

HÖHERE TECHNISCHE BUNDESLEHRANSTALT HOLLABRUNN

Höhere Abteilung für Elektronik – Technische Informatik

Klasse/ Jahrgang:	Gruppe:	Übungsleiter:
3BHEL	1b	Prof. Stoll
Übungsnummer:	Übungstitel:	
5/4	Paketanalyse	
Datum der Übung:	Teilnehmer:	
22.05.2014	Dorner, Mötz, Oberhamberger, Prinz	
Datum der Abgabe:	Schriftführer:	Unterschrift:
05.06.2014	Mötz Florian	

	Beurteilung
Aufgabenstellung	
Dokumentation	
Messschaltungen	
Messtabellen	
Berechnungen	
Programmlistings	
Auswertung	
Diagramme	
Berechnungen	
Simulationen	
Schlussfolgerungen	
Kommentare	
Inventarliste	
Messprotokoll	
Form	
Summe	

Allgemeiner Teil

Titel der Übung: Paketanalyse

Übungsnummer: 5/4

Übungsplatz: 3

Datum der Übung: 22.05.2014

Klasse: 3BHEL

Schriftführer: Mötz Florian

Übungsteilnehmer: Dorner Fabian

Oberhamberger Christoph

Prinz Bernhard

Inhaltsverzeichnis

1. Snifferprogramm	4
1.1 Wireshark	4
1.2 Anleitung	4
2. Netzwerkaufbau	5
3. Ping-Request analysieren	6
3.1 Ethernet Header	6
3.1.1 Zieladresse	6
3.1.2 Quelladresse	6
3.1.3 Typ	6
3.2 IP Header	6
3.2.1 Version	6
3.2.2 Länge des Headers	6
3.2.3 Typ	7
3.2.4 Länge	7
3.2.5 Identifikation	7
3.2.6 Flags	7
3.2.7 Fragment Offset	7
3.2.8 Time to Live	7
3.2.9 Protokoll	8
3.2.10 Checksumme	8
3.2.11 Quell-IP-Adresse	8
3.2.12 Ziel-IP-Adresse	8
3.3 IP Daten	8
3.3.1 Typ	8
3.3.2 Code	8
3.3.3 Checksumme	8
3.3.4 Identifier	9
3.3.5 Sequenznummer	9
3.3.6 Daten	9
4. TCP Paket	9
4.1 TCP Header	9
4.1.1 Zieladresse	9
4.1.2 Quelladresse	9
4.1.3 Typ	10
4.1.4 IP Header	10
4.2 TCP Paket	10
4.2.1 Quell-Port	10
4.2.2 Ziel-Port	10
4.2.3 Sequenz-Nummer	10
4.2.4 Header-Länge	10
4.2.5 Flags	11
4.2.6 Window Size	11
4.2.7 Checksumme	11
4.2.8 Urgent Pointer	11
4.2.9 Options	11
5. Erkenntnis	12

1. Snifferprogramm

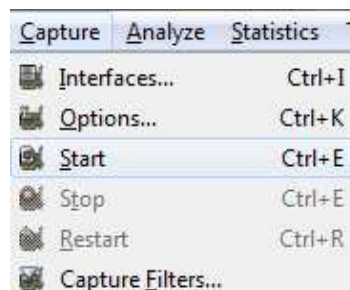
1.1 Wireshark

Wireshark ist ein Sniffing-Programm, mit welchem man alle Datenpakete, die am PC ankommen oder ausgehen, angesehen werden können.

Dieses kostenlose Programm wurde in der Übung dazu verwendet, Paketdaten mitzulesen und zu interpretieren.

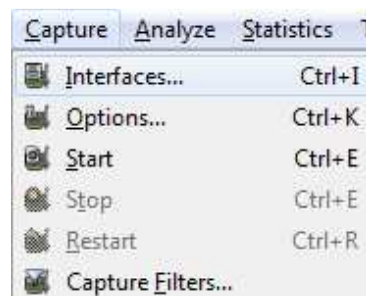
1.2 Anleitung

Nachdem man Wireshark gestartet hat, kann man ein neues Capture (= Aufnahme der Datenpakete) starten. Dazu kann man unter dem Menüpunkt Capture den Punkt Start auswählen.

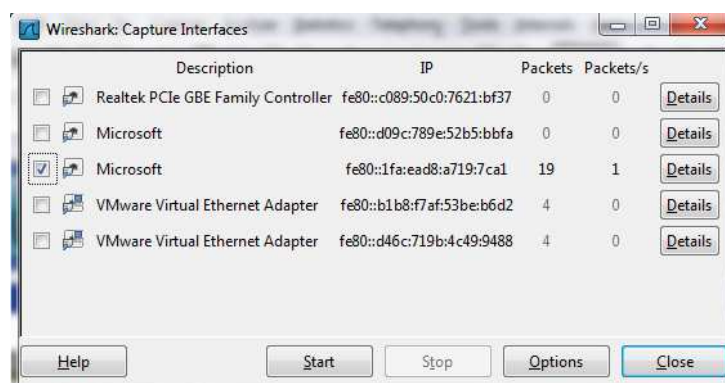


Wenn man das erste Mal nach dem Start des Programmes ein Capture starten will, muss man zuerst ein Interface wählen, von dem die Datenpakete aufgenommen werden sollen.

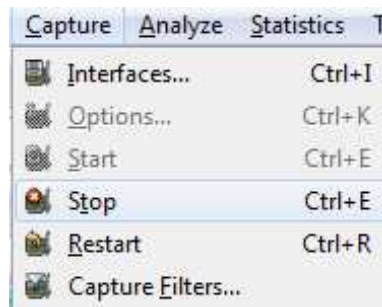
Dafür gibt es unter dem Menüpunkt Capture den Punkt Interface. Dieser öffnet ein neues Fenster, in welchem man das gewünschte Interface wählen kann.



Wenn man das gewünschte Interface gewählt hat, kann man das Capture mit einem Klick auf Start starten.



Um das Capture zu Beenden, gibt es unter dem Menüpunkt Capture den Punkt Stopp.

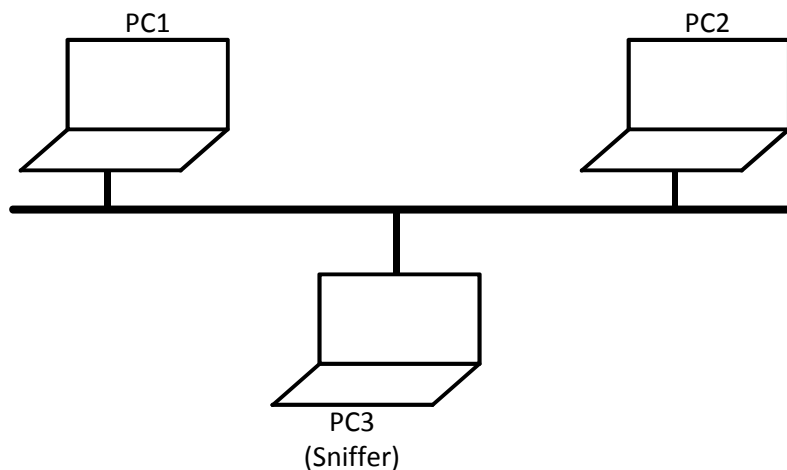


2. Netzwerkaufbau

Für diese Übung wurde ein Netzwerk bestehend aus insgesamt 3 PCs und einem Hub aufgebaut.

2 der PCs sollten miteinander kommunizieren (TCP-Pakete und Ping-Pakete), der 3. sollte diese Kommunikation "mitschniffen".

Zuerst wurde das Netzwerk mit einem Switch zwischen den 3 PCs aufgebaut. Der Übungsleiter wies uns jedoch nach einigen Fehlgeschlagenen „sniff“-Versuchen darauf hin, dass bei der Verwendung eines Switches keine Pakete, die an einen bestimmten Empfänger gedacht sind, zu einem anderen PC gesendet werden. Bei der Verwendung eines Hubs tritt dieses Problem nicht auf, da alle Pakete an alle in diesem Netzwerk befindlichen Geräte gesendet werden.



3. Ping-Request analysieren

Erster Teil der Laborübung war es, den Header eines Ping-Paketes zu analysieren. Dazu wurde von PC1 auf PC2 ein PING gesendet, PC3 agierte als Sniffer.

Das Paket bildet sich aus einem Ethernet Header, einem IP Header und der IP Data.

3.1 Ethernet Header

3.1.1 Zieladresse

Aus der Paketanalyse des Programmes Wireshark erkannten wir die ersten 6 Byte als folgende:

a0 b3 cc 44 a9 15

Diese 6 Byte entsprechen der MAC-Adresse des Empfängers, welcher in unserem Netzwerk PC1 war.

3.1.2 Quelladresse

Die nächsten 6 Byte entsprechen der MAC-Adresse des Senders, welcher in unserem Netzwerk PC2 war.

54 be f7 6d d5 ae

3.1.3 Typ

Die nächsten 2 Byte definierten, dass es sich um ein IP-Paket handelt.

08 00

3.2 IP Header

3.2.1 Version

Die nächsten 4 Bit definierten die IP-Version:

4

Der Inhalt bedeutet, dass es sich um ein IPv4 Paket handelte.

3.2.2 Länge des Headers

Die nächsten 4 Bit definierten die Länge:

5

Das bedeutet, dass sich in diesem Header 5 Wörter mit der Länge von 32 Bits befinden.

3.2.3 Typ

Das nächste Byte definierte den Typ:

00

Das bedeutete, dass es sich um keinen ECT-Transport handelt.

3.2.4 Länge

Die nächsten 2 Bytes definierten die Länge des Paketes, inklusive Header und Data.

00 3c

Die bedeutete, dass das gesamte IP-Paket 60 Bit lang war.

3.2.5 Identifikation

Die nächsten 2 Bytes dienen der Identifikation.

26 ca

Dies ist notwendig, um die einzelnen Pakete wieder in der richtigen Reihenfolge zu verbinden.

3.2.6 Flaggen

Die nächsten 3 Bit definieren bestimmte Flags:

binary: 000

Das erste Bit ist das „Reserve“ Bit, in diesem Fall nicht gesetzt.

Das zweite Bit ist das „Don't Fragment“-Bit, in diesem Fall nicht gesetzt.

Das dritte Bit ist das „More Fragments“ Bit, in diesem Fall nicht gesetzt.

3.2.7 Fragment Offset

Die nächsten 13 Bit definieren den Fragment Offsets.

binary: 00000000000000

Dies ist eine Nummer, die bei fragmentierten Paketen bestimmt, ab welcher Position innerhalb des Paketes das Fragment anfängt.

3.2.8 Time to Live

Die nächsten 2 Byte definieren die Time to Live (kurz: TTL)

80

Das definiert, dass nach 128 hops das Paket ungültig wird.

3.2.9 Protokoll

Das nächste Byte definiert das Protocol:

01

In unserem Fall bedeutet das, dass das Protokoll ICMP verwendet wird.

3.2.10 Checksumme

Die nächsten 2 Bytes sind Header-Checksums.

f3 66

Wireshark definierte diese Checksumme als richtig.

3.2.11 Quell-IP-Adresse

Die nächsten 32 Bit (4 Byte) definieren die Sender-IP Adresse.

a9 fe 0d 5c

3.2.12 Ziel-IP-Adresse

Die nächsten 32 Bit (4 Byte) definieren die Empfänger-IP Adresse.

a9 fe bf 37

3.3 IP Data

3.3.1 Typ

Das erste Byte des ICMP-Paketes definiert den Typ:

08

Dies bedeutet, dass es sich um ein „Echo (ping) request“ Paket handelt.

3.3.2 Code

Das erste Byte des ICMP-Paketes definiert den Code:

00

3.3.3 Checksumme

Die nächsten 2 Bytes definieren die Checksumme.

4d 0a

3.3.4 Identifier

Die nächsten 2 Byte bilden den Identifier.

00 01

Dieser ist notwendig, um die einzelnen Pakete wieder in der richtigen Reihenfolge zu verbinden.

3.3.5 Sequenznummer

Die nächsten 2 Byte bilden die Sequenznummer.

00 51

Die Sequenznummern ist für die Sicherstellung einer vollständigen Übertragung in der richtigen Reihenfolge und ohne Duplikate notwendig.

3.3.6 Daten

Die nächsten 32 Byte bilden die Daten.

61 62 63 64 65 66 67 68 69 8a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76 77 61 62 63 64 65 66 67 68 69

Diese Daten sind in ASCII-Zeichen:

abcdefghijklmnopqrstuvwxyz

Der Sinn dieser Daten konnte in der Übung nicht erfasst werden.

4. TCP Paket

4.1 TCP Header

4.1.1 Destination Address

Aus der Paketanalyse des Programmes Wireshark erkannten wir die ersten 6 Byte als folgende:

a0 b3 cc 44 a9 15

Diese 6 Byte entsprechen der MAC-Adresse des Empfängers, welcher in unserem Netzwerk PC1 war.

4.1.2 Source Address

Die nächsten 6 Byte entsprechen der MAC-Adresse des Senders, welcher in unserem Netz PC2 war.

54 be f7 6d d5 ae

4.1.3 Type

Die nächsten 2 Byte sind folgende:

08 00

Diese 2 Byte definieren, dass es sich um ein IP-Paket handelt.

4.1.4 IP Header

Der Header entspricht dem Muster des IP-Header der Pinganfrage (siehe Punkt 3.2).

Es gab nur inhaltliche Abweichungen zu dem IP-Header der Pinganfrage:

Die Länge betrug 48 Bit, die Identifikations-Bits weichten ab (HEX: 38 A9), bei den Flags war das „Don't Fragment“-Bit gesetzt und als Protokoll war TCP angegeben.

4.2 TCP Paket

4.2.1 Quell-Port

c6 25

Diese 2 Byte definierten den Port des Quell-PCs (Port 50725).

4.2.2 Ziel-Port

00 50

Die nächsten 2 Byte definierten den Port des Ziel-PCs (Port 80 [http]).

4.2.3 Sequenz-Nummer

b6 f5 b0 6e

Diese 4 Byte definierten die Sequenz-Nummer des Paketes.

4.2.4 Header Länge

7

Diese 4 Bit definierten die Länge des Headers.

4.2.5 Flags

Die nächsten 12 bits definierten Flags:

Bit	Name	Inhalt
1 – 3:	reserved	(not set)
4.	nonce	0
5.	Congestion Window Reduced (CWR)	0
6.	ECN-Echo	0
7.	Urgent	0
8.	Acknowledgement	0
9.	Push	0
10.	Reset	0
11.	SYN	1
12.	FIN	0

4.2.6 Window Size

20 00

Diese 2 Byte definierten die Größe des Zeitfensters (Window Size).

4.2.7 Checksumme

14 b5

Diese 2 Byte definierten die Checksumme.

4.2.8 Urgent Pointer

00 00

Diese 2 Byte definierten den Urgent-Pointer. Da dieser jedoch durch die Flags (siehe Punkt 4.2.5, 7. Bit) nicht gesetzt wurde, besteht der urgent-Pointer nur aus ungesetzten Bits.

4.2.9 Options

02 04 05 b4 01 01 04 02

Die letzten 8 Byte definierten diverse Optionen:

- Maximum Segment size (4 Byte)
- No Operation hex (1 Byte)
- No Operation hex (1 Byte)
- TCP Sack Permitted Option(2 Byte)

5. Erkenntnis

In dieser Übung wurde erkannt, wie man in einem Netzwerk Pakete „mitsniffen“ kann. Weiters wurden ein Ping- und ein TCP-Paket vollständig analysiert und die einzelnen Bits definiert. Uns wurde klar, wie wichtig Verschlüsselung ist, damit die gesendeten und empfangenen Daten nicht von Dritten gelesen werden können.