technische universität
dortmund

Seminararbeit

Packetsniffer

Anne-Cécile Klaassen
31. Januar 2014

Gutachter: Prof. Dr. Jan Jürjens
Dr. Thomas P. Ruhroth

Prof. Dr. Jan Jürjens Lehrstuhl 14 Software Engineering
Fakultät Informatik
Technische Universität Dortmund
Otto-Hahn-Straße 14
44227 Dortmund
http://www-jj.cs.uni-dortmund.de/secse

Anne-Cécile Klaassen
anne-cecile.klaassen@tu-dortmund.de
Matrikelnummer: 147663
Studiengang: Bachelor Angewandte Informatik

Werkzeugunterstützung für sichere Software
Thema: Packetsniffer

Eingereicht: 31. Januar 2014

Betreuer: Jens Bürger

Prof. Dr. Jan Jürjens Lehrstuhl 14 Software Engineering
Fakultät Informatik
Technische Universität Dortmund
Otto-Hahn-Straße 14
44227 Dortmund

Ehrenwörtliche Erklärung

Ich erkläre hiermit ehrenwörtlich, dass ich die vorliegende Arbeit selbstständig angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Dortmund, den 31. Januar 2014

Anne-Cécile Klaassen

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

In dieser Ausarbeitung beschäftige ich mich mit dem Thema *„Was ist Packetsniffing? Ist es als alltägliches Werkzeug für die Sicherheit in meinem System zu gebrauchen?“*.
Um diese Frage zu beantworten werde ich zunächst Packetsniffing definieren um dann im Anschluss das Programm Wireshark vorzustellen, dass als Werkzeug für Packetsniffing dienen soll. Mithilfe dieses Programms erläutere ich dann anhand eines simplen Beispiels die grundlegende Vorgehensweise.
Zuletzt möchte ich die verschiedenen Anwendungsmöglichkeiten dieser Methodik aufzeigen um dann ein Fazit aus meiner Arbeit zu ziehen und so eine Antwort auf die anfängliche Frage finden zu können.

2 Was ist Netzwerkanalyse und Packet Sniffing überhaupt?

Um zu entscheiden ob Packetsniffing als Werkzeug für sichere Systeme brauchbar ist, muss zunächst bekannt sein, was Netzwerkanalyse und Packetsniffing überhaupt sind.
Unter dem Begriff Netzwerkanalyse versteht man eine Prüfung eines Kommunikationsnetzes, wobei z.B. analysiert wird, welche zwei Clients miteinander kommunizieren, worüber, was und wieviel sie kommunizieren, welche Protokolle genutzt werden oder welche Applikationen verwendet werden.

Packetsniffing bezeichnet hingegen nur eine mögliche Art einer Netzwerkanalyse. Dabei wird der Datenverkehr aufgezeichnet indem jedes Paket abgefangen wird und auf das benutzte Netzwerkprotokoll überprüft. Ist das eingesetzte Protokoll bekannt, so lassen sich die in dem Paket enthaltenen Daten problemlos auswerten.

3 Wie funktioniert Packetsniffing?

Ein Packet Sniffer ist eine Hardware oder Software, die Packetsniffing betreibt. Da Hardware Sniffer heutzutage aber kaum noch Verwendung findet, wird im weiteren Verlauf nur auf Software Sniffer eingegangen.
Beim Packetsniffing wird nun eine solche Software an möglichst passender Stelle im Netzwerk eingesetzt. Dies ist wichtig zu beachten, da z.B. ein Switch dafür sorgen kann, der Datenverkehr der nicht für den eigenen Computer gedacht ist, nicht zu sehen ist.

3.1 Das Programm Wireshark

Wireshark *(früher: Ethereal)* ist ein Open-Source-Programm zur Analyse von Netzwerk-Kommunikationsverbindungen [Wik14] und das am weit verbreitetste Werkzeug für Netzwerkanalyse. Aktuell befindet es sich in der Version 1.11.

Um ein Netzwerk zu beobachten und mitzuschneiden braucht Wireshark Zugriff auf die Netzwerkkarte. Um diesen Zugriff zu vereinfachen gibt es für jedes Betriebssystem verschiedene Programmbibliotheken, Windows benutzt zum Beispiel WinPcap, wohingegen Linux die Bibliothek Libpcap benötigt. Im folgenden werde ich an einem Beispiel verdeutlichen, wie Packetsniffing mit Wireshark funktioniert.

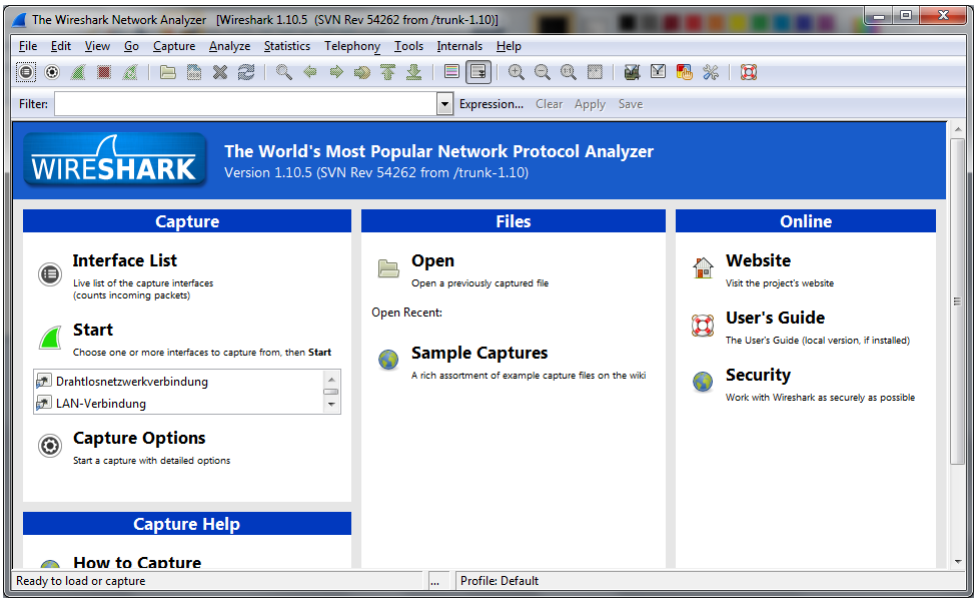

Abbildung 3.1: Wireshark Startbildschirm

In Abbildung 3.1 sieht man Wiresharks Startbildschirm. Um nun einen Mitschnitt zu erstellen, muss man das gewünschte Netzwerkinterface auswählen und anschließend auf „Start“ drücken. Dies führt dazu das eine neue Ansicht geöffnet wird und ungefiltert jedes Paket und dessen Informationen aufgenommen werden.

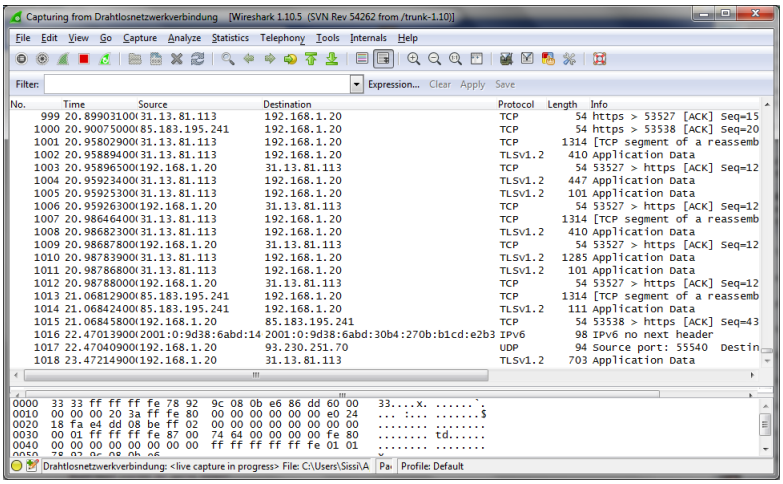

Abbildung 3.2: Ansicht nach ungefilterter Aufnahme

Eine solche ungefilterte Aufnahme wie in Abbildung 3.2 ist allerdings in den meisten Fällen eher unpraktisch, da unübersichtlich. Deshalb gibt es die Möglichkeit entweder bereits vorgefertigte oder selbst erstellte Filter anzuwenden.

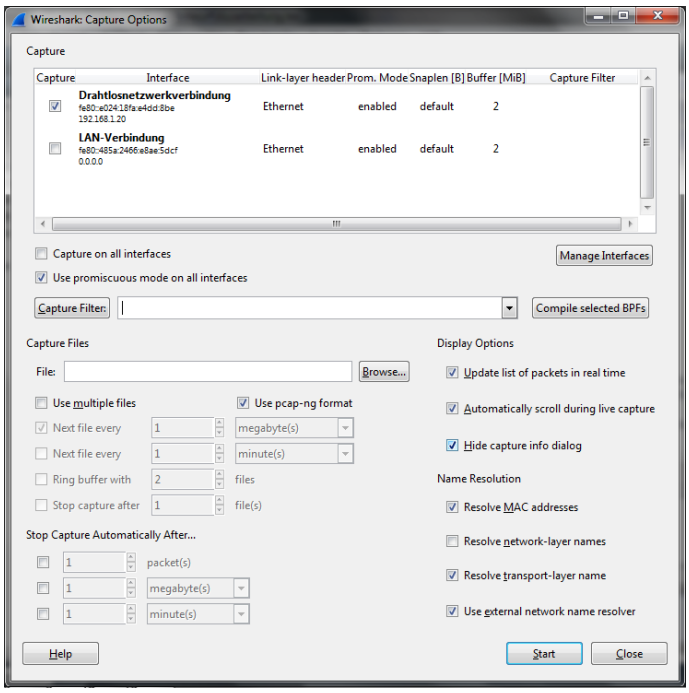

Abbildung 3.3: Wireshark „Capture Options“-Fenster

In Abbildung 3.3 sind die „Capture Options“ zu sehen, also Einstellungsmöglichkeiten den Mitschnitt weiter zu konfigurieren. In diesen Einstellungen kann man angeben, welche Netzwerkinterfaces zu überwachen sind, ob nur der Verkehr der Maschine, die Wireshark ausführt oder ob der Verkehr des ganzen Netzwerkes, soweit möglich, aufgezeichnet werden soll (der sogenannte *promiscious mode*).
Ebenso lässt sich die Ansicht der Pakete verändern und einstellen ob der Mitschnitt des Netzwerkes als Datei bzw. in mehreren Dateien gespeichert werden sollen.

Um nun die Anzeige der Pakete zu filtern, ist aber nur das „Capture Filter“-Feld interessant. Hier gibt es die Möglichkeit die Filtereigenschaft direkt in das Textfeld zu schreiben, oder durch einen Mausklick auf den Knopf „Capture Filter“ ein Fenster zu öffnen, wo bereits gespeicherte Filter ausgewählt werden können.

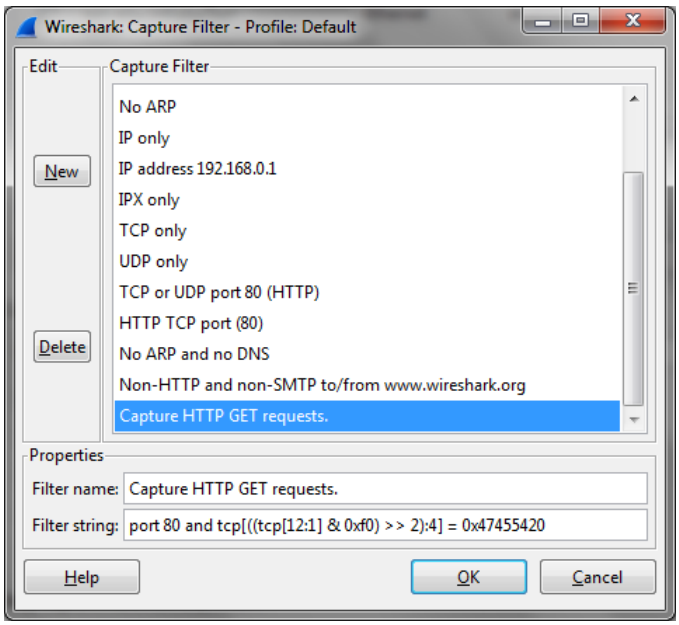

Abbildung 3.4: Wireshark „Capture Filter“-Fenster mit eigenem Filter

In der Abbildung 3.4 sieht man nun die vorgefertigten Standardfilter, die von Wireshark zur Verfügung gestellt werden. Für dieses Beispiel möchte ich mich nur auf Pakete konzentrieren, die ein HTTP GET-Request darstellen, daher habe ich einen neuen Filter namens „Capture HTTP GET requests“ erstellt und ausgewählt, der (vereinfacht ausgedrückt) nach dem TCP-Header nach den Bytes ’G’, ’E’ und ’T’ sucht. Nach Bestätigung dieser Auswahl startet eine neue Aufnahme, die dieses Mal nur Pakete des HTTP-Protokolls anzeigt, die ein GET-Request darstellen. Bei einem Aufruf der Website *www.tu-dortmund.de* ist nun die in Abbildung 3.5 gezeigte Ansicht sichtbar.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.1.20	129.217.231.107	HTTP	428	GET /uni/Uni/index.html HTTP/1.1
2	0.057344000	192.168.1.20	129.217.231.107	HTTP	456	GET /Medienpool/tu_Stylesheets/screen.css HTTP/1.1
3	0.066487000	192.168.1.20	129.217.231.107	HTTP	462	GET /Medienpool/tu_Stylesheets/custom_links.css HTTP/1.1
4	0.067771000	192.168.1.20	129.217.231.107	HTTP	455	GET /Medienpool/tu_Stylesheets/print.css HTTP/1.1
5	0.068876000	192.168.1.20	129.217.231.107	HTTP	455	GET /Medienpool/tu_Stylesheets/opera.css HTTP/1.1
6	0.070083000	192.168.1.20	129.217.231.107	HTTP	445	GET /Medienpool/JavaScripts/mailMaskierung.js HTTP/1.1
7	0.177594000	192.168.1.20	129.217.231.107	HTTP	470	GET /Medienpool/Seitenbilder/logo_tu.png HTTP/1.1
8	0.178918000	192.168.1.20	129.217.231.107	HTTP	475	GET /uni/meldungen/2014-01/bilder/zeugnis.jpg HTTP/1.1
9	0.181382000	192.168.1.20	129.217.231.107	HTTP	479	GET /uni/meldungen/2014-01/bilder/professoren.jpg HTTP/1.1
10	0.183164000	192.168.1.20	129.217.231.107	HTTP	473	GET /uni/meldungen/2014-01/bilder/mintu.jpg HTTP/1.1
11	0.184413000	192.168.1.20	129.217.231.107	HTTP	471	GET /uni/meldungen/2014-01/bilder/sfs.jpg HTTP/1.1
12	0.185622000	192.168.1.20	129.217.231.107	HTTP	484	GET /uni/Medienpool/Bereichsbilder/werb-logos/UAMR.jpg HTTP/1.1
13	0.266800000	192.168.1.20	129.217.231.107	HTTP	490	GET /uni/Medienpool/Bereichsbilder/werb-logos/Zertfamger.jpg HTTP/1.1
14	0.363140000	192.168.1.20	129.217.231.107	HTTP	488	GET /uni/Medienpool/Bereichsbilder/werb-logos/vielfalt.jpg HTTP/1.1
15	0.363261000	192.168.1.20	129.217.231.107	HTTP	511	GET /Medienpool/tu_Stylesheets/css-icons/icon_english.png HTTP/1.1
16	0.429875000	192.168.1.20	129.217.231.107	HTTP	475	GET /uni/Medienpool/Bereichsbilder/tu_uni.jpg HTTP/1.1
17	0.438834000	192.168.1.20	129.217.231.107	HTTP	501	GET /Medienpool/tu_Stylesheets/css-icons/tu-liste.png HTTP/1.1
18	0.439641000	192.168.1.20	129.217.231.107	HTTP	510	GET /Medienpool/tu_Stylesheets/css-icons/icon_extern.png HTTP/1.1
19	0.440649000	192.168.1.20	129.217.231.107	HTTP	508	GET /Medienpool/tu_Stylesheets/css-icons/icon_mail.png HTTP/1.1

Abbildung 3.5: Aufruf *www.tu-dortmund.de*

Die Abbildung 3.5 lässt erkennen, das tatsächlich nur Pakete des Protokolltyps HTTP zu sehen sind. Ebenfalls sieht man an der „Info-Spalte“, das jedes dieser Pakete auch nur ein GET-Request verschickt hat.
Da diese Webseite zum ersten Mal aufgerufen wurde, mussten entsprechend viele Daten vom Server abgefragt werden. Anhand dieser Übersicht wird euch deutlich welche IP-Adresse der Seite *www.tu-dortmund.de* zugewiesen wurde (129.217.231.107).
Für eine genauere Inspektion der Pakete ist nun ein Blick in die jeweilige Detailansicht vonnöten.

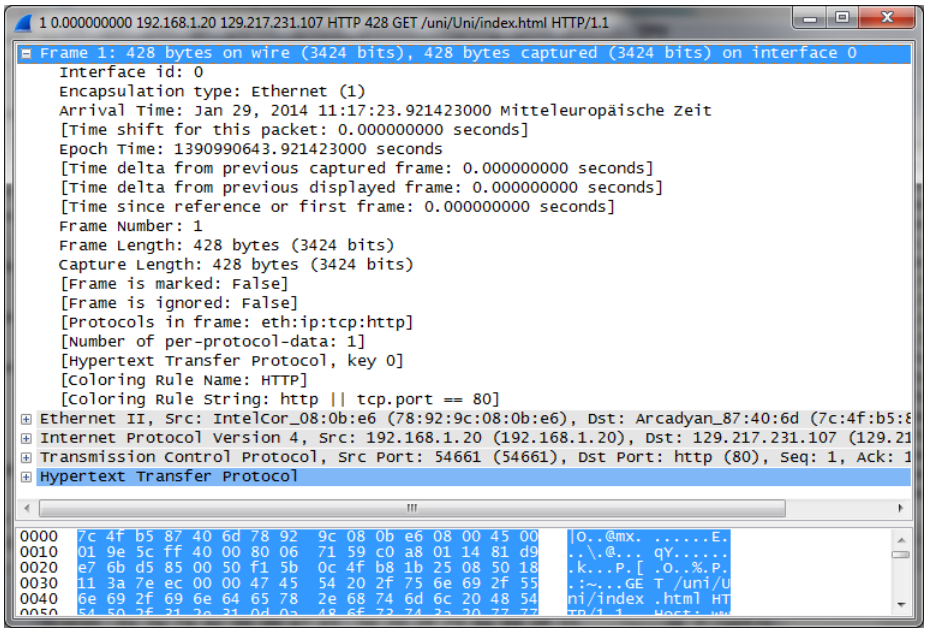

Abbildung 3.6: Paket Detailansicht

Eine solche Detailansicht sieht so aus wie in Abbildung 3.6 dargestellt. Die Daten, die unter „Frame“ zu finden sind, geben Informationen darüber, wann das Paket erfasst wurde, die Zeitdifferenz zum vorherigen Paket und die Größe und aufgezeichnete Größe des Pakets.

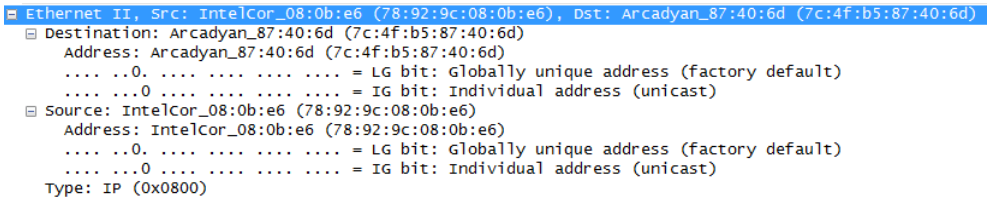

Abbildung 3.7: Paket Detailansicht Reiter Ethernet

In der nächsten Zeile der Details gibt es die sogenannten Ethernet-Daten wie in Abbildung 3.7 zu sehen sind. Hier lassen sich die MAC-Adressen des Absenders und des Empfängers ablesen. In diesem Beispiel lässt sich erkennen, dass der Absender einen Router verwendet, der von der Firma Arcadyan gefertigt wurde.

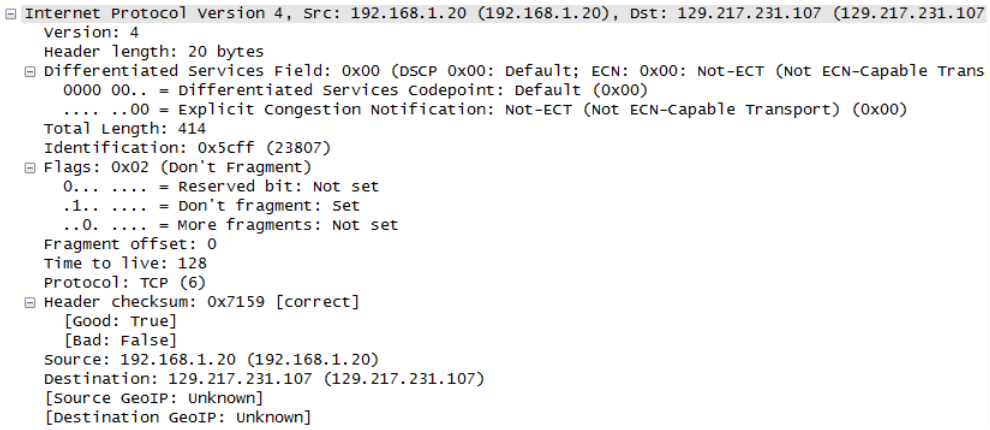

Abbildung 3.8: Paket Detailansicht Reiter IPv4

Durch den in Abbildung 3.8 dargestellten IPv4-Datensatz sind die IP-Adressen des Absenders und Empfängers einsehbar. Ebenso werden die genutzten IP-Flags und die sogenannte Time to Live (TTL) angezeigt.

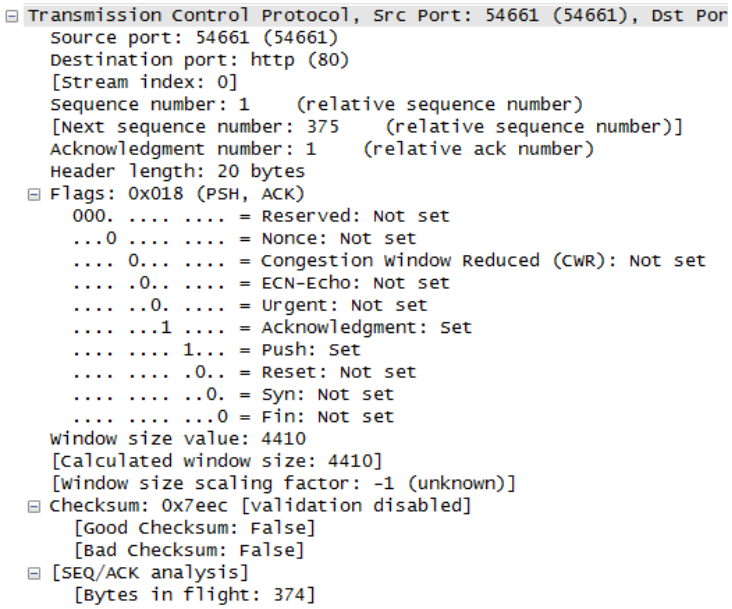

Abbildung 3.9: Paket Detailansicht Reiter TCP

Unter dem TCP-Reiter (Abbildung 3.9) lassen sich Informationen über den verwendeten Port des Absenders sowie dem des Empfängers erkennen. Zusätzlich sind die Sequenznummer, Acknowledgementnummer, die verwendeten Flags und die Prüfsumme abzulesen.

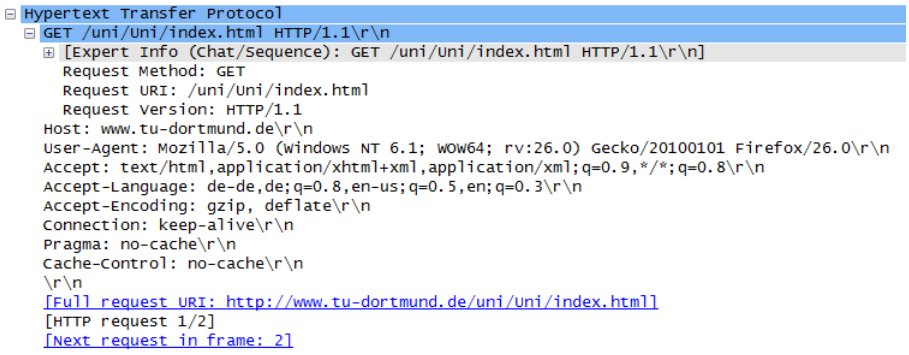

Abbildung 3.10: Paket Detailansicht Reiter HTTP

In dem letzten Informationsreiter findet man die Details zum HTTP-Protokoll wie in Abbildung 3.10 zu sehen. Hier ist die Aufrufmethode (GET), die URL sowie die URI zu erkennen.
Ebenfalls finden sich Informationen über den Absender des Requests. In diesem Beispiel verwendete ich den Internetbrowser Mozilla Firefox, was in der Zeile „User Agent“ deutlich ablesbar ist. Der Browser akzeptiert die Sprachen Deutsch (de-de), Englisch (en) und amerikanisches Englisch (en-us).
Es befanden sich keine Daten im Cache (no-cache) und da auch kein Cookie gesetzt war, finden sich zu diesem keine Informationen. Hätte es einen Cookie gegeben, wären die Daten des Cookies hier einsehbar gewesen.

Anhand dieses Beispiels sollte klar geworden sein, wie schnell und einfach Pakete ausgelesen und die Informationen analysiert werden können.

3.2 Wozu wird es benutzt?

Für Packetsniffing gibt es mehrere mögliche Anwendungsgebiete, wie z.B. Diagnose eines Netzwerkes. In dem Falle das eine Internetverbindung viel zu langsam erscheint, lässt sich anhand einer Analyse durch Wireshark herausfinden, ob dies nun am Internet Provider, dem Website-Server, dem eigenen Netzwerk, dem DNS-Server oder dem Computer liegt.[CHA10, p. 5] Ebenso kann man den genauen Punkt finden, an dem Pakete eventuell verloren gehen.

Außerdem lässt sich Packetsniffing für die Optimierung eines Netzwerkes einsetzen, indem man die aktuelle Ausnutzung der Bandbreite analysiert oder um die effizienteste Paketgröße zu bestimmen.

Auch auf dem Gebiet der Netzwerksicherheit können Packetsniffer hilfreich sein, da sie es ermöglichen sicherzustellen, das Datenpakete nur die Wege nehmen, die sie nehmen sollen und nicht eventuell über fremde Server geleitet werden. In dem Fall das das Netzwerk von jemand anderem kompromittiert wurde, kann der gespeicherte Datenverkehr auch als späteres Beweismittel dienen.

Gleichzeitig allerdings kann Packetsniffing auch für Angriffe auf ein fremdes Netzwerk genutzt werden, indem wichtige Daten wie z.B. Passwörter mitgeschnitten werden.

4 Fazit

Zum Schluss steht also nur noch die Frage offen, ob Packetsniffing für die Sicherheit in einem System zu gebrauchen ist.
Mit Packetsniffing lässt sich ziemlich leicht Datenverkehr in einem Netzwerk mitschneiden und auswerten, wie in Kapitel 3 gezeigt. Dadurch ist es zumindest für die Diagnose von Netzwerkproblemen gut geeignet, im Bereich der Sicherheit ist es jedoch etwas komplizierter.
Eindringlinge im eigenen Netzwerk lassen sich per Packetsniffing nur dann erkennen, wenn sie Pakete umleiten. Sollte es schon so weit gekommen sein, dass ein Rechner innerhalb des Netzwerkes kompromittiert wurde und selbst den Datenverkehr mitschneidet um sicherheitsrelevante Daten wie Passwörter abzufangen, ist es unmöglich dies zu erkennen, da Packetsniffing nur erfasst, selbst aber keine Daten versendet. Es ist also völlig passiv. Für den schlechtmöglichsten Fall ist Packetsniffing also nur wenig hilfreich.
Zusätzlich wird die Paketanalyse umso aufwändiger, je mehr Daten es zu analysieren gibt und im Bezug auf Wireshark können die gesammelten Daten schnell mehr Speicherplatz benötigen als durch den Rechner zur Verfügung steht.
Da selbst die in meiner Literatursuche gefundenen Bücher das Thema Sicherheit wenn überhaupt nur anschneiden und nicht vertiefen, komme ich zu dem Schluss, dass Packetsniffing als Diagnosewerkzeug bei konkretem Verdacht zwar sehr nützlich sein kann, es als dauerhaftes Werkzeug im Rahmen der Sicherheit in einem System allerdings zu aufwändig und zu teuer ist.

Literatur

[CHA10] Laura CHAPPELL. *Wireshark network analysis*. 1. Auflage. San Jose: Chappell Univ., 2010.

[SAN11] Chris SANDERS. *Practical Packet Analysis : using Wireshark to solve real-world network problems*. 2. Auflage. San Francisco, CA: No Starch Press, 2011.

[Wik14] Wikipedia. 2014. URL: `http://de.wikipedia.org/wiki/Wireshark`.