Åsa Pehrsson – asap@kth.se

1 Abstract

This report is written as part of the assignment given in the course 2G1305 Internetworking. The aim of the assignment is by practical experience gain deeper knowledge within a chosen topic. The topic is investigate the TLS session resumption impact of HTTP performance.

2 Introduction

Secured Socket Layer (SSL) and Transport Layer Security (TLS) are protocol used in the Internet for securing communications. In this paper, we analyze the performance impact of HTTP terms of TLS session resumption.

We begin with an overview of the HTTP and SSL/TLS protocols, then we present the findings from our tests, and finish with some conclusions.

3 Transport Layer Security (TLS)

HTTP messages are in plain text, the increased use of HTTP for sensitive applications has required security measures. SSL, and its successor TLS were designed to provide privacy over the Internet. They act as a separate protocol, inserted between HTTP and TCP.

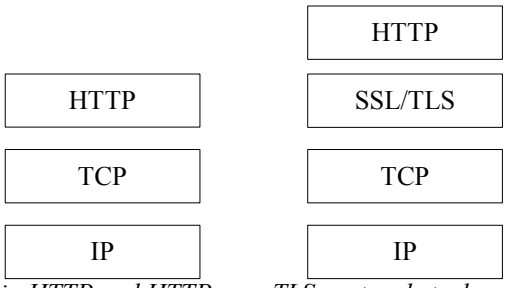

Figure 1. Plain HTTP and HTTP over TLS protocol stacks

A TLS session is an association between a client and a server, and consists of two phases: the initial handshake (using the handshake protocol) and the data transfer (using the application data protocol). During the handshake phase, the client and server use a public-key encryption algorithm to determine secret-key parameters. During the data transfer phase, both sides use the secret key to encrypt and decrypt successive data transmissions. At any time during the transmission, both client and server can change ciphering strategies (by using the change cipher spec protocol). The next figure shows the message flow for the full TLS handshake.

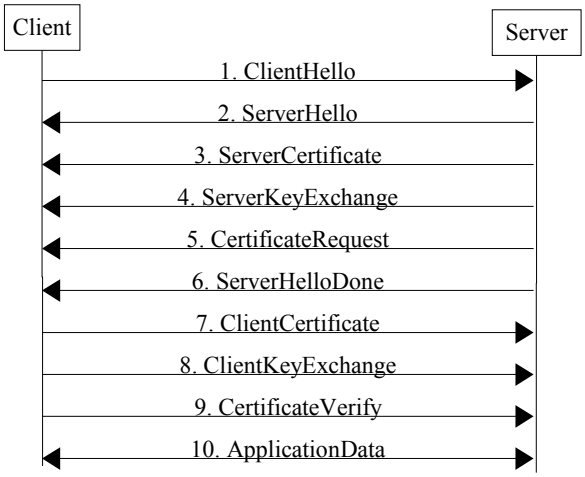

Figure 2. Message Flow in a Full Handshake

The handshake protocol involves the following steps (the numbers within parenthesis corresponds to a message in the figure above):

1. Exchange hello messages to agree on algorithms, exchange random values, and check for session resumption. When a client first connects to a server it must send the ClientHello (1) with an cipher suite list, a random number and optional session identifier. The server will respond with a ServerHello(2) with the selected cipher suite list, a random number and session identifier that is equal to the clients if the server accepts session resume.
2. Exchange the necessary cryptographic parameters to allow the client and server to agree on a pre-master secret. If the server certificate message did not contain information to build a pre-master secret, the ServerKeyExchange (4) message is sent. The ClientKeyExchange (8) message, carrying the pre-master secret, is always sent.
3. Exchange certificates (3, 5, 7) and cryptographic information. If the agreed-upon key exchange method is not an anonymous one, the server must send the ServerCertificaten (3) message. The server have the option to authenticate the user by sending the CerificateRequest (5) message. If client authentication is not used, the CerificateRequest (5), Certificate (7) and CertificateVerify (9) messages are omitted.
4. The server finishes its contribution to the handshake phase by sending the ServerHelloDone message.

The ability to resume a sessions is very useful, the time consuming negotiation of security parameters can be omitted. A resumed session starts with the “abbreviated handshake”. See figure below (messages marked with ** belongs to the change cipher spec protocol).

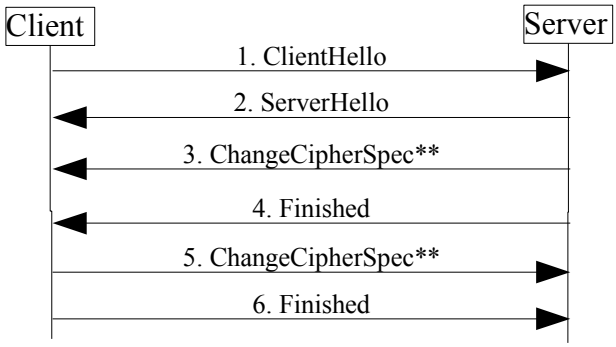

Figure 3. Message flow for an abbreviated handshake

1 Hypertext Transfer Protocol (HTTP)

HTTP is a stateless application-level protocol. HTTP operates over TCP, the default port is 80. There are two versions defined: 1.0 [2] and 1.1[3]. The protocol is based on a request/response paradigm. A client

establishes a connection with a server and sends a request to the server in the form of a protocol version, request method and URI. The server responds with a status line, followed by a MIME-like message containing server information, entity meta information, and possible body content.

Under version 1.0, the browser opens a new TCP connection to a server for every URL it fetches. If an HTML document contains three unique in line graphics or pages (as the case when using frames), the browser will open a total of four TCP connections: one for each image file, and one for the HTML document itself. Each new TCP connection incurs a series of delays: the initial TCP handshake when the connection is opened and another handshake when closing the connection. In addition, there's a slowdown at the beginning of each connection as the TCP/IP protocol adjusts its transmission speed to match the available bandwidth (“slow start”).

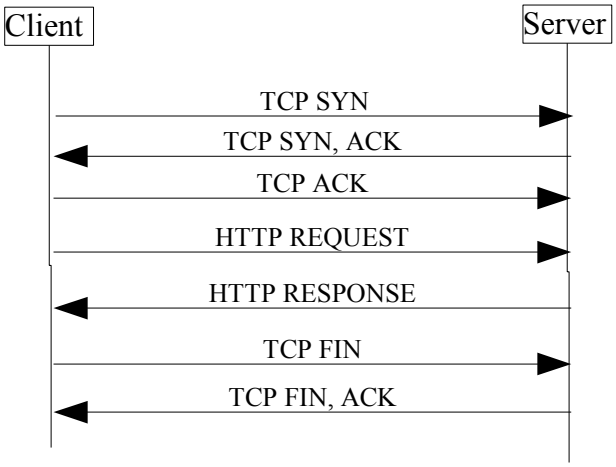

Figure 4. HTTP 1.0

Version 1.1 addresses the TCP overhead issue by keeping the initial connection open for subsequent requests. There is no need to open more that one TCP connections since HTTP 1.1 allows several URL requests to be piggybacked on top of a single TCP session. When the browser first contacts the server and downloads an HTML document, the connection remains open. The browser can then use the connection to request additional documents from the server; for example, the URL for each in line image.

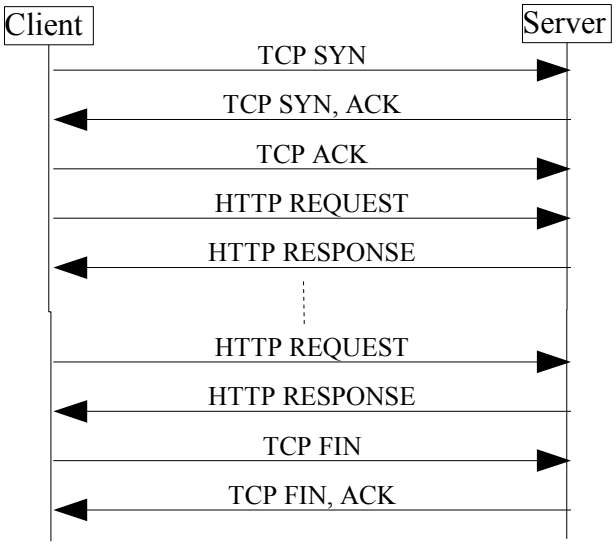

Figure 5. HTTP 1.1 Persistent Connection

Browsers can also "pipeline" requests for a new URL, submitting a request without waiting for the previous one to complete.

The agent acting as HTTP client also act as the TLS client. It initiates the connection, and then begin the TLS handshake. When the TLS handshake has finished, the client continue with the first HTTP request. All HTTP data is sent as TLS "application data".

Common practice has been to run HTTP/TLS over a separate port (default 443) in order to distinguish which protocol is being used.

4 Performance Evaluation

4.1 Testbed and Configurations

Two computers connected to the same subnetwork where used, one acting as server and the other as client.

The server, “iguana” (192.168.0.105) is a desktop PC with Linux, Fedora Core 3, Linux 2.6.11-1.14. 256M memory. OpenSSL provides TLS/SSL support, and Apache was used as the web server. The needed CA and supporting keys and certificates are created. Details on server configuration and installation procedure can be found at the end of this document.

The client, “chameleon” (192.168.0.102) is a laptop with Microsoft's Windows XP SP2. 512M memory. Microsoft's Internet Explorer, Netscape's Firefox and Opera are used as clients (browsers). Default settings on all browsers are used, except for cache size that is set to zero.

As for TLS configuration, we assumed that client authentication was not needed.

4.2 Tests and Data

We measured the performance as the elapsed time from the initial TCP SYN request to the last HTTP response (the most meaningful measurement point since some clients used persistent connections). The test was started by entering the URL to the test file (see Appendix) in the browser.

Test case 1: Plain HTTP request. The purpose of this test case was to simplify the analysis of the HTTP client behavior.

Test case 2: TLS protected HTTP request when TLS session resume is enabled.

Test case 3. TLS protected HTTP request when TLS session resume is disabled (see Appendix).

4.3 Explorer

4.3.1 Test case 1: Plain HTTP request

No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.0.102	Broadcast	ARP	Who has 192.168.0.104?
2	0.000222	192.168.0.104	192.168.0.102	ARP	192.168.0.104 is at 00:0a:e6:a1:2c:88
3	**0.000231**	**192.168.0.102**	**192.168.0.104**	**TCP**	**1160 > http [SYN]**
4	0.000425	192.168.0.104	192.168.0.102	TCP	http > 1160 [SYN, ACK]
5	0.000455	192.168.0.102	192.168.0.104	TCP	1160 > http [ACK]
6	0.000655	192.168.0.102	192.168.0.104	HTTP	GET /secure/index.htm HTTP/1.1
7	0.000881	192.168.0.104	192.168.0.102	TCP	http > 1160 [ACK]
8	0.001492	192.168.0.104	192.168.0.102	HTTP	HTTP/1.1 200 OK
...					
27	0.122687	192.168.0.102	192.168.0.104	HTTP	GET /secure/c1.htm HTTP/1.1
28	0.123592	192.168.0.104	192.168.0.102	HTTP	HTTP/1.1 200 OK (text/html)
29	0.123658	192.168.0.102	192.168.0.104	HTTP	GET /secure/c2.htm HTTP/1.1
30	0.124528	192.168.0.104	192.168.0.102	HTTP	HTTP/1.1 200 OK (text/html)
31	0.144789	192.168.0.102	192.168.0.104	HTTP	GET /secure/c3.htm HTTP/1.1
32	**0.145706**	**192.168.0.104**	**192.168.0.102**	**HTTP**	**HTTP/1.1 200 OK (text/html)**
33	0.285482	192.168.0.102	192.168.0.104	TCP	1160 > http [ACK]
34	0.285518	192.168.0.102	192.168.0.104	TCP	1161 > http [ACK]
...					

A detailed examination of the Ethereal capture reveals that the client opened two persistent connections to the server, using source ports 1160 and 1161. Pipelining is not used.

The total time to get the web pages is 0.146s.

4.3.2 Test Case 2: TLS protected HTTP request and session resumption

```
No.  Time  Source  Destination  Protocol Info

  1 0.000000  192.168.0.102  192.168.0.104  TCP  1181 > https [SYN]
  2 0.000226  192.168.0.104  192.168.0.102  TCP  https > 1181 [SYN, ACK]
  3 0.000252  192.168.0.102  192.168.0.104  TCP  1181 > https [ACK]
  4 0.000690  192.168.0.102  192.168.0.104  TLS  Client Hello
  5 0.000902  192.168.0.104  192.168.0.102  TCP  https > 1181 [ACK]
  6 0.001767  192.168.0.104  192.168.0.102  TLS  Server Hello, Certificate,
                                                 Server Hello Done
[...]
 32 0.041504  192.168.0.102  192.168.0.104  TLS  Client Hello
 33 0.041683  192.168.0.104  192.168.0.102  TCP  https > 1184 [ACK]
 34 0.041899  192.168.0.104  192.168.0.102  TLS  Server Hello, Change Cipher Spec,
                                                 Encrypted Handshake Message
 35 0.042004  192.168.0.102  192.168.0.104  TLS  Client Hello
 36 0.042195  192.168.0.104  192.168.0.102  TCP  https > 1185 [ACK]
 37 0.042685  192.168.0.102  192.168.0.104  TLS  Change Cipher Spec,
                                                 Encrypted Handshake Message
 38 0.042908  192.168.0.104  192.168.0.102  TLS  Server Hello, Change Cipher Spec,
                                                 Encrypted Handshake Message
[...]
144 0.147250  192.168.0.104  192.168.0.102  TLS  Application Data
145 0.147381  192.168.0.104  192.168.0.102  TLS  Application Data
146 0.147411  192.168.0.102  192.168.0.104  TCP  1190 > https [ACK]
147 0.147729  192.168.0.104  192.168.0.102  TCP  https > 1190 [FIN, ACK]
```

A closer look at the Ethereal capture reveals that the there are ten parallel connections to the server (source ports 1181 to 1190). First one is opened, the initial full TLS handshake is completed, data transferred and the connection is closed. Then all the remaining nine connections are opened simultaneously, the abbreviated handshake (as expected) completed, data transferred and finally connections are closed. Persistent connections are not used.

The total time to get the web pages is 0.147s.

4.3.3 Test Case 3: TLS protected HTTP request and no session resumption

```
No.  Time  Source  Destination  Protocol Info
  1 0.000000  192.168.0.102  Broadcast  ARP  Who has 192.168.0.104?  Tell 192.168.0.102
  2 0.000225  192.168.0.104  192.168.0.102  ARP  192.168.0.104 is at 00:0a:e6:a1:2c:88
  3 0.000235  192.168.0.102  192.168.0.104  TCP  1229 > https [SYN]
  4 0.000438  192.168.0.104  192.168.0.102  TCP  https > 1229 [SYN, ACK]
  5 0.000467  192.168.0.102  192.168.0.104  TCP  1229 > https [ACK]
  6 0.001011  192.168.0.102  192.168.0.104  TLS  Client Hello
  7 0.001239  192.168.0.104  192.168.0.102  TCP  https > 1229 [ACK]
  8 0.003157  192.168.0.104  192.168.0.102  TLS  Server Hello, Certificate,
                                                 Server Hello Done
[...]
 15 0.042058  192.168.0.102  192.168.0.104  TLS  Client Hello
 16 0.042261  192.168.0.104  192.168.0.102  TCP  https > 1230
 17 0.044349  192.168.0.104  192.168.0.102  TLS  Server Hello, Certificate,
                                                 Server Hello Done
 18 0.045588  192.168.0.102  192.168.0.104  TLS  Client Key Exchange,
                                                 Change Cipher Spec,
                                                 Encrypted Handshake Message
 19 0.052753  192.168.0.104  192.168.0.102  TLS  Change Cipher Spec, Finished
[...]
154 0.305903  192.168.0.102  192.168.0.104  TLS  Application Data
155 0.306849  192.168.0.104  192.168.0.102  TLS  Application Data
156 0.306945  192.168.0.104  192.168.0.102  TLS  Application Data
157 0.306976  192.168.0.102  192.168.0.104  TCP  1239 > https [ACK]
158 0.307311  192.168.0.104  192.168.0.102  TCP  https > 1239 [FIN, ACK]
```

The TCP connections have the same behaviors as in the previous test case. As expected, each connection must do the full TLS handshake.

The total time to get the web pages is 0.307s

4.4 Opera

4.4.1 Test Case 1: Plain HTTP request

```
No.  Time     Source    Destination   Protocol Info

 1 0.000000   192.168.0.102  Broadcast  ARP  Who has 192.168.0.104?  Tell 192.168.0.102
 2 0.000223   192.168.0.104  192.168.0.102  ARP  192.168.0.104 is at 00:0a:e6:a1:2c:88
 3 0.000232   192.168.0.102  192.168.0.104  TCP  1849 > http [SYN]
 4 0.000417   192.168.0.104  192.168.0.102  TCP  http > 1849 [SYN, ACK]
 5 0.000449   192.168.0.102  192.168.0.104  TCP  1849 > http [ACK]
 6 0.019242   192.168.0.102  192.168.0.104  HTTP  GET /secure/index.htm HTTP/1.1
 7 0.019562   192.168.0.104  192.168.0.102  TCP  http > 1849 [ACK]
 8 0.020721   192.168.0.104  192.168.0.102  HTTP  HTTP/1.1 200 OK
[...]
84 0.128123   192.168.0.102  192.168.0.104  HTTP  GET /secure/c3.htm HTTP/1.1
85 0.128392   192.168.0.104  192.168.0.102  TCP  http > 1857 [ACK]
86 0.129160   192.168.0.104  192.168.0.102  HTTP  HTTP/1.1 200 OK (text/html)
87 0.289463   192.168.0.102  192.168.0.104  TCP  1857 > http [ACK]
```

Web standards suggest a browser should use no more than two persistent connections per server, but Opera's default is 8 connections per server, where half of the "max connections per server" are persistent. Opera uses pipelining by default [5].

The Ethereal capture shows that nine connections are opened (source ports 1849 to 1847). All TCP signaling seems to be handled in lock-step – first the initial handshake is done in turn on all channels, next the HTTP request on all channels and finally the close handshake.

The total time to get the web pages is 0.129s.

4.4.2 Test Case 2: TLS protected HTTP request and session resumption

```
No.  Time  Source   Destination  Protocol Info

  1 0.000000  192.168.0.102  Broadcast  ARP  Who has 192.168.0.104?  Tell 192.168.0.102
  2 0.000214  192.168.0.104  192.168.0.102  ARP  192.168.0.104 is at 00:0a:e6:a1:2c:88
  3 0.000222  192.168.0.102  192.168.0.104  TCP  1941 > https [SYN]
  4 0.000423  192.168.0.104  192.168.0.102  TCP  https > 1941 [SYN, ACK]
  5 0.000454  192.168.0.102  192.168.0.104  TCP  1941 > https [ACK]
  6 0.022420  192.168.0.102  192.168.0.104  TLS  Client Hello
  7 0.022682  192.168.0.104  192.168.0.102  TCP  https > 1941 [ACK]
  8 0.043632  192.168.0.104  192.168.0.102  TLS  Server Hello, Certificate,
                                                 Server Key Exchange, Server Hello Done
  9 0.154863  192.168.0.102  192.168.0.104  TCP  1941 > https [ACK]
 10 0.350160  192.168.0.102  192.168.0.104  TLS  Client Key Exchange
 11 0.390380  192.168.0.104  192.168.0.102  TCP  https > 1941 [ACK]
 12 0.390452  192.168.0.102  192.168.0.104  TLS  Change Cipher Spec,
                                                 Encrypted Handshake Message
 13 0.390623  192.168.0.104  192.168.0.102  TCP  https > 1941 [ACK]
 14 0.391184  192.168.0.104  192.168.0.102  TLS  Change Cipher Spec,
                                                 Encrypted Handshake Message
 15 0.394637  192.168.0.102  192.168.0.104  TLS  Application Data
[...]
 61 0.462955  192.168.0.102  192.168.0.104  TLS  Client Hello
 62 0.463107  192.168.0.104  192.168.0.102  TCP  https > 1949 [ACK]
 63 0.463186  192.168.0.104  192.168.0.102  TLS  Server Hello, Change Cipher Spec,
                                                 Encrypted Handshake Message
 64 0.464693  192.168.0.102  192.168.0.104  TLS  Change Cipher Spec
[...]
166 0.648339  192.168.0.102  192.168.0.104  TLS  Encrypted Handshake Message,
                                                 Application Data
167 0.648597  192.168.0.104  192.168.0.102  TCP  https > 1950 [ACK]
168 0.649513  192.168.0.104  192.168.0.102  TLS  Application Data
```

The TCP connections have the same behaviors as in the plain HTTP case, except that one more channel is set up.

The first connection do the full TLS handshake, the following the abbreviated handshake. The total time to get the web pages is 0.650s.

4.4.3 Test Case 3: TLS protected HTTP request and no session resumption

```
No.  Time  Source   Destination  Protocol Info
  1 0.000000  192.168.0.102  192.168.0.104  TCP  1244 > https [SYN]
  2 0.000247  192.168.0.104  192.168.0.102  TCP  https > 1244 [SYN, ACK]
  3 0.000279  192.168.0.102  192.168.0.104  TCP  1244 > https [ACK]
  4 0.021871  192.168.0.102  192.168.0.104  TLS  Client Hello
  5 0.022117  192.168.0.104  192.168.0.102  TCP  https > 1244 [ACK]
  6 0.040686  192.168.0.104  192.168.0.102  TLS  Server Hello, Certificate,
                                              Server Key Exchange, Server Hello Done
  7 0.171995  192.168.0.102  192.168.0.104  TCP  1244 > https [ACK]
  8 0.338912  192.168.0.102  192.168.0.104  TLS  Client Key Exchange
  9 0.378466  192.168.0.104  192.168.0.102  TCP  https > 1244 [ACK]
 10 0.378529  192.168.0.102  192.168.0.104  TLS  Change Cipher Spec,
                                              Encrypted Handshake Message
 11 0.378692  192.168.0.104  192.168.0.102  TCP  https > 1244 [ACK]
 12 0.379267  192.168.0.104  192.168.0.102  TLS  Change Cipher Spec,
                                              Encrypted Handshake Message
[...]
 44 0.450395  192.168.0.102  192.168.0.104  TLS  Client Hello
 45 0.450570  192.168.0.104  192.168.0.102  TCP  https > 1245 [ACK]
 46 0.469163  192.168.0.104  192.168.0.102  TLS  Server Hello, Certificate,
                                              Server Key Exchange, Server Hello Done
 47 0.573282  192.168.0.102  192.168.0.104  TCP  1245 > https [ACK]
 48 0.779757  192.168.0.102  192.168.0.104  TLS  Client Key Exchange
 49 0.819350  192.168.0.104  192.168.0.102  TCP  https > 1245 [ACK]
 50 0.819424  192.168.0.102  192.168.0.104  TLS  Change Cipher Spec,
                                              Encrypted Handshake Message
 51 0.819588  192.168.0.104  192.168.0.102  TCP  https > 1245 [ACK]
 52 0.820159  192.168.0.104  192.168.0.102  TLS  Change Cipher Spec,
                                              Encrypted Handshake Message
[...]
194 4.226310  192.168.0.102  192.168.0.104  TLS  Application Data
195 4.227693  192.168.0.104  192.168.0.102  TLS  Application Data
```

The TCP connections are handled as in test case 2.

All connections do the full TLS handshake. The total time to get the web pages is 4.228s.

4.5 Firefox

4.5.1 Test Case 1: Plain HTTP request

```
No.  Time    Source    Destination   Protocol Info

 1 0.000000   192.168.0.102  192.168.0.104  TCP  1152 > http [SYN]
 2 0.000269   192.168.0.104  192.168.0.102  TCP  http > 1152 [SYN, ACK]
 3 0.000302   192.168.0.102  192.168.0.104  TCP  1152 > http [ACK]
 4 0.027932   192.168.0.102  192.168.0.104  HTTP  GET /secure/index.htm HTTP/1.1
 5 0.028221   192.168.0.104  192.168.0.102  TCP  http > 1152 [ACK]
 6 0.028808   192.168.0.104  192.168.0.102  HTTP  HTTP/1.1 200 OK
...
33 0.295476   192.168.0.102  192.168.0.104  HTTP  GET /favicon.ico HTTP/1.1
34 0.296465   192.168.0.104  192.168.0.102  HTTP  HTTP/1.1 404 Not Found (text/html)
35 0.339371   192.168.0.102  192.168.0.104  TCP  1152 > http [ACK]
36 0.439683   192.168.0.102  192.168.0.104  TCP  1153 > http [ACK]
...
```

Two persistent TCP connections are used (source ports 1152 and 1153). Pipelining is not used.

The total time to get the web pages is 0.296s.

4.5.2 Test Case 2: TLS protected HTTP request and session resumption

```
No.  Time  Source  Destination  Protocol Info

  1 0.000000  192.168.0.102  Broadcast  ARP  Who has 192.168.0.104?  Tell 192.168.0.102
 2 0.000225  192.168.0.104  192.168.0.102  ARP  192.168.0.104 is at 00:0a:e6:a1:2c:88
 3 0.000234  192.168.0.102  192.168.0.104  TCP  2012 > https [SYN]
 4 0.000435  192.168.0.104  192.168.0.102  TCP  https > 2012 [SYN, ACK]
 5 0.000462  192.168.0.102  192.168.0.104  TCP  2012 > https [ACK]
 6 0.000830  192.168.0.102  192.168.0.104  TLS  Client Hello
 7 0.001028  192.168.0.104  192.168.0.102  TCP  https > 2012 [ACK]
 8 0.021983  192.168.0.104  192.168.0.102  TLS  Server Hello, Certificate, Server Key
Exchange, Server Hello Done
9 0.039220  192.168.0.102  192.168.0.104  TLS  Client Key Exchange, Change Cipher Spec,
Encrypted Handshake Message
10 0.052956  192.168.0.104  192.168.0.102  TLS  Change Cipher Spec, Client Key Exchange

[...]
23 0.171934  192.168.0.102  192.168.0.104  TLS  Client Hello
24 0.172145  192.168.0.104  192.168.0.102  TCP  https > 2013 [ACK]
25 0.172584  192.168.0.104  192.168.0.102  TLS  Application Data
26 0.172752  192.168.0.104  192.168.0.102  TLS  Application Data
27 0.172776  192.168.0.102  192.168.0.104  TCP  2012 > https [ACK]
28 0.174294  192.168.0.104  192.168.0.102  TLS  Server Hello, Change Cipher Spec,
Encrypted Handshake Message
29 0.181015  192.168.0.102  192.168.0.104  TLS  Change Cipher Spec, Encrypted Handshake
Message, Application Data

[...]
61 0.382208  192.168.0.102  192.168.0.104  TLS  Application Data
62 0.383326  192.168.0.104  192.168.0.102  TLS  Application Data
63 0.383408  192.168.0.104  192.168.0.102  TLS  Application Data
64 0.383438  192.168.0.102  192.168.0.104  TCP  2012 > https [ACK]
```

The TCP connections are handled as in the plain HTTP test case.

The first connection do the full TLS handshake, the second the abbreviated handshake. The total time to get the web pages is 0.383s.

4.5.3 Test Case 3: TLS protected HTTP request and no session resumption

```
No.  Time  Source  Destination  Protocol Info
 1 0.000000  192.168.0.102  192.168.0.104  TCP  2072 > https [SYN]
 2 0.000252  192.168.0.104  192.168.0.102  TCP  https > 2072 [SYN, ACK]
 3 0.000280  192.168.0.102  192.168.0.104  TCP  2072 > https [ACK]
 4 0.000777  192.168.0.102  192.168.0.104  TLS  Client Hello
 5 0.000995  192.168.0.104  192.168.0.102  TCP  https > 2072 [ACK]
 6 0.023493  192.168.0.104  192.168.0.102  TLS  Server Hello, Certificate, Server Key
Exchange, Server Hello Done
 7 0.041907  192.168.0.102  192.168.0.104  TLS  Client Key Exchange, Change Cipher Spec,
Encrypted Handshake Message
 8 0.055869  192.168.0.104  192.168.0.102  TLS  Change Cipher Spec, Encrypted Handshake
Message
[...]
17 0.184936  192.168.0.102  192.168.0.104  TLS  Client Hello
18 0.185119  192.168.0.104  192.168.0.102  TCP  https > 2073 [ACK]
19 0.185909  192.168.0.104  192.168.0.102  TLS  Application Data, Application Data
20 0.193301  192.168.0.102  192.168.0.104  TLS  Application Data
21 0.202659  192.168.0.104  192.168.0.102  TLS  Application Data, Application Data
22 0.203341  192.168.0.102  192.168.0.104  TLS  Application Data
23 0.208000  192.168.0.104  192.168.0.102  TLS  Server Hello, Certificate, Server Key
Exchange, Server Hello Done
24 0.208838  192.168.0.104  192.168.0.102  TLS  Application Data, Application Data
25 0.225254  192.168.0.102  192.168.0.104  TLS  Client Key Exchange, Change Cipher Spec,
Encrypted Handshake Message
[...]
40 0.418995  192.168.0.104  192.168.0.102  TLS  Application Data, Application Data
41 0.537393  192.168.0.102  192.168.0.104  TCP  2073 > https [ACK]
42 0.537426  192.168.0.102  192.168.0.104  TCP  2072 > https [ACK]
```

The TCP connections are handled as in the plain HTTP test case.

All connections do the full TLS handshake. The total time to get the web pages is 0.419s.

4.6 Test Result Summary

	TC1 - plain HTTP	*TC2 – HTTP over TLS*	*TC3 - HTTP over TSL, no cache*	*Difference cache-no cache*
Internet Explorer	0.146s	0.147s	0.307s	109%
Opera	0.129s	0.650	4.228s	550%
Firefox	0.296s	0.383	0.419s	9%

5 Conclusion

In this paper, TLS session resumption impact on HTTP performance has been presented and analyzed by using and comparing three different web clients, Microsoft's Internet Explorer, Netscape's Firefox and Opera.

As expected, disabling session resumption affects performance in a negative way. However, it seems like that the specific implementation of the HTTP client has the most influence on the overall performance.

6 References

[1] Ralf S. Engelschall: “User Manual mod_ssl ver 2.8” [homepage on the Internet] Available from: http://www.modssl.org/docs/2.8/ [cited September 2005]

[2] T. Berners-Lee, R. Fielding and H. Frystyk. “Hypertext Transfer Protocol -- HTTP/1.0”, RFC 1945. IETF May 1996

[3] R. Fielding et al. “Hypertext Transfer Protocol -- HTTP/1.1”, RFC 2616. IETF, June 1999

[4] E. Rescorla. “HTTP Over TLS”, RFC 2818. IETF, May 2000

[5] Opera Browser Wiki [homepage on the Internet] Available from: http://nontroppo.org/wiki/HttpProtocol [cited September 2005]

[6] T. Dierks and C. Allen, "The TLS Protocol - Version 1.0", RFC 2246, IETF, January 1999.

7 Appendix

7.1 Test Object

The test file is a simple HTML file named index.htm. It have a frame set of 3x3 frames, each of them include a tiny HTLM file. In total 10 files that must be fetched when the test page is requested.

```
<html>
<FRAMESET COLS="33%,33%,33%">
  <FRAMESET ROWS="33%,33%,33%">
    <FRAME SRC="a1.htm">
    <FRAME SRC="a2.htm">
    <FRAME SRC="a3.htm">
  </FRAMESET>
  <FRAMESET ROWS="33%,33%,33%">
    <FRAME SRC="b1.htm">
    <FRAME SRC="b2.htm">
    <FRAME SRC="b3.htm">
  </FRAMESET>
  <FRAMESET ROWS="33%,33%,33%">
    <FRAME SRC="c1.htm">
    <FRAME SRC="c2.htm">
    <FRAME SRC="c3.htm">
  </FRAMESET>
</FRAMESET>
</html>
```

One of the included files (a1.htm) is shown below (all are designed in the same way):

```
<html>
<head>
<title>A1</title>
</head>
<body>
A1
</body>
</html>
```

7.2 Server Configuration

7.2.1 OpenSSL, CA and Certificates

OpenSSL is a open source cryptographic library. It provides implementations of encryption algorithms as well as message digest algorithms and message authentication codes. We installed OpenSSL simply by using yum:

```
[root@localhost ca]#yum install openssl.i386
[root@localhost ca]#yum install openssl-devel.i386 //needed by Apache
```

OpenSSL supports several certificate formats. Certificates are based on the DSA signature algorithm and the RSA algorithm for public-key cryptography according to PKCS algorithms. Certificate format depends on the application, but available formats are Privacy Enhanced Mail (PEM) and Distinguished Encoding Rules(DER).and PKCS12

OpenSSL supports two standard formats for storing and exchanging key pairs, PEM and DER.

Certificate files are ASN.1-encoded objects that may be encrypted according to DES (Data Encryption Standard). The files can optionally be encrypted using a symmetric cipher algorithm, such as 3DES.

First we create a RSA private key for the server. It will be Triple-DES encrypted and in PEM format.

$ openssl genrsa -des3 -out server.key 1024

```
[root@localhost ca]# openssl genrsa -des3 -out server.key 1024
Generating RSA private key, 1024 bit long modulus
.....................................................++++++
.............++++++
e is 65537 (0x10001)
Enter pass phrase for server.key:
Verifying - Enter pass phrase for server.key:
[root@localhost ca]#
```

Then we create a Certificate Signing Request (CSR) using the server RSA private key:

```
[root@localhost ca]# openssl req -new -key server.key -out server.csr
Enter pass phrase for server.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [GB]:SE
State or Province Name (full name) [Berkshire]:n/a
Locality Name (eg, city) [Newbury]:Stockholm
Organization Name (eg, company) [My Company Ltd]:KTH
Organizational Unit Name (eg, section) []:2G1305
Common Name (eg, your name or your server's hostname) []:chikchak.homeip.net
Email Address []:asap@kth.se

Please enter the following 'extra' attributes to be sent with your certificate request
A challenge password []:
An optional company name []:
[root@localhost ca]#
```

The Certificate Signing Request (CSR) must be signed by a Certifying Authority (CA):

```
[root@localhost ca]# ./sign.sh server.csr
CA signing: server.csr -> server.crt:
Using configuration from ca.config
Enter pass phrase for ./ca.key:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'SE'
stateOrProvinceName   :PRINTABLE:'n/a'
localityName          :PRINTABLE:'Stockholm'
organizationName      :PRINTABLE:'KTH'
organizationalUnitName:PRINTABLE:'2G1305'
commonName            :PRINTABLE:'chikchak.homeip.net'
emailAddress          :IA5STRING:'asap@kth.se'
Certificate is to be certified until Aug 28 17:05:12 2006 GMT (365 days)
Sign the certificate? [y/n]:y
1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
CA verifying: server.crt <-> CA cert
server.crt: OK
[root@localhost ca]#
```

7.2.2 Apache

We selected Apache as web server. There are two open-source choices when it comes to using SSL with Apache – either the mod_ssl module or the Apache-SSL distribution (both require OpenSSL). In version 2.0 and later mod_ssl is shipped with the source code distribution. We download, unzip and build the code:

```
[root@localhost ca]#./configure –enable-ssl
[root@localhost ca]#make
[root@localhost ca]#make install
```

Apache is installed at the default location, /usr/local/apache2.

Next, we modified /usr/local/apache_ssl/conf/ssl.conf (modifications are in bold). Line 6 was uncommented to disable session resumption.

```
1 <IfDefine SSL>
2 Listen 443
3 [...]
4 # Inter-Process Session Cache: Configure the SSL Session Cache: First the mechanism
5 # to use and second the expiring timeout (in seconds).
6 # We will modify thereparameters later when evaluating performance.
7 #SSLSessionCache        none
8 SSLSessionCache         dbm:/usr/local/apache2/logs/ssl_scache
```

```
 9 SSLSessionCacheTimeout  60
10 <VirtualHost _default_:443>
11 #   General setup for the virtual host
12 DocumentRoot "/usr/local/apache2/htdocs"
13 ServerName chikchak.homeip.net:443
14 ServerAdmin asap@kth.se
15 ErrorLog /usr/local/apache2/logs/error_log
16 TransferLog /usr/local/apache2/logs/access_log
17
18 #   SSL Engine Switch: Enable SSL for this virtual host.
19 SSLEngine on
20
21 #   SSL Cipher Suite: Lists the ciphers that the client is permitted to negotiate.
21 SSLCipherSuite ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL
22
23 #   Server Certificate: Point SSLCertificateFile at a PEM encoded certificate.
24 #   Server Private Key: If the key is not combined with the certificate, use this
25 #   directive to point at the key file.
26
27 SSLCertificateFile /usr/local/apache2/conf/ca/server.crt
28 SSLCertificateKeyFile /usr/local/apache2/conf/ca/server.key
29
30 #There is no need to have client authentication in this test.
31 SSLVerifyClient none
32 </VirtualHost>
33 </IfDefine>
```

Apache was started:

```
[root@localhost conf]# /usr/local/apache2/bin/apachectl -D SSL -k start
Apache/2.0.54 mod_ssl/2.0.54 (Pass Phrase Dialog)
Some of your private key files are encrypted for security reasons.
In order to read them you have to provide us with the pass phrases.

Server chikchak.homeip.net:443 (RSA)
Enter pass phrase:
Ok: Pass Phrase Dialog successful.

[root@localhost conf]#
```