

ITM-Praktikum

Versuch 3: Dialogbasierte Protokolle

Andreas Klingler, Hannes Stahl, Simon Lüke

8. Juni 2009

Inhaltsverzeichnis

1	Vorbereitende Fragen	2
1.1	Verbindungsaufbau, Datenübertragung, Verbindungsabbau per TCP	2
1.2	Portnummern bei TCP und UDP	2
1.3	URL und URI	2
1.4	HTTP	2
1.5	Mehere Domains auf einem Webserver	2
1.6	TLS und Virtuelles Hosting	2
1.7	HTTP-Syntax für den Seitenabruf	3
1.8	SMTP	3
1.9	Zustellung einer Mail ohne eigene Adresse im TO-Feld	3
1.10	SMTP-after-POP	3
1.11	FTP	3
1.12	Portnummern	4
2	Versuchsdurchführung	4
2.1	Abrufen einer Homepage von Hand	4
2.2	Sniffen einer Browsersitzung	6
2.3	Sniffen einer SSH Sitzung	9
2.4	Simple Mail Transfer Protocol (SMTP)	10
2.5	Analyse einer FTP Sitzung	12

1 Vorbereitende Fragen

1.1 Verbindungsaufbau, Datenübertragung, Verbindungsabbau per TCP

- Verbindungsaufbau mit 3-Wege Handshake: Sender schickt Paket x mit syn=1, Empfänger acknowledged x+1, syn=1, ack=1, Sender acknowledged y+1, syn=1, ack=1
- Wenn die Verbindung steht, kann sowohl Sender/Empfänger gleichberechtigt Pakete senden
- Der Verbindungsabbau geschieht gleich wie der Aufbau, nur mit gesetztem FIN Bit. Allerdings müssen beide Teilnehmer ihre Verbindung mit dem FIN Bit beenden, Acknowledgt beispielsweise der Client nicht mit einem FIN Bit, bleibt die Verbindung in diese Richtung noch geöffnet.

1.2 Portnummern bei TCP und UDP

Ports werden verwendet, um mit dem gleichen Protokoll (z. B. TCP) unterschiedlichen Anwendungen Daten zukommen zu lassen.

1.3 URL und URI

Uniform Resource Identifier bzw. **Locator** (URI bzw -URL) dienen dazu eine Resource (im Netzwerk) zu bezeichnen und anzusprechen. Dabei wird grundsätzlich folgende Konvention verwendet: *Schema:durch das Schema def. Resource.*

URIs werden z.B. für FTP (*ftp ...*), HTTP (*http ...*), Emailadressen (*mailto ...*), Dateien (*file ...*), Telefonie (*tel ...* oder auch (*sip ...*) und vieles mehr verwendet. WWW-Adressen sind sicher die bewusstesten bzw. gebräuchlichsten URIs bzw. genauer URLs:

`http://www.meinedomain.de:8080/unterseite`

http gibt an, dass es sich um eine Resource handelt, die über das HTTP angesprochen werden kann und die Location dementsprechend aufzulösen ist. Also: über *Port 8080* auf dem **Server** **www.meinedomain.de** wird die *Resource unterseite* aufgerufen.

1.4 HTTP

HTTP kann beliebige Daten übertragen. Im Headerfeld „Content-Type“ kann ein MIME Datentyp angegeben werden.

1.5 Mehre Domains auf einem Webserver

Technisch können auf einem Server mehrerer virtuelle Hosts laufen (vhosts). HTTP muss dementsprechend im Header den Host oder die komplette URL mitschicken, damit der Server (z. B. Apache) den Request an den richtigen Host leiten können. Jeder Host kann weiterhin z. B. Port 80 für HTTP Anfragen verwenden trotz gleichem Rechner und gleicher IP.

1.6 TLS und Virtuelles Hosting

SSL und die ersten Versionen von TLS übermittelten beim Verbindungsaufbau keinen Hostnamen im Klartext. Somit war es nicht möglich unterschiedliche Zertifikate für verschiedene Hosts zu verwenden. Ab TLS 1.2 wird bereits beim Verbindungsaufbau der gewünschte Host übermittelt, so dass jeder Host sein eigenes Zertifikat verwenden kann.

1.7 HTTP-Syntax für den Seitenabruf

Für das einfache Abholen einer Website wird ein GET Befehl über HTTP gesendet:

```
GET /infotext.html HTTP/1.1
host: e-technik.uni-ulm.de
```

Desweiteren gibt es folgende Befehle:

- GET fordert eine Seite an. Es kann auch eine Query übergeben werden
- POST für Formulardaten/Datenübermittlung
- HEAD
- PUT Dateien auf den Server kopieren. Kaum noch benutzt.
- DELETE selten genutzte Möglichkeit um Dateien zu löschen.
- TRACE
- OPTIONS fordert vom Server eine Auflistung aller unterstützten Funktionen an
- CONNECT ?

1.8 SMTP

Das **Simple Mail Transfer Protocol (SMTP)** dient dazu Emails vom Sender(Mailclient) über die Mailserver zum für den vom Empfänger genutzten Mailserver zu leiten.

1.9 Zustellung einer Mail ohne eigene Adresse im TO-Feld

Der Eintrag im *TO*-Feld dient nicht der Verteilung der Email an den oder die Empfänger. Welches Postfach die Mail zugestellt bekommen soll, wird im *RCPT TO*-Aufruf bei der weitergabe an den Server definiert. Sind die Mailserver lasch genug konfiguriert, müssen *TO*-Header und *RCPT TO*-Aufruf nicht zusammenpassen.

Ausserdem kann es sein, dass die Mail über einen lokalen Alias auf dem Mailserver im eigenen Postfach abgelegt worden.

1.10 SMTP-after-POP

SMTP unterstützt keine Authentifizierung und so kann prinzipiell jeder ans Internet angeschlossene Rechner beliebige Mails aufgeben, was von Spam-Versender ausgenutzt werden kann. Will man nun nur authentifizierte Nutzer (z.B. die, die beim annehmenden Server wirklich angemeldet sind und ein Postfach haben) trotzdem mittels SMTP Mails über diesen Server versenden lassen, kann man die Authentifizierung des POP-Verfahrens mitnutzen: man setzt einfach voraus, dass ein Nutzer von einer IP-Adresse aus nur Mails versenden darf (per SMTP), nachdem er von dieser IP aus erfolgreich eine POP-Abfrage durchgeführt hat.

1.11 FTP

Aktives FTP Bei einer aktiven FTP-Verbindung schickt der Client von einem frei wählbaren Port > 1024 einen Request (*PORT*) an den Port 21 des Servers. Dieser bestätigt die Anfrage wieder über diesen Port 21 an den beim Request verwendeten Port des Clients, die Daten werden dann von Port 20 des Servers zum Request-Port des Clients geschickt. So können Client und Server auch weiterhin über Port 21 Kommunizieren, auch wenn die Datenübertragung schon läuft.

Passives FTP Im Passiven Modus stellt der Server, nachdem er vom Client um den Passiven-Modus (*PASV*) gebeten wurde, einen dynamischen Port (> 1024) über den Kontroll-Port 21 zur Verfügung und wartet danach, bis sich der Client auf diesen Port verbindet.

Wird NAT (Network Address Translation) verwendet, können bei FTP Probleme entstehen, da Client und Server beim Verbindungsaufbau IP-Adressen und gewünschte Portnummern austauschen, die Pakete mit diesen Informationen müssen also auch vom NAT-Router modifiziert werden.

ASCII- und Binary-Modus Im **ASCII-Modus** werden die zu übertragenden Daten mit Symbolen aus dem ASCII-Zeichensatz übertragen und entsprechen der Quell- und Zielplattform vom Client bzw. Server konvertiert, falls diese ein anderes Dateiformat verwenden. Im **Binary-Modus** werden die Dateien Byte für Byte übertragen ohne Berücksichtigung der verwendeten Kodierung.

1.12 Portnummern

- HTTP: 80
- SSH: 22
- SMTP: 25 oder 587 (für Einspeisen von Mail durch Clients)
- IMAP: 143; IMAP3: 220; IMAPS: 993
- POP3: 110; 995 (Verschlüsselt)
- FTP: 20 (Daten) und 21 (Kontrolle)

2 Versuchsdurchführung

2.1 Abrufen einer Homepage von Hand

Es wird in einer Telnetsitzung die Verbindung zu `www.uni-ulm.de` auf Port 80 hergestellt und mit HTTP/1.1 der Head abgerufen. Bei HTTP 1.1 kann auch der Host angegeben werden. Es folgt eine positive Antwort mit diversen Serverinformationen.

```
apfelsaft:~# telnet www.uni-ulm.de 80
Trying 134.60.1.25...
Connected to www.uni-ulm.de.
Escape character is '^]'.
HEAD /index.php HTTP/1.1
host: www.uni-ulm.de

HTTP/1.1 200 OK
Date: Wed, 27 May 2009 13:00:35 GMT
Server: Apache/2.2.11 (Unix) mod_ssl/2.2.11 OpenSSL/0.9.8i PHP/5.2.8
X-Powered-By: PHP/5.2.8
Set-Cookie: fe_typo_user=aa30f7fd7a2bbcc1f8fb4b0101456192; path=/
Content-Type: text/html; charset=utf-8
X-Pad: avoid browser bug

Connection closed by foreign host.
```

Nun wird versucht die Datei mittels HTTP/1.0 zu bekommen. Der Server liefert zurück, dass er gerne HTTP/1.1 haben möchte und die Datei nicht finden kann. Außerdem wird ein HTML Code generiert, dass der Fehler 404 im Browser des Client angezeigt wird:

```
apfelsaft:~# telnet www.uni-ulm.de 80
Trying 134.60.1.25...
Connected to www.uni-ulm.de.
Escape character is '^]'.
GET /index.php/ HTTP/1.0

HTTP/1.1 404 Not Found
Date: Wed, 27 May 2009 13:03:18 GMT
Server: Apache/2.2.11 (Unix) mod_ssl/2.2.11 OpenSSL/0.9.8i PHP/5.2.8
Content-Length: 331
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html>
.
.
</html>
Connection closed by foreign host.
```

Nun wird die HTML Datei mit HTTP/1.1 abgerufen. Der Quellcode wird zurückgeliefert:

```
apfelsaft:~# telnet www.uni-ulm.de 80
Trying 134.60.1.25...
Connected to www.uni-ulm.de.
Escape character is '^]'.
GET /index.php HTTP/1.1
host: www.uni-ulm.de

HTTP/1.1 200 OK
Date: Wed, 27 May 2009 13:04:02 GMT
Server: Apache/2.2.11 (Unix) mod_ssl/2.2.11 OpenSSL/0.9.8i PHP/5.2.8
X-Powered-By: PHP/5.2.8
Set-Cookie: fe_typo_user=3939a0566ece63a78aeecb92d9a52a67; path=/
Transfer-Encoding: chunked
Content-Type: text/html; charset=utf-8

8867
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html
    PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de" lang="de">
<body>
.
.
.
</body>
</html>
```

Connection closed by foreign host.

2.2 Sniffen einer Browsersitzung

Jetzt wird eine Seite per Browser abgerufen und der Netzwerkverkehr gesnift:

No. .	Time	Source	Destination	Protocol	Info
3	2.328782	10.3.1.2	10.0.0.99	DNS	Standard query A www.perdu.com
4	2.360711	10.3.1.2	10.0.0.99	DNS	Standard query AAAA www.perdu.com
5	2.481057	10.0.0.99	10.3.1.2	DNS	Standard query response A 82.165.176.145
6	2.493731	10.0.0.99	10.3.1.2	DNS	Standard query response
7	2.495158	10.3.1.2	10.0.0.99	DNS	Standard query AAAA www.perdu.com.itm.e-technik.uni-ulm.de
8	2.495856	10.0.0.99	10.3.1.2	DNS	Standard query response, No such name
9	2.496286	10.3.1.2	82.165.176.145	TCP	3930 > www [SYN, ACK] Seq=0 Len=0 MSS=1460 TSV=757448 TSER=0 WS=2
10	2.594396	82.165.176.145	10.3.1.2	TCP	www > 3930 [SYN, ACK] Seq=0 Len=0 MSS=1460 TSV=411589866 TSER=757448 WS=6
11	2.594568	10.3.1.2	82.165.176.145	TCP	3930 > www [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSV=757472 TSER=411589866
12	2.595970	10.3.1.2	82.165.176.145	HTTP	GET / HTTP/1.1
13	2.691861	82.165.176.145	10.3.1.2	TCP	www > 3930 [ACK] Seq=1 Ack=343 Win=6912 Len=0 TSV=411589964 TSER=757473
14	2.692921	82.165.176.145	10.3.1.2	HTTP	HTTP/1.1 200 OK (text/html)
15	2.692988	82.165.176.145	10.3.1.2	TCP	www > 3930 [FIN, ACK] Seq=459 Ack=343 Win=6912 Len=0 TSV=411589965 TSER=757473
16	2.693073	10.3.1.2	82.165.176.145	TCP	3930 > www [ACK] Seq=343 Ack=459 Win=6912 Len=0 TSV=757497 TSER=411589965
17	2.731589	10.3.1.2	82.165.176.145	TCP	3930 > www [ACK] Seq=343 Ack=460 Win=6912 Len=0 TSV=757507 TSER=411589965
18	4.998657	TyanComp_22:51:f3	3com_9c:0d:c4	ARP	Who has 10.3.1.1? Tell 10.3.1.100
19	4.998709	3com_9c:0d:c4	TyanComp_22:51:f3	ARP	10.3.1.1 is at 00:01:02:9c:0d:c4
20	5.165355	10.3.1.2	82.165.176.145	TCP	3930 > www [FIN, ACK] Seq=343 Ack=460 Win=6912 Len=0 TSV=758115 TSER=411589965
21	5.261281	82.165.176.145	10.3.1.2	TCP	www > 3930 [ACK] Seq=460 Ack=344 Win=6912 Len=0 TSV=411592534 TSER=758115

Abbildung 1: Sniffen einer Browsersitzung

Kurze Erklärung zum Screenshot (Bild 1):

- Die blauen Pakete sind die DNS Abfrage um die URL www.perdu.com aufzulösen.
- Paket 9, 10, 11 stellen die Verbindung her (Handshake), der Client hat Port 3930 gewählt.
- Paket 12 ist die eigentliche HTTP Anfrage vom Client, welche gleich noch detaillierter besprochen werden soll.
- Die angeforderten Daten vom Server stecken in Paket 14.
- Paket 16, 20 und 21 beenden die Verbindung.

Im Folgenden betrachten wir den HTTP Anteil von Paket 14 (GET Anfrage):

```

Hypertext Transfer Protocol
  GET / HTTP/1.1\r\n
User-Agent: Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.5 (like Gecko) (Debian)\r\n
Accept: text/html, image/jpeg, image/png, text/*, image/*, */*\r\n
Accept-Encoding: x-gzip, x-deflate, gzip, deflate\r\n
Accept-Charset: utf-8, utf-8;q=0.5, */q=0.5\r\n
Accept-Language: de, en\r\n
Host: www.perdu.com\r\n
Connection: Keep-Alive\r\n
\r\n

```

Abbildung 2: HTTP Get Anfrage

Man kann erkennen, dass der Browser über sich einige Informationen preisgibt, z. B:

- User-Agent: Browsertyp, Version, ...
- Accept: Welche Dateitypen akzeptiert der Browser
- Accept-Charset: Zeichenformat
- Connection: Keep-Alive, damit die Verbindung nicht für jede einzelne GET-Anfrage erneut aufgebaut werden muss.

Auf diese GET Anfrage, antwortet der Server in Paket 14 mit folgendem HTTP Inhalt:

```

Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
  Date: Wed, 27 May 2009 12:46:50 GMT\r\n
  Server: Apache/2.2.9 (Fedora)\r\n
  Last-Modified: Mon, 24 Apr 2006 07:45:49 GMT\r\n
  Etag: "c00ca59-cc-412286d532540"\r\n
  Accept-Ranges: bytes\r\n
  Content-Length: 204
  Connection: close\r\n
  Content-Type: text/html\r\n
  \r\n
Line-based text data: text/html
<html><head><title>Vous Etes Perdu ?</title></head><body><h1>Perdu sur l'Internet ?</h1><h2>Pas de panique, on va vous aider</h2><strong><pre> * <----- vous &eirc;

```

Abbildung 3: HTTP GET Antwort

Der Server übersendet einen Header mit diversen Informationen wie Servertyp, letzte Modifizierung der Seite, Datentyp und schließt die Verbindung gleich wieder. Ganz unten bei „Line-based text/date: text/html“ folgt dann der Quellcode der HTML Datei.

2.3 Sniffen einer SSH Sitzung

Im Folgenden wurde ein SSH Verbindung gestartet und mitgesniff:

No..	Time	Source	Destination	Protocol	Info
1	0.000000	10.3.1.1	10.0.0.99	DNS	Standard query AAAA schorle.itm.e-technik.uni-ulm.de
2	0.000785	10.0.0.99	10.3.1.1	DNS	Standard query response
3	0.024029	10.3.1.1	10.0.0.99	DNS	Standard query A schorle.itm.e-technik.uni-ulm.de
4	0.035031	10.0.0.99	10.3.1.1	DNS	Standard query response A 10.3.1.2
5	0.035393	10.3.1.1	10.3.1.2	TCP	50809 > ssh [SYN] Seq=0 Len=0 MSS=1460 TSV=126544 TSER=0 WS=7
6	0.035679	10.3.1.2	10.3.1.1	TCP	ssh > 50809 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 TSV=150025 TSER=126544 WS=2
7	0.035740	10.3.1.1	10.3.1.2	TCP	50809 > ssh [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSV=126544 TSER=150025
8	0.104715	10.3.1.2	10.3.1.1	SSH	Server Protocol: SSH-2.0-OpenSSH_4.3p2 Debian-9
9	0.104789	10.3.1.1	10.3.1.2	TCP	50809 > ssh [ACK] Seq=1 Ack=32 Win=5888 Len=0 TSV=126561 TSER=150042
10	0.105049	10.3.1.1	10.3.1.2	SSH	Client Protocol: SSH-2.0-OpenSSH_4.3p2 Debian-9
11	0.105292	10.3.1.2	10.3.1.1	TCP	ssh > 50809 [ACK] Seq=32 Ack=32 Win=5792 Len=0 TSV=150043 TSER=126561
12	0.106811	10.3.1.1	10.0.0.99	DNS	Standard query A schorle.itm.e-technik.uni-ulm.de
13	0.107601	10.0.0.99	10.3.1.1	DNS	Standard query response A 10.3.1.2
14	0.108233	10.3.1.1	10.0.0.99	DNS	Standard query PTR 2.3.10.in-addr.arpa
15	0.108773	10.0.0.99	10.3.1.1	DNS	Standard query response PTR schorle.itm.e-technik.uni-ulm.de
16	0.111008	10.3.1.1	10.0.0.99	DNS	Standard query A schorle.itm.e-technik.uni-ulm.de
17	0.111275	10.3.1.2	10.3.1.1	SSHv2	Server: Key Exchange Init
18	0.111889	10.0.0.99	10.3.1.1	DNS	Standard query response A 10.3.1.2
19	0.112256	10.3.1.1	10.0.0.99	DNS	Standard query PTR 2.1.3.10.in-addr.arpa
20	0.112906	10.0.0.99	10.3.1.1	DNS	Standard query response PTR schorle.itm.e-technik.uni-ulm.de
21	0.114452	10.3.1.1	10.3.1.2	SSHv2	Client: Key Exchange Init
22	0.152780	10.3.1.2	10.3.1.1	TCP	ssh > 50809 [ACK] Seq=736 Ack=744 Win=7216 Len=0 TSV=150055 TSER=126563
23	0.152827	10.3.1.1	10.3.1.2	SSHv2	Client: Diffie-Hellman key agreement
24	0.153066	10.3.1.2	10.3.1.1	TCP	ssh > 50809 [ACK] Seq=736 Ack=768 Win=7216 Len=0 TSV=150055 TSER=126573

Abbildung 4: Sniffen einer SSH-Sitzung

Bei einer SSH Sitzung gibt es nicht viel zu sehen:

- Die Verbindung wird normal als TCP Verbindung an Port 22 aufgebaut.
- In Paket 17 schickt der Server seinen öffentlichen Schlüssel
- In Paket 21 schickt der Client seinen öffentlichen Schlüssel, ab nun ist die Verbindung verschlüsselt und somit nicht mehr nachvollziehbar.
- Am Ende wird die TCP Verbindung normal abgebaut.

Nun sniffen wir im Vergleich dazu eine Telnetverbindung:

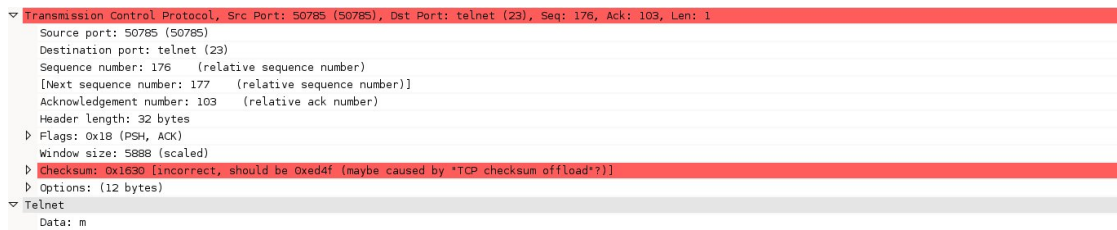


Abbildung 5: Sniffen einer Telnetsitzung

Die letzte Zeile im Screenshot (Bild 5) zeigt den Inhalt eines Telnetpaketes. Unter „Data:“ ist nur ein „m“ zu sehen. Dieser Buchstabe stammt vom Passwort (letzter Buchstabe von „praktikum“). Eine Telnetsitzung ist also eine sehr unsichere Sache.

2.4 Simple Mail Transfer Protocol (SMTP)

Wir verbinden uns per Telnet mit dem Mailserver der Uni und versenden eine Email mit gefälschter „From:“ und „To:“ im Datenfeld:

```
telnet mail.uni-ulm.de 25
Trying 134.60.1.11...
Connected to mail.uni-ulm.de.
Escape character is '^]'.
220 mail.uni-ulm.de ESMTP Sendmail 8.14.2/8.14.2; Wed, 27 May 2009 14:11:03
+0200 (MEST)
HELO apfelsaft.itm.e-technik.uni-ulm.de
250 mail.uni-ulm.de Hello videomax.e-technik.uni-ulm.de [134.60.30.95],
pleased to meet you
MAIL FROM: simon.lueke@uni-ulm.de
250 2.1.0 simon.lueke@uni-ulm.de... Sender ok
RCPT TO: simon.lueke@uni-ulm.de
250 2.1.5 simon.lueke@uni-ulm.de... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
Date: 03.05.1999
From: bill.gates@microsoft.com
To: steve.jobs@apple.jp.co
Subject: Test
Hallo, wie gehts?
Wie laufen die Gesch fte?
.
250 2.0.0 n4RCB3Lq018475 Message accepted for delivery
QUIT
221 2.0.0 mail.uni-ulm.de closing connection
Connection closed by foreign host.
```

Die Mail kommt trotz falscher Angaben im DATA Feld an, die Empfängeradresse im Header war gültig. Der Header der empfangenen Mail sieht so aus:

```
From - Wed May 27 14:15:46 2009
X-Mozilla-Status: 0001
```

```

X-Mozilla-Status2: 00000000
Return-Path: <simon.lueke@uni-ulm.de>
Received: from mail.uni-ulm.de ([unix socket])
        by poseidon (Cyrus v2.3.11) with LMTPA;
        Wed, 27 May 2009 14:15:36 +0200
X-Sieve: CMU Sieve 2.3
Received: from apfelsaft.itm.e-technik.uni-ulm.de
        (videomax.e-technik.uni-ulm.de [134.60.30.95])
        by mail.uni-ulm.de (8.14.2/8.14.2) with SMTP id n4RCB3Lq018475
        for simon.lueke@uni-ulm.de; Wed, 27 May 2009 14:13:17 +0200 (MEST)
Message-Id: <200905271213.n4RCB3Lq018475@mail.uni-ulm.de>
Date: 03.05.1999
From: bill.gates@microsoft.com
To: steve.jobs@apple.jp.co
Subject: Test
X-DCC-CTc-dcc2-Metrics: poseidon 1031; Body=1 Fuz1=1
X-Virus-Scanned: by amavisd-new

Hallo, wie gehts?
Wie laufen die Gesch fte?

```

Im Header kann man erkennen, dass die Absender und Empfänger gefälscht waren. Im Mailclient (Thunderbird) erscheint auch nur der gefälschte Absender, was User natürlich verwirren kann. Nebenbei sei erwähnt, dass unser angegebenes Datum falsches Format hatte. Der Mailclient konnte es nicht richtig darstellen.

Daraufhin haben wir noch den Absender sowohl im Header als auch im Datenfeld auf einen garantiert ungültigen Namen gefälscht, außerdem wurde auch der Domainname des Clients falsch eingegeben. Daraufhin kam folgende Mail bei mir an:

```

From - Wed May 27 14:23:50 2009
X-Account-Key: account3
X-UIDL: d6e89a6f03204348cabd70e40b353fa3
X-Mozilla-Status: 0001
X-Mozilla-Status2: 00000000
X-Mozilla-Keys:
X-Envelope-From: <freaky.baby@uni-ulm.de>
X-Envelope-To: <andy@elektro-klingler.de>
X-Delivery-Time: 1243427006
X-UID: 15781
Return-Path: <freaky.baby@uni-ulm.de>
X-RZG-CLASS-ID: mi
Received: from mail.uni-ulm.de ([134.60.1.11])
        by mailin.webmailer.de (hamish mi8) (RZmta 18.37)
        with ESMTP id r01a8el4RCBRj2 for <andy@elektro-klingler.de>;
        Wed, 27 May 2009 14:23:26 +0200 (MEST)
Received: from freak.itm.e-technik.uni-ulm.de
        (videomax.e-technik.uni-ulm.de [134.60.30.95])
        by mail.uni-ulm.de (8.14.2/8.14.2) with SMTP id n4RCJXjd006402
        for andreas.klingler@uni-ulm.de; Wed, 27 May 2009 14:21:11 +0200 (MEST)
Message-Id: <200905271221.n4RCJXjd006402@mail.uni-ulm.de>
Date: 26 Oct 2006

```

```

From: freak@etechiker.de
To: hannes@stahl.de
Subject: Spammer
X-DCC-CTc-dcc2-Metrics: poseidon 1031; Body=1 Fuzl=1
X-Virus-Scanned: by amavisd-new

```

Hoi,
im ITM Praktikum wird gespammt....
Greetz

Die einzige von uns richtig eingegebene Angabe war der Empfänger „andreas.klingler@uni-ulm.de“. Jetzt ist nur noch am Servernamen erkennbar woher die Mail kam. Außerdem taucht die IP des NAT Routers auf, wodurch der Spammer wohl im Netz des ITM Praktikums zu suchen ist ;-). (Nebenbemerkung: In diesem Datumsformat konnte Thunderbird das Datum richtig anzeigen).

2.5 Analyse einer FTP Sitzung

Nun betrachten wir den Verlauf einer FTP Sitzung in Wireshark.

No.	Time	Source	Destination	Protocol	Info
8	10.199677	10.0.0.99	10.3.1.1	DNS	Standard query response CNAME ftp.rz.uni-ulm.de A 134.60.1.5
9	10.233955	10.3.1.1	134.60.1.5	TCP	43725 > ftp [SYN] Seq=0 Len=0 MSS=1460 TSV=639116 TSER=0 WS=7
10	10.234611	134.60.1.5	10.3.1.1	TCP	ftp > 43725 [SYN, ACK] Seq=0 Ack=1 Win=49232 Len=0 TSV=283860071 TSER=639116 MSS=1460 WS=0
11	10.234681	10.3.1.1	134.60.1.5	TCP	43725 > ftp [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSV=639116 TSER=283860071
12	10.244096	134.60.1.5	10.3.1.1	FTP	Response: 220 (vsFTPd 2.1.0)
13	10.244205	10.3.1.1	134.60.1.5	TCP	43725 > ftp [ACK] Seq=1 Ack=21 Win=5888 Len=0 TSV=639119 TSER=283860072
14	12.935170	10.3.1.1	134.60.1.5	FTP	Request: USER anonymous
15	12.935774	134.60.1.5	10.3.1.1	TCP	ftp > 43725 [ACK] Seq=21 Ack=17 Win=49232 Len=0 TSV=283860341 TSER=639791
16	12.936037	134.60.1.5	10.3.1.1	FTP	Response: 331 Please specify the password.
17	12.936500	10.3.1.1	134.60.1.5	TCP	43725 > ftp [ACK] Seq=17 Ack=55 Win=5888 Len=0 TSV=639792 TSER=283860341
18	21.454028	10.3.1.1	134.60.1.5	FTP	Request: PASS andreas.klingler@uni-ulm.de
19	21.454698	134.60.1.5	10.3.1.1	TCP	ftp > 43725 [ACK] Seq=55 Ack=51 Win=49232 Len=0 TSV=283861193 TSER=641921
20	21.476365	134.60.1.5	10.3.1.1	FTP	Response: 230 Login successful.
21	21.476430	10.3.1.1	134.60.1.5	TCP	43725 > ftp [ACK] Seq=51 Ack=78 Win=5888 Len=0 TSV=641927 TSER=283861196
22	21.492829	10.3.1.1	134.60.1.5	FTP	Request: SYST
23	21.493426	134.60.1.5	10.3.1.1	FTP	Response: 215 UNIX Type: L8
24	21.532359	10.3.1.1	134.60.1.5	TCP	43725 > ftp [ACK] Seq=57 Ack=97 Win=5888 Len=0 TSV=641941 TSER=283861197
25	28.308367	TyanComp_22:51:f3	Broadcast	ARP	Who has 10.4.1.4? Tell 10.4.1.100
26	28.313203	Siemens_20:ca:ac	Broadcast	ARP	Gratuitous ARP for 10.4.1.4 (Reply)
27	31.593059	10.3.1.1	134.60.1.5	FTP	Request: CWD /usr/lib64
28	31.591345	134.60.1.5	10.3.1.1	FTP	Response: 550 Failed to change directory.
29	31.591427	10.3.1.1	134.60.1.5	TCP	43725 > ftp [ACK] Seq=75 Ack=130 Win=5888 Len=0 TSV=644555 TSER=283862247
30	33.584733	10.3.1.1	134.60.1.5	FTP	Request: PASV
31	33.585889	134.60.1.5	10.3.1.1	FTP	Response: 227 Entering Passive Mode (134,60,1,5,212,248).
32	33.585967	10.3.1.1	134.60.1.5	TCP	43725 > ftp [ACK] Seq=81 Ack=179 Win=5888 Len=0 TSV=644954 TSER=283862407
33	33.586135	10.3.1.1	134.60.1.5	TCP	47993 > 54520 [SYN] Seq=0 Len=0 MSS=1460 TSV=644954 TSER=0 WS=7
34	33.587193	134.60.1.5	10.3.1.1	TCP	54520 > 47993 [SYN, ACK] Seq=0 Ack=1 Win=49232 Len=0 TSV=283862407 TSER=644954 MSS=1460 WS=0
35	33.587248	10.3.1.1	134.60.1.5	TCP	47993 > 54520 [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSV=644954 TSER=283862407
36	33.587332	10.3.1.1	134.60.1.5	FTP	Request: LIST
37	33.588279	134.60.1.5	10.3.1.1	FTP	Response: 150 Here comes the directory listing.
38	33.588753	134.60.1.5	10.3.1.1	FTP-DATA	FTP Data: 126 bytes
39	33.588821	134.60.1.5	10.3.1.1	TCP	54520 > 47993 [FIN, ACK] Seq=127 Ack=1 Win=49232 Len=0 TSV=283862407 TSER=644954
40	33.588946	10.3.1.1	134.60.1.5	TCP	47993 > 54520 [ACK] Seq=1 Ack=127 Win=5888 Len=0 TSV=644955 TSER=283862407
41	33.589123	10.3.1.1	134.60.1.5	TCP	47993 > 54520 [FIN, ACK] Seq=1 Ack=128 Win=5888 Len=0 TSV=644955 TSER=283862407
42	33.589540	134.60.1.5	10.3.1.1	TCP	54520 > 47993 [ACK] Seq=128 Ack=2 Win=49232 Len=0 TSV=283862407 TSER=644955

Transmission Control Protocol, Src Port: 43725 (43725), Dst Port: ftp (21), Seq: 17, Ack: 55, Len: 34

Source port: 43725 (43725)

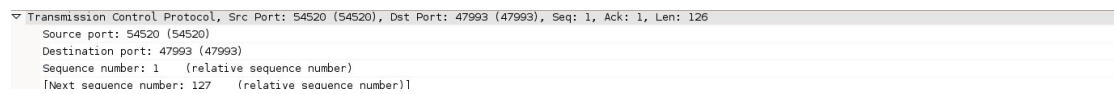
Destination port: ftp (21)

Sequence number: 17 (relative sequence number)

Next sequence number: 51 (relative sequence number)

Abbildung 6: Sniffen einer FTP Sitzung

Nach dem obligatorischen Handshake wird mit Paket 14 der Username übertragen. Paket 18 übermittelt das Passwort im Klartext. Im Screenshot (Bild 6) unten sieht man die Portnummern, über welche die Steuerverbindung läuft. Dies ist beim Client der Port 43725 und beim Server Port 21. Diese Ports bleiben über die ganze Session gleich. Paket 38 ist nun das erste Datenpaket, welches ein Directory-Listing enthält. Dieses Paket sieht so aus:



Transmission Control Protocol, Src Port: 54520 (54520), Dst Port: 47993 (47993), Seq: 1, Ack: 1, Len: 126
Source port: 54520 (54520)
Destination port: 47993 (47993)
Sequence number: 1 (relative sequence number)
[Next sequence number: 127 (relative sequence number)]

Abbildung 7: Sniffen einer Telnetsitzung

Der Server sendet nun von Port 54520 an den Zielport 47993 beim Client. Diese Portkombination bleibt nicht gleich, sondern wird bei jeder Datenübertragung neu gewählt.