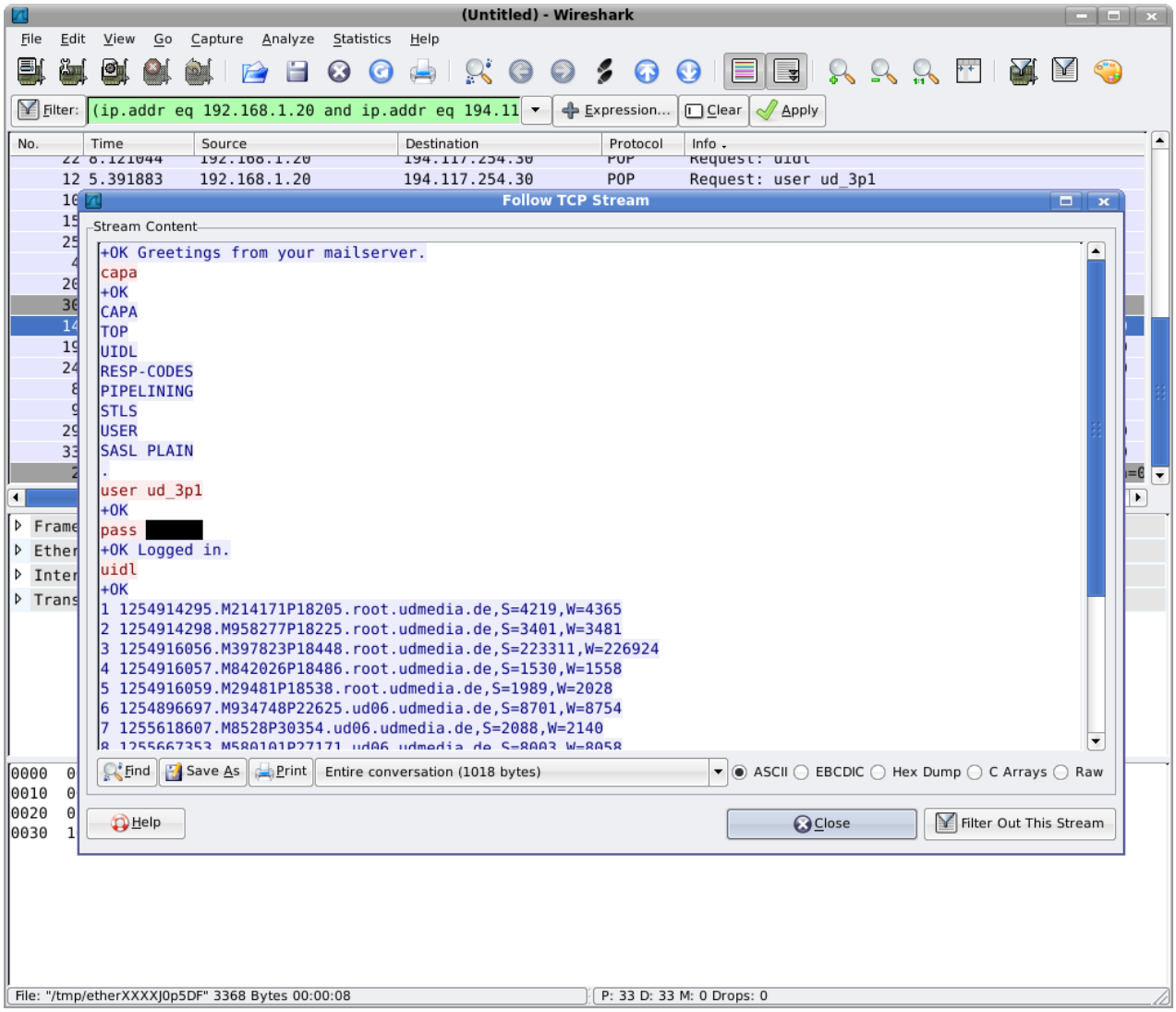


Protokoll: POP3

Das POP3 „*Post Office Protocol*“ Protokoll dient zum Abruf von E-Mails in einem Postfach. Es ist ein äußerst simples Protokoll und unterstützt beispielsweise kein verschieben von E-Mails in Ordner oder eine servserseitige Suche.

Die Authentifizierung erfolgt bei POP3 üblicherweise in Klartext, sofern nicht SSL/TLS verwendet wird. Die Authentifizierungsdaten werden hierbei direkt in Klartext übertragen.

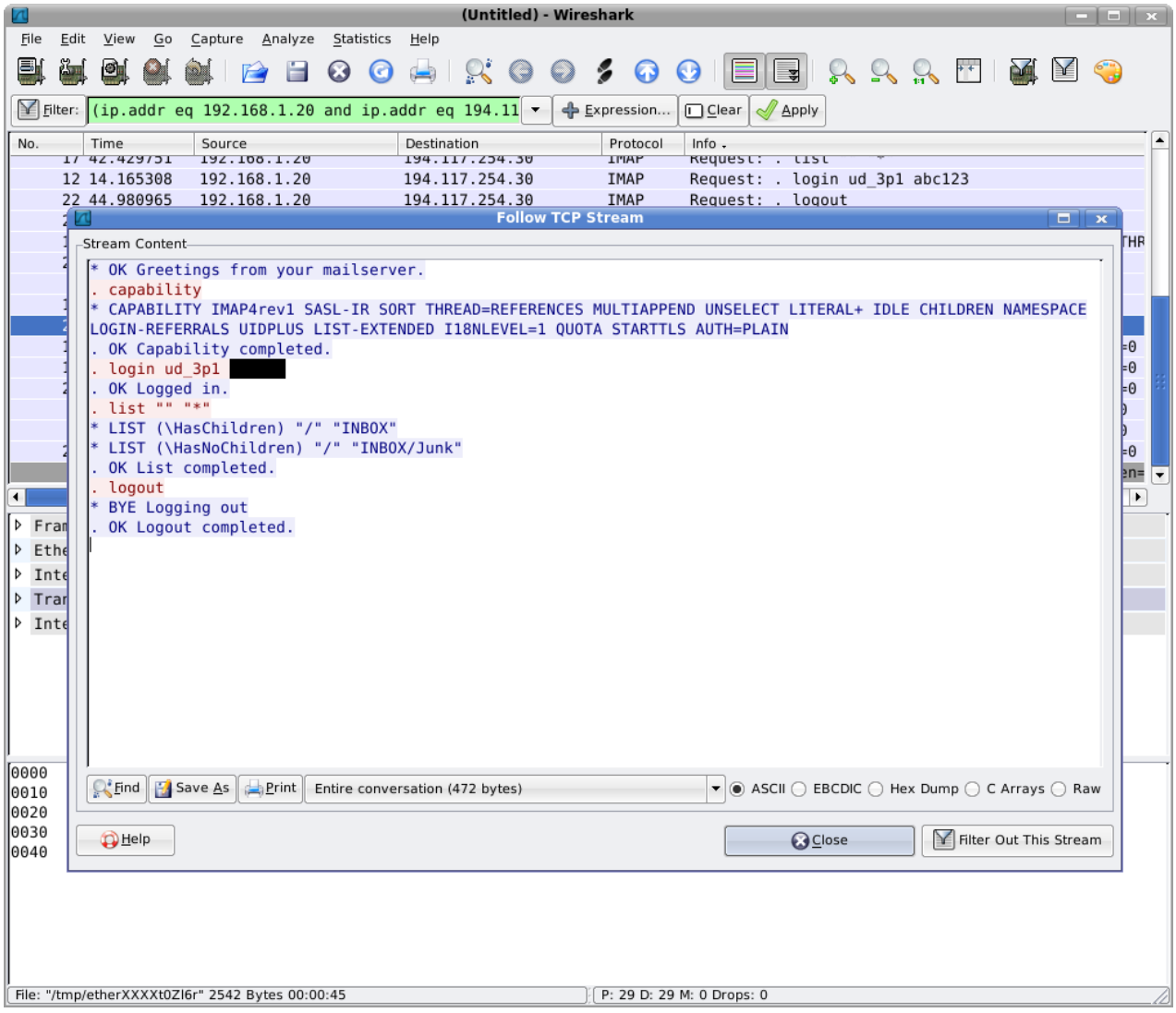
In der Aufzeichnung ersichtlich, ist zudem eine Liste von E-Mails, die in dem Postfach enthalten sind. Der Inhalt der E-Mails kann über den Befehl „*RETR*“ abgerufen werden.



Protokoll: IMAP

Das IMAP „*Internet Message Access Protocol*“ Protokoll, dient wir das POP3-Protokoll, zur Übertragung von E-Mails, unterstützt jedoch eine Vielzahl an zusätzlichen Funktionen. Üblicherweise wird es verwendet, um E-Mails am Server zu belassen, gleichzeitig jedoch effizient mit einer hohen Anzahl an E-Mails zu arbeiten. So werden die Befehle meist serverseitig ausgeführt.

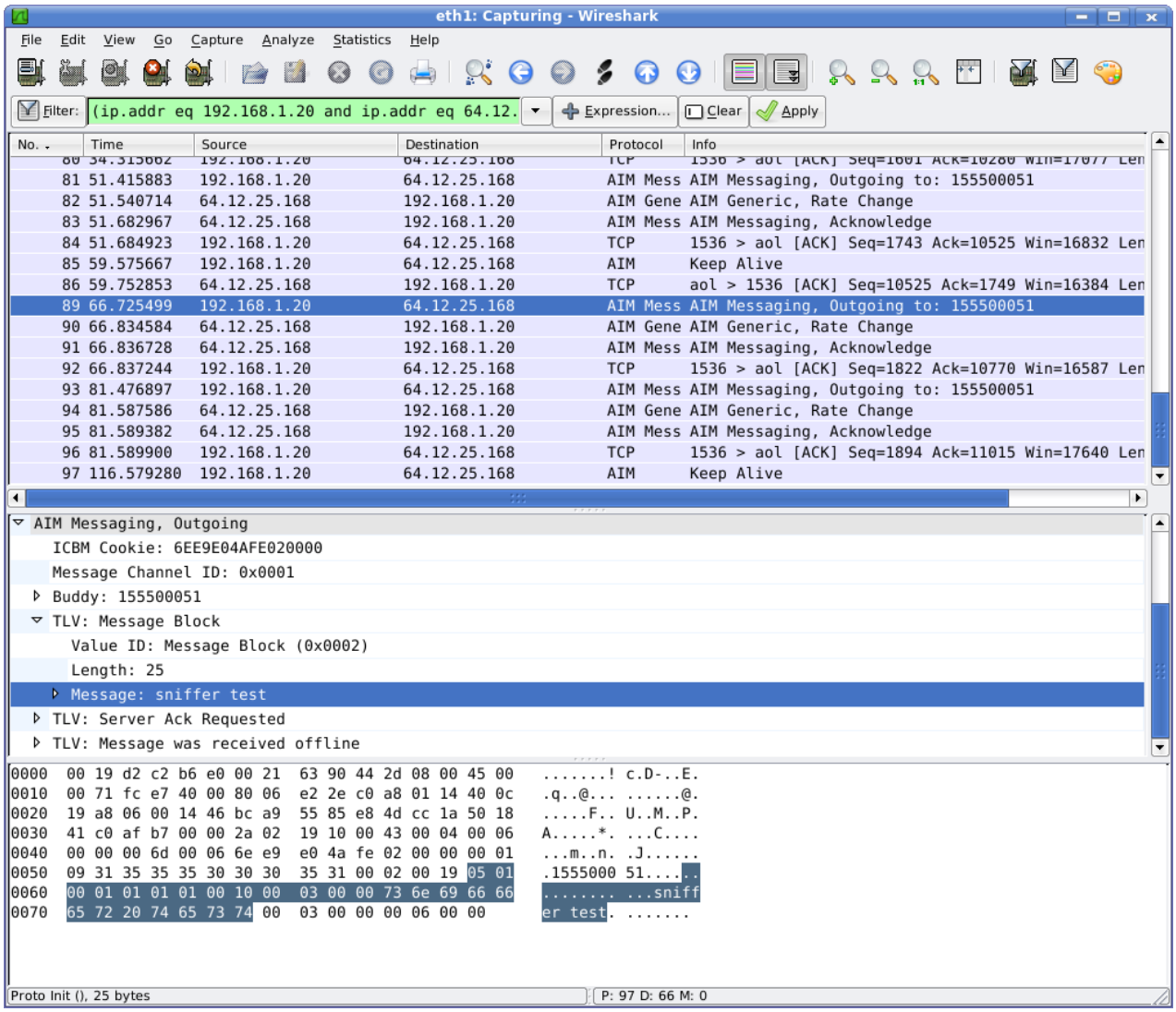
Der Login bei IMAP in IMAP erfolgt ebenfalls im Klartext, sofern kein Login über eine Hash-Wert verwendet wird. Ebenfalls möglich ist die sichere Verwendung von SSL/TLS.



Protokoll: AOL

Das AOL- bzw. ICQ-Protokoll ist ein Chatprotokoll und wurde ursprünglich von AOL entwickelt. Der Login hierbei erfolgt heutzutage zwingend über eine Hash-Wert Berechnung, weshalb das nicht Passwort lesbar ist. Die Übertragung der einzelnen Nachrichten erfolgt jedoch nach wie vor im Klartext.

In diesem Screenshot ersichtlich ist eine ausgehende Nachricht an den Benutzer mit der Nummer *15xy00xy1* (zensiert) mit dem Inhalt „*sniffer test*“.



Protokoll: Jabber

Das Jabber-Protokoll, dient wie das AOL-Protokoll, zum Chatten, ist jedoch als offener Standard definiert. Das heißt jeder kann einen Jabber-Server sowie -Client entwickeln und man ist nicht an einen einzelnen Anbieter gebunden.

Wie beim AOL-Protokoll ist der Login ebenfalls Hash-Wert-Verschlüsselt, die Nutzdaten jedoch nach wie vor im Klartext, und werden in XML übertragen. In diesem Screenshot ersichtlich ist eine ausgehende Nachricht an den Benutzer „exy-b@jabber.org“ (zensiert) mit dem Inhalt „*sniffer test*“.

Zur Vollständigkeit sei auch hier erwähnt, dass auch Jabber die Verschlüsselung über SSL/TLS unterstützt.