Ethereal Intro && Ethereal HTTP

René Hansen
Anders Bjerg Pedersen
Michael Nilou
Hold 1

September 12, 2007

Indledning

Denne opgave går ud på at anvende programmet Wireshark til at analysere trafik over netværk, med specielt henblik på HTTP-protokollen. Første del er en tutorial i brug af programmet; anden del er en grundigere undersøgelse af

Ethereal INTRO

1: Opsnappede protokoller

Følgende protokoller blev opsnappet ved test-sniffningen:

- AIM: AOL Instant Messenger
- ARP: Address Resolution Protocol
- BROWSER: Microsoft Windows Browser Protocol
- DHCP: Dynamic Host Configuration Protocol
- DNS: Domain Name Service
- HTTP: Hyper Text Transport Protocol
- IP RIP: ???
- MDNS: ???
- MSNMS: MSN Messenger Service
- NBNS: NetBIOS Name Service
- STP: Spanning Tree Protocol
- TCP: Transmission Control Protocol

2: HTTP Delay

Følgende starttider blev målt ved kontakt til www.diku.dk, som blev brugt i stedet for den i opgaven angivne adresse, der ikke længere er tilgængelig (skærmdumps og pakkeindhold kan ses længere nede):

- HTTP request sendt til tiden: 7.887333.
- HTTP OK modtaget til tiden: 7.913923.
- Latency: 0,026593

3: IP-adresser

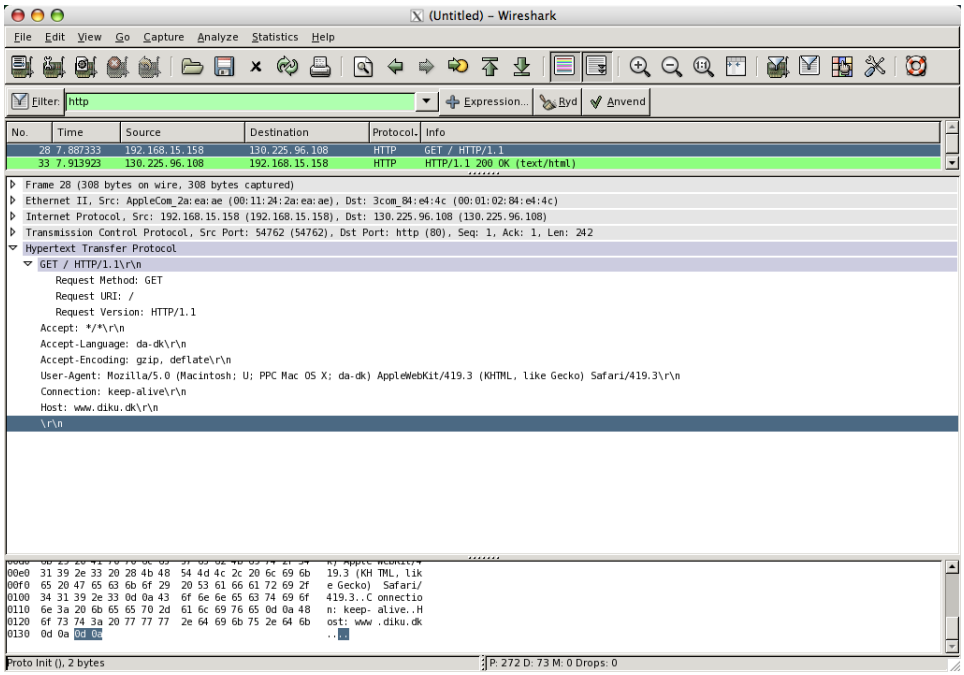
Den i opgaven angivne fil var ikke tilgængelig, så vi har kontaktet www.diku.dk i stedet. IP-adresserne aflæses i linien begyndende med "Internet Protocol".

- IP-adresse for www.diku.dk: 130.225.96.108.
- Lokal IP-adresse: 192.168.15.158.

4: Packet prints og skærmdumps

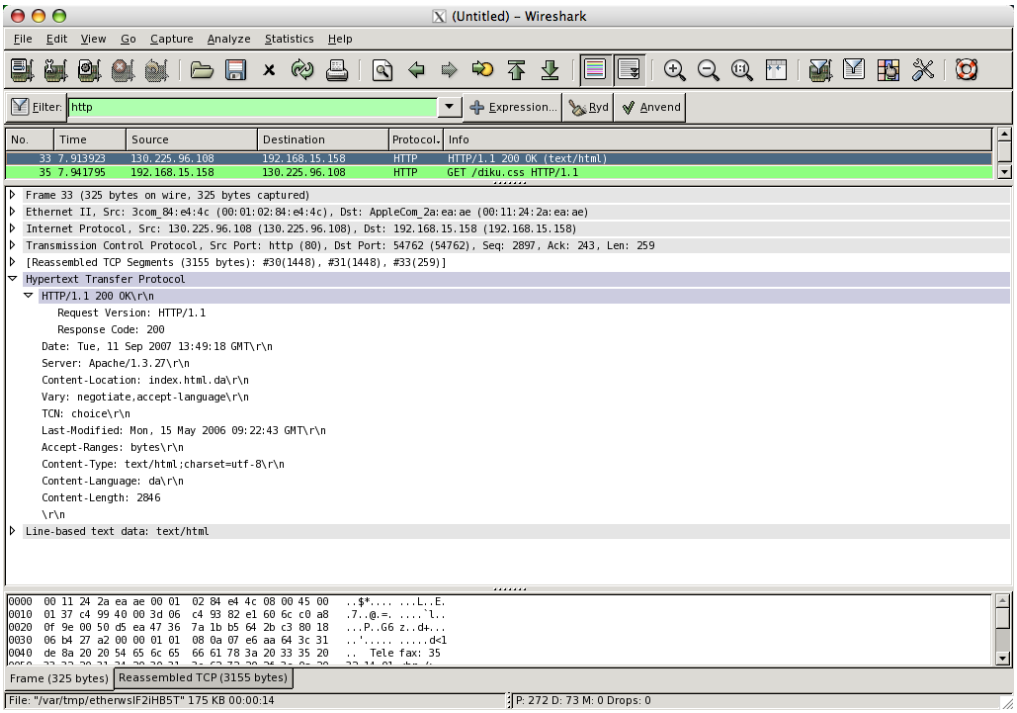
- Packet HTTP GET:

```
 No.     Time        Source                Destination           Protocol Info
      28 7.887333    192.168.15.158        130.225.96.108        HTTP     GET / HTTP/1.1
Frame 28 (308 bytes on wire, 308 bytes captured)
Ethernet II, Src: AppleCom_2a:ea:ae (00:11:24:2a:ea:ae), Dst: 3com_84:e4:4c (00:01:02:84:e4:4c)
Internet Protocol, Src: 192.168.15.158 (192.168.15.158), Dst: 130.225.96.108 (130.225.96.108)
Transmission Control Protocol, Src Port: 54762 (54762), Dst Port: http (80), Seq: 1, Ack: 1, Len: 242
Hypertext Transfer Protocol
  GET / HTTP/1.1\r\n
    Request Method: GET
    Request URI: /
    Request Version: HTTP/1.1
  Accept: */*\r\n
  Accept-Language: da-dk\r\n
  Accept-Encoding: gzip, deflate\r\n
  User-Agent: Mozilla/5.0 (Macintosh; U; PPC Mac OS X; da-dk) AppleWebKit/419.3 (KHTML, lik
  e Gecko) Safari/419.3\r\n
  Connection: keep-alive\r\n
  Host: www.diku.dk\r\n
  \r\n
```



- Packet HTTP OK

```
No.     Time        Source                Destination           Protocol Info
     33 7.913923    130.225.96.108        192.168.15.158        HTTP     HTTP/1.1 200 OK
(text/html)
Frame 33 (325 bytes on wire, 325 bytes captured)
Ethernet II, Src: 3com_84:e4:4c (00:01:02:84:e4:4c), Dst: AppleCom_2a:ea:ae (00:11:24:2a:
ea:ae)
Internet Protocol, Src: 130.225.96.108 (130.225.96.108), Dst: 192.168.15.158 (192.168.15.
158)
Transmission Control Protocol, Src Port: http (80), Dst Port: 54762 (54762), Seq: 2897, A
ck: 243, Len: 259
[Reassembled TCP Segments (3155 bytes): #30(1448), #31(1448), #33(259)]
Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
    Request Version: HTTP/1.1
    Response Code: 200
  Date: Tue, 11 Sep 2007 13:49:18 GMT\r\n
  Server: Apache/1.3.27\r\n
  Content-Location: index.html.da\r\n
  Vary: negotiate,accept-language\r\n
  TCN: choice\r\n
  Last-Modified: Mon, 15 May 2006 09:22:43 GMT\r\n
  Accept-Ranges: bytes\r\n
  Content-Type: text/html;charset=utf-8\r\n
  Content-Language: da\r\n
  Content-Length: 2846
  \r\n
Line-based text data: text/html
```



Ethereal Lab: HTTP

1. The Basic HTTP GET/response interaction

Vi starter med packet dumps af vores HTTP GET- og HTTP OK-pakker.

HTTP GET:

```
No.     Time        Source                Destination           Protocol Info
     40 9.030222    192.168.15.158        128.119.245.12        HTTP     GET /ethereal-la
bs/HTTP-ethereal-file1.html HTTP/1.1
Frame 40 (352 bytes on wire, 352 bytes captured)
Ethernet II, Src: AppleCom_2a:ea:ae (00:11:24:2a:ea:ae), Dst: 3com_84:e4:4c (00:01:02:84:e4:4c)
Internet Protocol, Src: 192.168.15.158 (192.168.15.158), Dst: 128.119.245.12 (128.119.245.12)
Transmission Control Protocol, Src Port: 54939 (54939), Dst Port: http (80), Seq: 1, Ack: 1, Len: 286
Hypertext Transfer Protocol
  GET /ethereal-labs/HTTP-ethereal-file1.html HTTP/1.1\r\n
    Request Method: GET
    Request URI: /ethereal-labs/HTTP-ethereal-file1.html
    Request Version: HTTP/1.1
  Accept: */*\r\n
  Accept-Language: da-dk\r\n
  Accept-Encoding: gzip, deflate\r\n
  User-Agent: Mozilla/5.0 (Macintosh; U; PPC Mac OS X; da-dk) AppleWebKit/419.3 (KHTML, lik
  e Gecko) Safari/419.3\r\n
  Connection: keep-alive\r\n
  Host: gaia.cs.umass.edu\r\n
  \r\n
```

HTTP OK:

```
No.     Time        Source                Destination           Protocol Info
     42 9.141306    128.119.245.12        192.168.15.158        HTTP     HTTP/1.1 200 OK
(text/html)
Frame 42 (499 bytes on wire, 499 bytes captured)
Ethernet II, Src: 3com_84:e4:4c (00:01:02:84:e4:4c), Dst: AppleCom_2a:ea:ae (00:11:24:2a:ea:ae)
Internet Protocol, Src: 128.119.245.12 (128.119.245.12), Dst: 192.168.15.158 (192.168.15.158)
Transmission Control Protocol, Src Port: http (80), Dst Port: 54939 (54939), Seq: 1, Ack: 287, Len: 433
Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
    Request Version: HTTP/1.1
    Response Code: 200
  Date: Tue, 11 Sep 2007 13:00:13 GMT\r\n
  Server: Apache/2.0.52 (CentOS)\r\n
  Last-Modified: Tue, 11 Sep 2007 13:00:02 GMT\r\n
  ETag: "126453-7e-aee97880"\r\n
  Accept-Ranges: bytes\r\n
  Content-Length: 126
  Keep-Alive: timeout=10, max=100\r\n
  Connection: Keep-Alive\r\n
  Content-Type: text/html; charset=ISO-8859-1\r\n
  \r\n
Line-based text data: text/html
```

Q1: **Request:**
Af følgende linie i vore HTTP GET ses det, at vores browser kører HTTP 1.1:
`GET /ethereal-labs/HTTP-ethereal-file1.html HTTP/1.1`
Response:
Af følgende linie i vore HTTP reply ses det, at vores browser kører HTTP 1.1:
`HTTP/1.1 200 OK (text/html)`

Q2: Følgende linie i vores HTTP GET-pakke siger, at browseren foretrækker siden på dansk: `Accept-Language: da-dk`

Q3: Følgende linie i vores HTTP-GET-pakke fortæller, at vores egen IP-adresse (Src:) er 192.168.15.158, og at modtagerens IP-adresse (Dst:) er 128.119.245.12.
`Internet Protocol, Src: 192.168.15.158 (192.168.15.158),`
`Dst: 128.119.245.12 (128.119.245.12)`

Q4: Serveren returnerer `HTTP/1.1 200 OK.`

Q5: Serveren fortæller os i en HTTP-header:
`Last-Modified: Tue, 11 Sep 2007 13:00:02 GMT.`

Q6: Der returneres 126 bytes til os, som vi kan se fra følgende headers:

```
Accept-Ranges: bytes
Content-Length: 126
```

Q7: Vi kunne ikke umiddelbart finde flere headers, der ikke blev listet i Wireshark.

2. The HTTP CONDITIONAL GET/response interaction

Q8: Nej, denne header findes ikke i den første HTTP GET-pakke.

Q9: Serveren *har* returneret indholdet af filen; det befinder sig i Wireshark nederst i pakken, under "`Line based text data: text/html`".

Q10: Ja, headeren "`IF-MODIFIED-SINCE:`" er sat til "`Tue, 11 Sep 2007 14:05:02 GMT`".

Q11: Vi får `HTTP/1.1 304 Not modified` retur. Serveren svarer altså, at indholdet ikke har ændret sig siden sidst og returnerer derfor ikke noget indhold.

3. Retrieving Long Documents

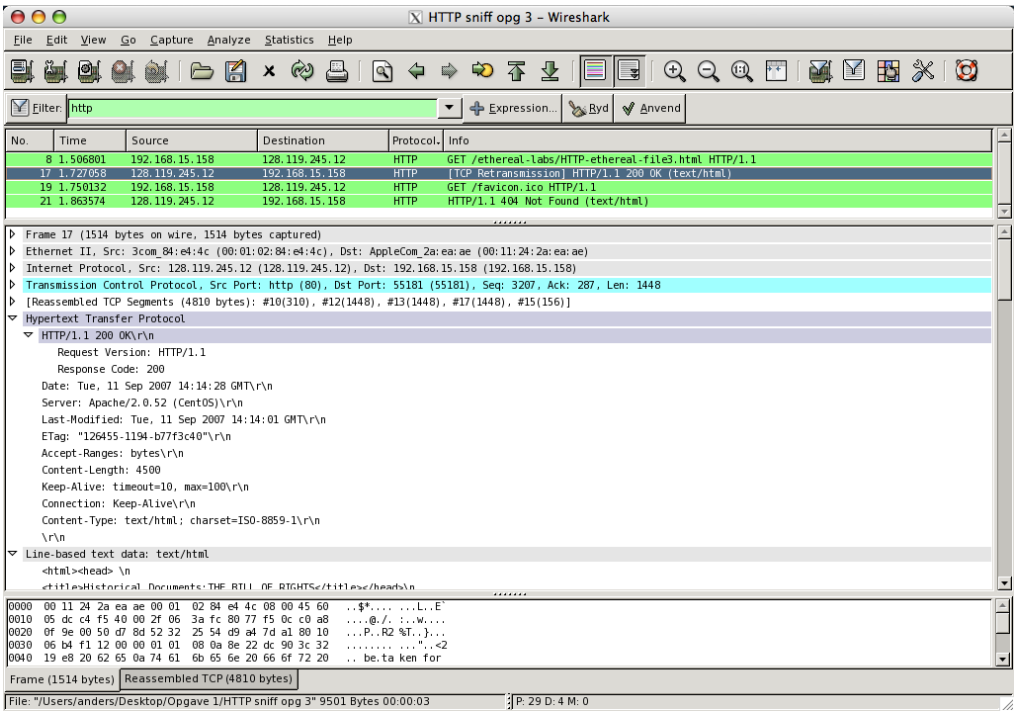
Screendump af relevante billeder fra Wireshark findes efter spørgsmålene.

Q12: Browseren sender kun én HTTP GET-request.

Q13: I Wireshark kan vi se, at der under TCP-laget er listet:
`[Reassembled TCP Segments (4810 bytes): #10(310), #12(1448),... ]`.
Altså har TCP automatisk delt filen op i 5 segmenter (pakker), der er blevet sendt hver for sig. HTTP-protokollen får ikke besked om dette.

Q14: Serveren returnerer `HTTP/1.1 200 OK` efter endt overførsel af alle 5 pakker.

Q15: Som der også står i opgaveteksten: "We stress here that there is no "Continuation" message in HTTP!" Altså er HTTP ikke klar over, hvorvidt data bliver sendt i brudstykker eller ej.

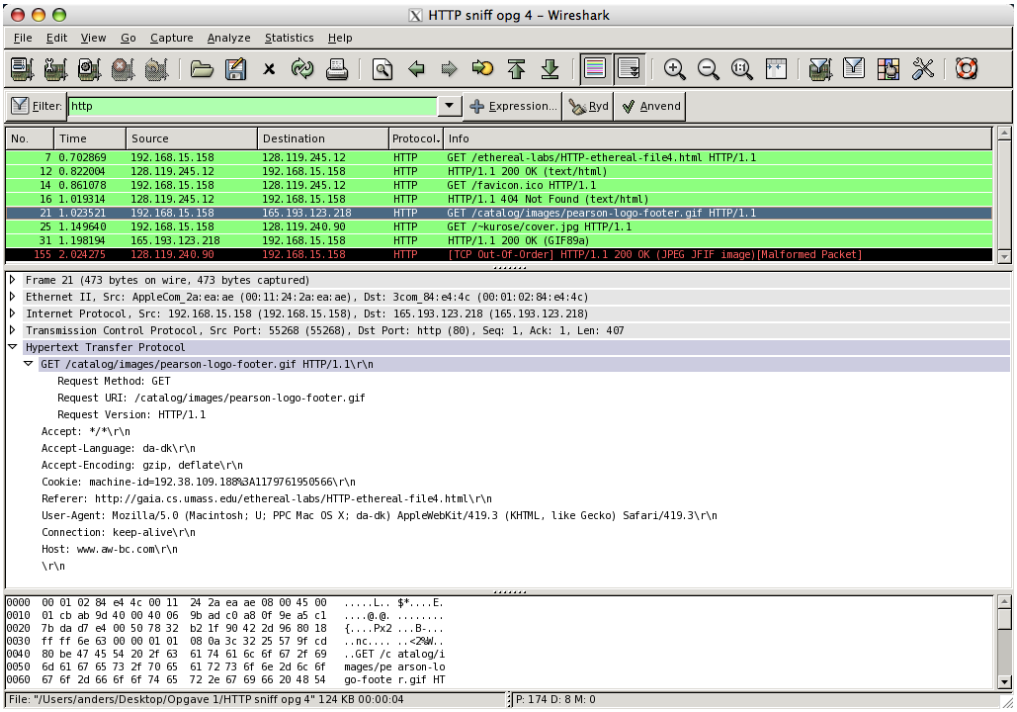


4. HTML Documents with Embedded Objects

Screendump af relevante billeder fra Wireshark findes efter spørgsmålene.

Q16: Browseren sender 3 GET-requests: en til 128.119.145.12 (HTML-filen), en til 165.193.123.218 (første billede) og en til 128.119.240.90 (andet billede).

Q17: Download af billederne foregår parallelt: to forskellige GETs sendes på samme tid, uden at vente på, at den første får response tilbage fra serveren (se evt. screenshot).

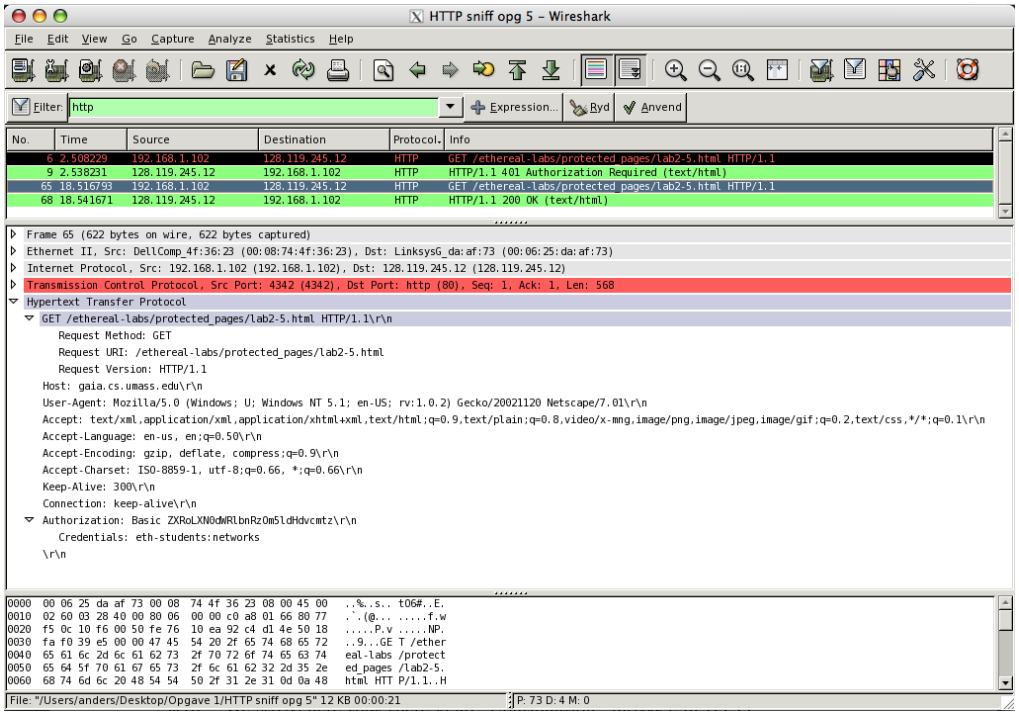


2.5 HTTP Authentication

Vi bruger i denne opgave de udleverede traces fra hjemmesiden. Screendump af relevante billeder fra Wireshark findes efter spørgsmålene.

Q18: Serveren returnerer: `HTTP/1.1 401 Authorization Required`.

Q19: Feltet "Authorization: Basic" er nu tilstede i den anden GET-request til serveren. I denne header er inkluderet vores brugernavn og password, krypteret med Base64-formatet (som er særdeles let at bryde!).



Konklusion

Vi har i denne opgave set, hvordan Wireshark kan bruges som packet sniffer til at opsnappe og afkode ukrypterede pakker på netværk, analysere dem og monitorere brugeres adfærd på nettet. Vi har ligeledes fået et indblik i, hvordan HTTP-protokollen kommunikerer med en host.