

## **WEB SERVER** **(Prof. Fischetti Pietro)**

In informatica un server web è un'applicazione software che, in esecuzione su un server, è in grado di gestire le richieste di trasferimento di pagine web di un client, tipicamente un web browser[1]. La comunicazione tra server e client avviene tramite il protocollo HTTP, che utilizza la porta TCP di default 80 (o 8080), o eventualmente la versione sicura HTTPS, che utilizza invece di default la 443. Su un server web risiedono dunque i siti web tramite hosting. L'insieme di tutti i server web interconnessi a livello mondiale dà vita al World Wide Web.

### XAMPP (windows)

Il pacchetto Xampp (nel proseguo si ipotizza che sia stato caricato nella directory c:\xampp) per windows (portabile quindi gestibile e configurabile anche se non si è amministratori) contiene tra le altre cose il web server Apache portato dal mondo Linux nel mondo Windows.

Come prima cosa si controlli le porte disponibili con il comando netstat -n in modo da configurare opportunamente apache utilizzando eventualmente altre porte se quella http 80 https 443 sono occupate (si ricorda che non può esserci più di un servizio in ascolto su una porta).

I siti devono essere memorizzati nella cartella: C:\xampp\htdocs come sottocartelle, ad esempio se voglio un sito che chiamo mySite che contiene semplicemente un file il cui contenuto è: ciao

il file può avere qualsiasi nome ma se si chiama index.html verrà presentato di default quando viene richiesto dal browser: <http://localhost/mysite> non server <http://localhost/mysite/index.html>, mentre se volessi un altro file devo specificarlo ad es <http://localhost/mysite/pagina2.html> ma anche <http://localhost/mysite/es.txt>. Questi sono file statici ma se voglio rendere le risposte dinamiche cioè dipendenti da qualche parametro passato dal browser posso usare il linguaggio PHP. in pratica viene eseguito del codice lato server racchiuso da dei tag speciali. Se il Web Server è correttamente configurato ad esempio:

XAMPP Control Panel v3.2.2 [ Compiled: Nov 12th 2015 ]

## XAMPP Control Panel v3.2.2

Service	Module	PID(s)	Port(s)	Actions
<input type="checkbox"/>	Apache	2484 7248	80, 443	Stop Admin Config Logs
<input type="checkbox"/>	MySQL	8636	3306	Stop Admin Config Logs
<input type="checkbox"/>	FileZilla			Start Admin Config Logs
<input type="checkbox"/>	Mercury			Start Admin Config Logs
<input type="checkbox"/>	Tomcat			Start Admin Config Logs

12:42:54 [main] XAMPP Installation Directory: "c:\xampp\  
 12:42:54 [main] Checking for prerequisites  
 12:42:56 [main] All prerequisites found  
 12:42:56 [main] Initializing Modules  
 12:42:56 [Apache] XAMPP Apache is already running on port 443  
 12:42:56 [mysql] XAMPP MySQL is already running on port 3306  
 12:42:56 [main] Starting Check-Timer  
 12:42:56 [main] Control Panel Ready

Ma nel caso il server http dalla porta 80 si potrebbe ad esempio spostare sulla porta 8080 (ovvio se libera altrimenti si cerchi una libera) modificando il file httpd.conf attivabile dal bottone Config, e cercando la linea: Listen 80



## XAMPP Control Panel v3.2.2

Modules

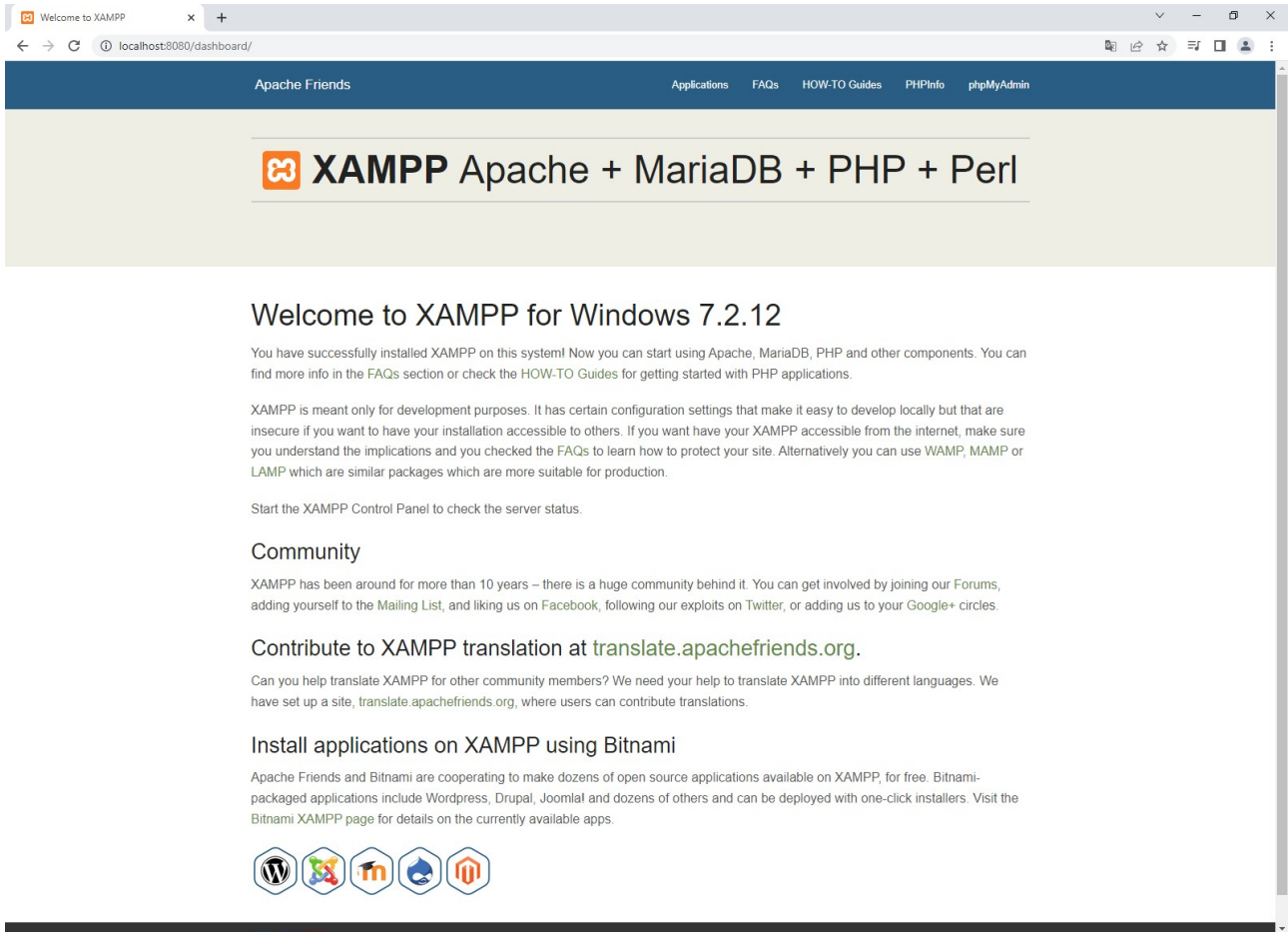
Service	Module	PID(s)	Port(s)	Actions
<input type="checkbox"/>	Apache	15712 15776	443, 8080	Stop Admin Config Logs
<input type="checkbox"/>	MySQL	14544	3306	Stop Admin Config Logs
<input type="checkbox"/>	FileZilla			Start Admin Config Logs
<input type="checkbox"/>	Mercury			Start Admin Config Logs
<input type="checkbox"/>	Tomcat			Start Admin Config Logs

- Config
- Netstat
- Shell
- Explorer
- Services
- Help
- Quit

```

10:41:44 [main]      Initializing Control Panel
10:41:44 [main]      Windows Version: Enterprise 64-bit
10:41:44 [main]      XAMPP Version: 7.2.12
10:41:44 [main]      Control Panel Version: 3.2.2 [ Compiled: Nov 12th 2015 ]
10:41:44 [main]      You are not running with administrator rights! This will work for
10:41:44 [main]      most application stuff but whenever you do something with services
10:41:44 [main]      there will be a security dialogue or things will break! So think
10:41:44 [main]      about running this application with administrator rights!
10:41:44 [main]      XAMPP Installation Directory: "c:\xampp\"
10:41:44 [main]      Checking for prerequisites
10:41:46 [main]      All prerequisites found
10:41:46 [main]      Initializing Modules
10:41:46 [main]      Starting Check-Timer
10:41:46 [main]      Control Panel Ready
10:42:28 [Apache]     Attempting to stop Apache (PID: 15028)
10:42:28 [Apache]     Attempting to stop Apache (PID: 14856)
10:42:28 [Apache]     Status change detected: stopped
10:42:29 [Apache]     Attempting to start Apache app...
10:42:29 [Apache]     Status change detected: running
    
```

In questo caso digitando nel browser: <http://localhost:8080> dovrebbe apparire la pagina splash di Apache



Vediamo ora come creare un proprio sito ad esempio di nome mySite-

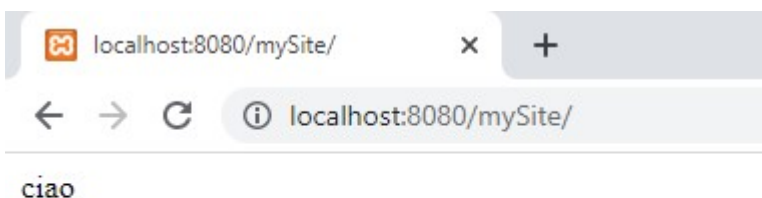
Creare il percorso: `c:\xampp\htdocs\mySite`

E crearci un file `index.html`

che contiene semplicemente la parola `ciao`:

<code>index.html</code>
<code>ciao</code>

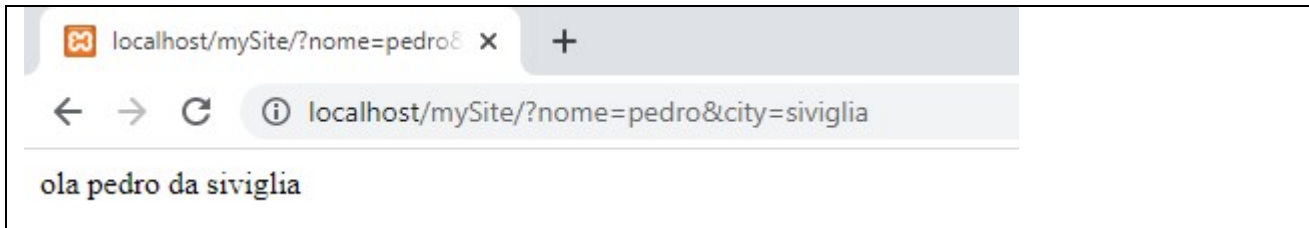
Digitare nel browser:



Vediamo come passare dei parametri dal browser ad un nostro programma (che chiamo ad esempio Index.php) in esecuzione sul Web Server. I metodi maggiormente utilizzati sono GET e POST. Con il metodo GET i parametri possono inserirli direttamente nella barra come coppie nome=valore separate dal carattere &. Quindi mettiamo nella dir mySite il file:

```
Index.php
<?php
print "ola " . $_GET["nome"] . " da " . $_GET["city"];
?>
```

Digitiamo nel browser: <http://localhost/mySite/?nome=pedro&city=siviglia>



Questo avviene perché l'URL viene passato nella query string del server:

modificando il file index.php:

```
Index.php
<?php
print "ola " . $_GET["nome"] . " da " . $_GET["city"];

echo "<pre>";
print_r($_SERVER);
echo "</pre>";
?>
```

Posso vedere nel dettaglio le stringhe configurate dal Web Server per il mio codice php.

localhost/mySite/?nome=pedro&city=siviglia

ola pedro da siviglia

Array

```
(
  [MIBDIRS] => C:/xampp/php/extras/mibs
  [MYSQL_HOME] => \xampp\mysql\bin
  [OPENSSL_CONF] => C:/xampp/apache/bin/openssl.cnf
  [PHP_PEAR_SYSCONF_DIR] => \xampp\php
  [PHPRC] => \xampp\php
  [TMP] => \xampp\tmp
  [HTTP_HOST] => localhost
  [HTTP_CONNECTION] => keep-alive
  [HTTP_CACHE_CONTROL] => max-age=0
  [HTTP_SEC_CH-UA] => "Google Chrome";v="111", "Not(A:Brand";v="8", "Chromium";v="111"
  [HTTP_SEC_CH-UA-MOBILE] => ?0
  [HTTP_SEC_CH-UA-PLATFORM] => "Windows"
  [HTTP_UPGRADE_INSECURE_REQUESTS] => 1
  [HTTP_USER_AGENT] => Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, l:
  [HTTP_ACCEPT] => text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp
  [HTTP_SEC_FETCH_SITE] => none
  [HTTP_SEC_FETCH_MODE] => navigate
  [HTTP_SEC_FETCH_USER] => ?1
  [HTTP_SEC_FETCH_DEST] => document
  [HTTP_ACCEPT_ENCODING] => gzip, deflate, br
  [HTTP_ACCEPT_LANGUAGE] => it-IT,it;q=0.9,en-US;q=0.8,en;q=0.7
  [HTTP_COOKIE] => PHPSESSID=r4iu842so6s334sqe88504ad72
  [PATH] => C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\Windo
  [SystemRoot] => C:\Windows
  [COMSPEC] => C:\Windows\system32\cmd.exe
  [PATHEXT] => .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
  [WINDIR] => C:\Windows
  [SERVER_SIGNATURE] =>
    Apache/2.4.28 (Win32) OpenSSL/1.0.2l PHP/7.1.10 Server at Localhost Port 80

  [SERVER_SOFTWARE] => Apache/2.4.28 (Win32) OpenSSL/1.0.2l PHP/7.1.10
  [SERVER_NAME] => localhost
  [SERVER_ADDR] => ::1
  [SERVER_PORT] => 80
  [REMOTE_ADDR] => ::1
  [DOCUMENT_ROOT] => C:/xampp/htdocs
  [REQUEST_SCHEME] => http
  [CONTEXT_PREFIX] =>
  [CONTEXT_DOCUMENT_ROOT] => C:/xampp/htdocs
  [SERVER_ADMIN] => postmaster@localhost
  [SCRIPT_FILENAME] => C:/xampp/htdocs/mySite/index.php
  [REMOTE_PORT] => 60106
  [GATEWAY_INTERFACE] => CGI/1.1
  [SERVER_PROTOCOL] => HTTP/1.1
  [REQUEST_METHOD] => GET
  [QUERY_STRING] => nome=pedro&city=siviglia
  [REQUEST_URI] => /mySite/?nome=pedro&city=siviglia
  [SCRIPT_NAME] => /mySite/index.php
  [PHP_SELF] => /mySite/index.php
  [REQUEST_TIME_FLOAT] => 1679498446.721
  [REQUEST_TIME] => 1679498446
)
```

Si veda il valore [QUERY\_STRING] che contiene le coppie nome=valore

Vediamo ora di utilizzare un file html con richiesta GET:

esGETPOST.html

```
<!DOCTYPE html>
<html>
<body>

<form action="http://localhost/mySite/my.php" method="GET">
  <label for="nome">nome:</label>
  <input type="text" id="nome" name="nome"><br><br>
  <label for="city">city:</label>
  <input type="text" id="city" name="city"><br><br>
```



```
<input type="submit" value="Submit">
</form>

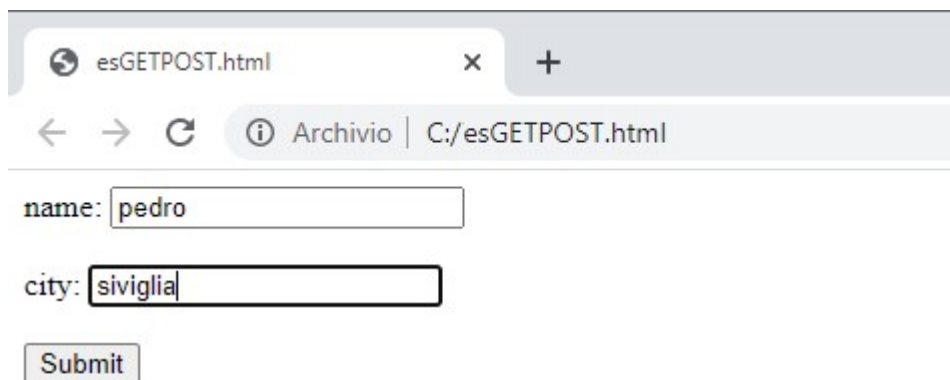
</body>
</html>
```

Inoltre aggiungiamo nel file index.php la gestione del case GET o POST

```
Index.php

<?php
    print "ola ";

    if($_SERVER['REQUEST_METHOD']=="GET"){
        if(isset($_GET["nome"]))
            print $_GET["nome"];
        if(isset($_GET["city"]))
            print " da " . $_GET["city"];
    }
    elseif($_SERVER['REQUEST_METHOD']=="POST"){
        if(isset($_POST["nome"]))
            print $_POST["nome"];
        if(isset($_POST["city"]))
            print " da " . $_POST["city"];
    }
    echo "<pre>";
    print_r($_SERVER);
    echo "</pre>";
?>
```



The screenshot shows a web browser window with a single tab titled 'esGETPOST.html'. The address bar shows 'C:/esGETPOST.html'. The page content includes a form with two input fields: 'name:' with the value 'pedro' and 'city:' with the value 'siviglia'. Below the fields is a 'Submit' button.

```
localhost/mySite/r x +
localhost/mySite/?nome=pedro&city=siviglia
ola pedro
Array
(
    [MIBDIRS] => C:/xampp/php/extras/mibs
    [MYSQL_HOME] => \xampp\mysql\bin
    [OPENSSL_CONF] => C:/xampp/apache/bin/openssl.cnf
    [PHP_PEAR_SYSCONF_DIR] => \xampp\php
    [PHPRC] => \xampp\php
    [TMP] => \xampp\tmp
    [HTTP_HOST] => localhost

    .....

    [CONTEXT_DOCUMENT_ROOT] => C:/xampp/htdocs
    [SERVER_ADMIN] => postmaster@localhost
    [SCRIPT_FILENAME] => C:/xampp/htdocs/mySite/my.php
    [REMOTE_PORT] => 51622
    [GATEWAY_INTERFACE] => CGI/1.1
    [SERVER_PROTOCOL] => HTTP/1.1
    [REQUEST_METHOD] => GET
    [QUERY_STRING] => nome=pedro&city=siviglia
    [REQUEST_URI] => /mySite/my.php?nome=pedro&city=siviglia
    [SCRIPT_NAME] => /mySite/my.php
    [PHP_SFILE] => /mySite/my.php
```

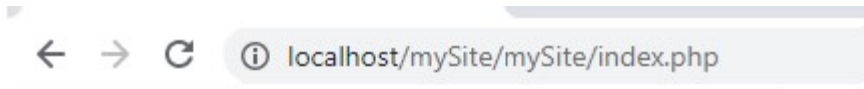
Nel metodo POST i parametri non vengono trasmessi nell'url come coppie. Ma sono inseriti nel corpo della richiesta, quindi non sono visibili e possono superare le limitazioni in lunghezza dell'URL presenti nel metodo GET (max 2048 caratteri)

```
esGETPOST.html
<!DOCTYPE html>
<html>
<body>

<form action="http://localhost/mySite/index.php" method="POST">
  <label for="nome">nome:</label>
  <input type="text" id="nome" name="nome"><br><br>
  <label for="city">city:</label>
  <input type="text" id="city" name="city"><br><br>
  <input type="submit" value="Submit">
</form>

</body>
</html>
```



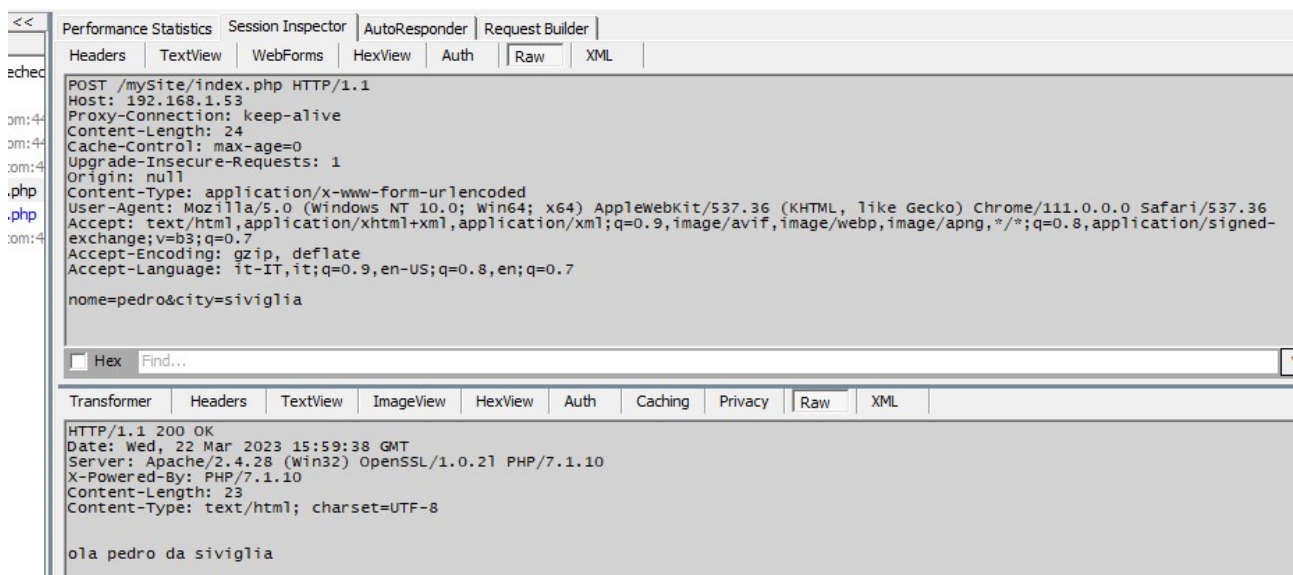


ola pedro da siviglia

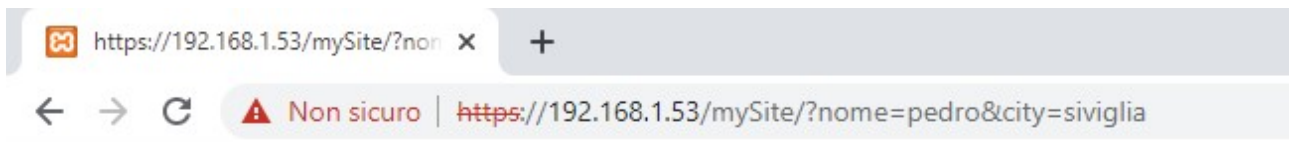
Array

```
(  
  [MIBDIRS] => C:/xampp/php/extras/mibs  
  [MYSQL_HOME] => \xampp\mysql\bin  
  [OPENSSL_CONF] => C:/xampp/apache/bin/openssl.cnf  
  [PHP_PEAR_SYSCONF_DIR] => \xampp\php  
  [PHPRC] => \xampp\php  
  [TMP] => \xampp\tmp  
  [HTTP_HOST] => localhost  
  
  .....  
  .....  
  
  [CONTEXT_DOCUMENT_ROOT] => C:/xampp/htdocs  
  [SERVER_ADMIN] => postmaster@localhost  
  [SCRIPT_FILENAME] => C:/xampp/htdocs/mySite/my.php  
  [REMOTE_PORT] => 51634  
  [GATEWAY_INTERFACE] => CGI/1.1  
  [SERVER_PROTOCOL] => HTTP/1.1  
  [REQUEST_METHOD] => POST  
  [QUERY_STRING] =>  
  [REQUEST_URI] => /mySite/my.php  
  [SCRIPT_NAME] => /mySite/my.php  
  [PHP_SELF] => /mySite/my.php  
  [REQUEST_TIME_FLOAT] => 1.678136668 000
```

Si nota che sono sparite le coppie nome=valore dalla barra URL del browser. Si potrebbe controllare per esercizio il traffico Request-Response tramite Ispezione dal Browser o con Wireshark, oppure con Fiddler.



Nella figura seguente la risposta ad una richiesta https di tipo GET:

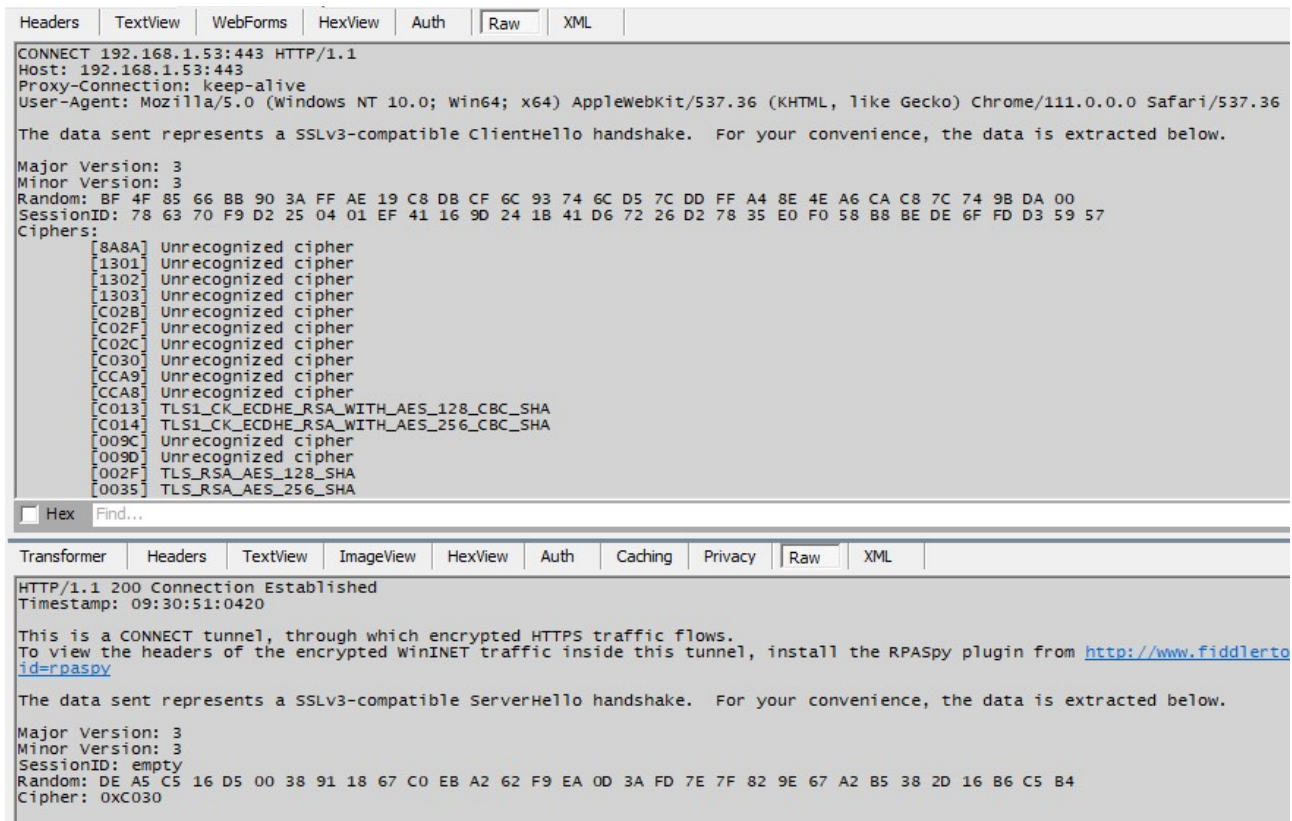


ola pedro da siviglia

Array

```
(  
  [MIBDIRS] => C:/xampp/php/extras/mibs  
  [MYSQL_HOME] => \xampp\mysql\bin  
  [OPENSSL_CONF] => C:/xampp/apache/bin/openssl.cnf  
  [PHP_PEAR_SYSCONF_DIR] => \xampp\php  
  [PHPRC] => \xampp\php  
  [TMP] => \xampp\tmp  
  [HTTPS] => on  
  [SSL_SERVER_S_DN_C] => IT  
  [SSL_SERVER_S_DN_ST] => Italy  
  [SSL_SERVER_S_DN_L] => Genova  
  [SSL_SERVER_S_DN_O] => acme  
  [SSL_SERVER_S_DN_OU] => cpp  
  [SSL_SERVER_S_DN_CN] => mysite.dev  
  [SSL_SERVER_S_DN_Email] => my@acme.it  
  [SSL_SERVER_I_DN_C] => IT  
  [SSL_SERVER_I_DN_ST] => Italy  
  [SSL_SERVER_I_DN_L] => Genova  
  [SSL_SERVER_I_DN_O] => acme  
  [SSL_SERVER_I_DN_OU] => cpp  
  [SSL_SERVER_I_DN_CN] => mysite.dev  
  [SSL_SERVER_I_DN_Email] => my@acme.it  
  [SSL_VERSION_INTERFACE] => mod_ssl/2.4.28  
  [SSL_VERSION_LIBRARY] => OpenSSL/1.0.2j  
  [SSL_PROTOCOL] => TLSv1.2
```

Nella figura seguete la cattura del traffico con Fiddler



## CGI

In informatica Common Gateway Interface (CGI) è una tecnologia standard usata dai web server per interfacciarsi con applicazioni esterne generando contenuti web dinamici. In pratica possiamo utilizzare dei programmi eseguibili ad esempio scritti in C per essere richiamati dal Web Server. Esempio: un programma C che calcoli la somma di 2 numeri:

```
cgisum.c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char **argv, char** env) {
int n1=0,n2=0,r;

char *data = getenv("QUERY_STRING");
r=sscanf(data,"number1=%d&number2=%d",&n1,&n2);

printf("Content-type: text/html\n\n");
printf("<html><head><title>CGI C Example</title></head>\n");
printf("<body><h1>CGI C Example</h1>\n");

if(r!=2)
{
printf("<p>Input data error\n");
}
else
{
printf("<p>%d + %d = %d<br><br>",n1,n2,n1+n2);
}
```

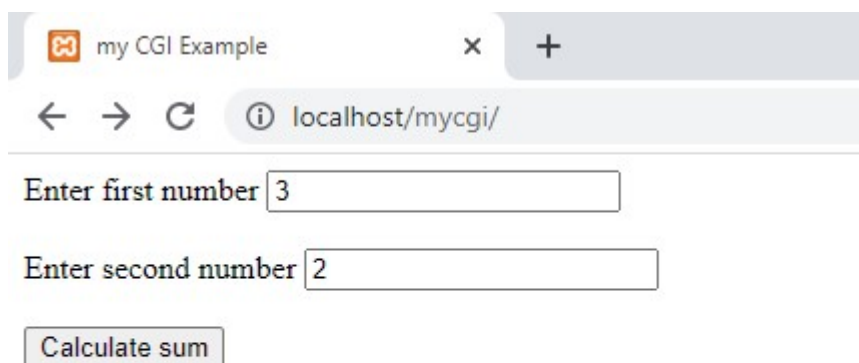
```
}  
  
//Vogliamo stampare le variabili d'ambiente per curiosita'  
while (*env)  
    printf("<br>%s", *env++);  
  
printf("</body></html>\n");  
  
return 0;  
}
```

Naturalmente il programma deve essere richiamato da un server e l'input non arriva normalmente da tastiera ma dalle variabili d'ambiente che possono essere lette dalla funzione `getenv()`. Inoltre si e' modificato la scrittura tramite `printf` per restituire un semplice pagina html con alcuni tag.

Poi creiamo un sito ed esempio: `C:\xampp\htdocs\mycgi`

con la seguente pagina

```
Index.html  
<html>  
<head>  
<title>my CGI Example</title>  
</head>  
<body>  
<form action="/cgi-bin/cgisum.exe" method="get">  
Enter first number <input type="text" name="number1"><br>  
Enter second number <input type="text" name="number2"><br>  
<input type="submit" value="Calculate sum">  
</form>  
</body>  
</html>
```



The screenshot shows a web browser window with the title "my CGI Example". The address bar displays "localhost/mycgi/". The page content includes two text input fields. The first field is labeled "Enter first number" and contains the value "3". The second field is labeled "Enter second number" and contains the value "2". Below the input fields is a button labeled "Calculate sum".

## CGI C Example

3 + 2 = 5

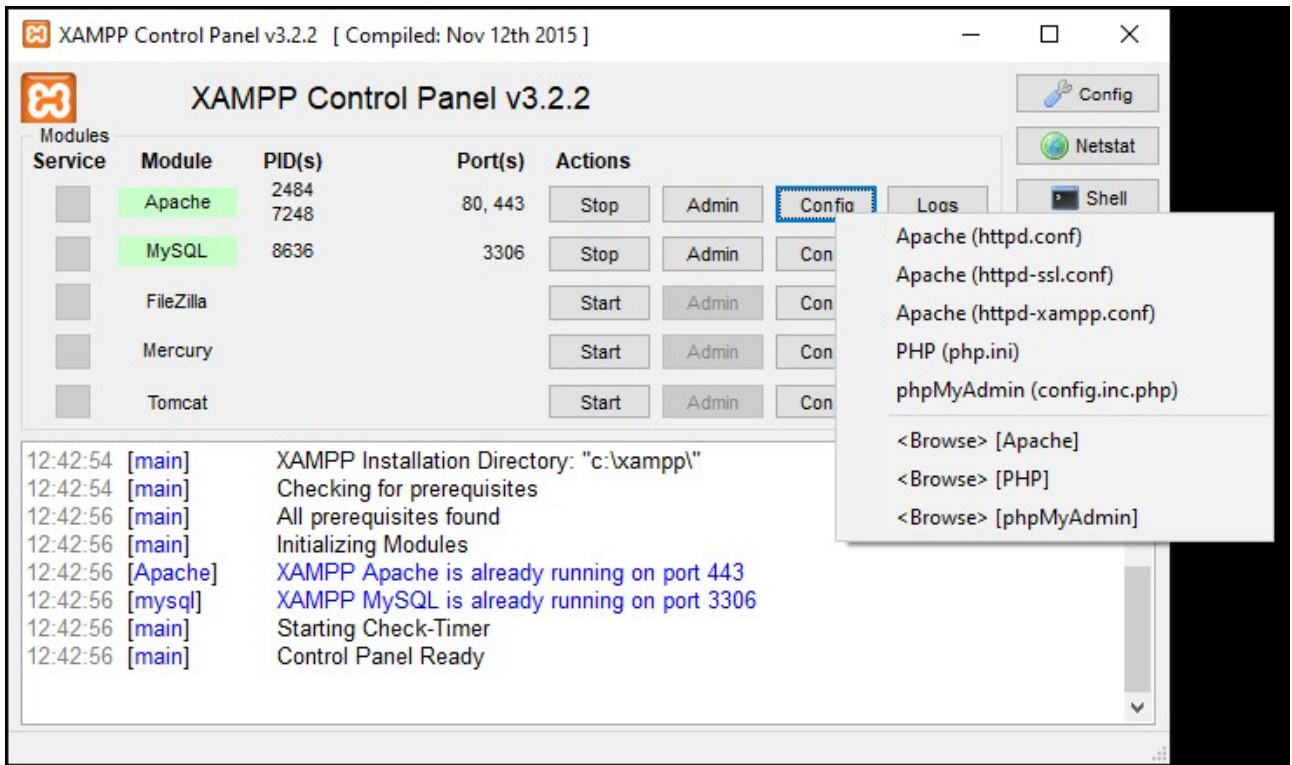
```
MIBDIRS=C:/xampp/php/extras/mibs
MYSQL_HOME=C:/xampp/mysql/bin
OPENSSL_CONF=C:/xampp/apache/bin/openssl.cnf
PHP_PEAR_SYSCONF_DIR=C:/xampp/php
PHPRC=C:/xampp/php
TMP=C:/xampp/tmp
HTTP_HOST=localhost
HTTP_CONNECTION=keep-alive
HTTP_SEC_CH-UA="Google Chrome";v="111", "Not(A:Brand";v="8", "Chromium";v="111"
HTTP_SEC_CH-UA-MOBILE=?
HTTP_SEC_CH-UA-PLATFORM="Windows"
HTTP_UPGRADE_INSECURE_REQUESTS=1
HTTP_USER_AGENT=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chron
HTTP_ACCEPT=text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,a
HTTP_SEC_FETCH_SITE=same-origin
HTTP_SEC_FETCH_MODE=navigate
HTTP_SEC_FETCH_USER=?1
HTTP_SEC_FETCH_DEST=document
HTTP_REFERER=http://localhost/mycgi/
HTTP_ACCEPT_ENCODING=gzip, deflate, br
HTTP_ACCEPT_LANGUAGE=it-IT,it;q=0.9,en-US;q=0.8,en;q=0.7
PATH=C:/Windows/system32;C:/Windows;C:/Windows/System32/Wbem;C:/Windows/System32/WindowsPowerShell/v
Server/Client SDK/ODBC/170/Tools/Binn/;C:/Users/FP/AppData/Local/Microsoft/Windows/Apps;C:/Users/FP/.dotnet/
SystemRoot=C:/Windows
COMSPEC=C:/Windows/system32/cmd.exe
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
WINDIR=C:/Windows
SERVER_SIGNATURE=
```

*Apache/2.4.28 (Win32) OpenSSL/1.0.2l PHP/7.1.10 Server at localhost Port 80*

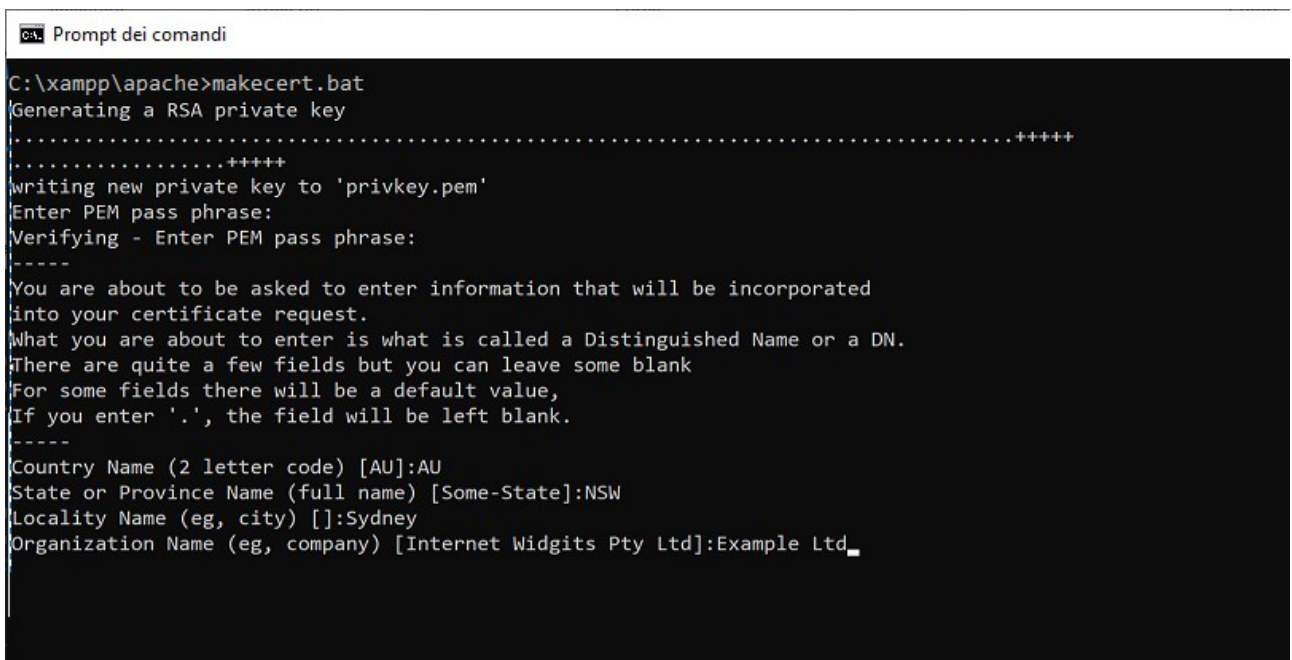
```
SERVER_SOFTWARE=Apache/2.4.28 (Win32) OpenSSL/1.0.2l PHP/7.1.10
SERVER_NAME=localhost
SERVER_ADDR=:1
SERVER_PORT=80
REMOTE_ADDR=:1
DOCUMENT_ROOT=C:/xampp/htdocs
REQUEST_SCHEME=http
CONTEXT_PREFIX=/cgi-bin/
CONTEXT_DOCUMENT_ROOT=C:/xampp/cgi-bin/
SERVER_ADMIN=postmaster@localhost
SCRIPT_FILENAME=C:/xampp/cgi-bin/cgisum.exe
REMOTE_PORT=51684
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=number1=3&number2=2
REQUEST_URI=/cgi-bin/cgisum.exe?number1=3&number2=2
SCRIPT_NAME=/cgi-bin/cgisum.exe
```

HTTPS





Per configurare Apache affinché risponda a richieste HTTPS, avviare il programma  
 c:\xampp\apache\makecert.bat



Una volta terminato vengono create i file:

il certificato:

C:\xampp\apache\conf\ssl.crt\server.crt

E la chiave privata:

C:\xampp\apache\conf\ssl.key\server.key



Il file che regola HTTPS o meglio SSL si trova in:

C:\xampp\apache\conf\extra\httpd-ssl.conf

Qui si trova la riga:

```
Listen 443
```

Che dice su quale porta apache ascolta le richieste https

Mentre la riga:

```
SSLCertificateFile "conf/ssl.crt/server.crt"
```

Indica il certificato utilizzato da apache per soddisfare le richieste https.

Infine:

.....

```
<VirtualHost _default_:443>
```

```
# General setup for the virtual host
```

```
DocumentRoot "C:/xampp/htdocs"
```

.....

Indica il percorso che apache proteggerà'.

Il certificato prodotto in precedenza con makecert e' autogenerato infatti possiamo verificarlo con il comando:

```
C:\xampp\apache\bin> openssl s_client -showcerts -connect localhost:443
```

```

No ALPN negotiated
SSL-Session:
  Protocol   : TLSv1.2
  Cipher     : ECDHE-RSA-AES256-GCM-SHA384
  Session-ID: C36E032C83ADC7EFB11E73E8A268B80746D4C4A35573F8A29831469D5FE875CF
  Session-ID-ctx:
  Master-Key: CD3491D9E94B82CD6784F05EE71C42E3DABB494F58AD26138338733CD20A192320494C78F24
  Key-Arg    : None
  PSK identity: None
  PSK identity hint: None
  SRP username: None
  TLS session ticket lifetime hint: 300 (seconds)
  TLS session ticket:
0000 - 7e fc 18 4b b5 87 4d 8d-5e 01 45 53 62 b4 94 b4 ~..K..M.^.ESb...
0010 - 5c 78 3d 4e 53 d2 4f 60-2d 3d df b2 42 9f b0 e6 \x=NS.O`-=..B...
0020 - 9a 43 59 60 6d fb 69 a0-30 60 d2 77 1e 93 06 7f .CY`m.i.0`.w....
0030 - a4 4a d1 25 10 1d 81 fd-e5 43 35 0d 33 1e 0b d4 .J.%.C5.3...
0040 - e5 3a 2a 78 b8 75 e3 2d-9e a7 68 59 13 f1 37 53 .:*x.u.-.hY..7S
0050 - 0c 9b 6c 2a fe 48 d5 b8-98 af 0e c4 97 3f 18 3e ..l*.H.....?>
0060 - a9 4e eb 2c d7 5d d4 cd-b6 e5 a1 88 cd 32 0b 1f .N.,.].....2..
0070 - e1 c8 4b 6b 10 a6 03 70-cc 2c d6 d9 98 2b 89 e8 ..Kk...p.,...+..
0080 - 3d 75 55 72 cb 9b 50 ea-be f6 7a 1c 79 d7 68 01 =uUr..P...z.y.h.
0090 - ba 0e 8d cc 04 2b 58 cb-93 74 45 02 e7 8a d6 52 .....+X..tE....R
00a0 - b3 39 a3 31 2e 5d 09 99-08 bd 49 f2 e7 ad 87 22 .9.1.]....I...."
00b0 - 45 8b 77 27 93 90 85 f6-d7 73 5c 3d e2 f0 eb 69 E.w'.....s\=...i

  Start Time: 1679489106
  Timeout    : 300 (sec)
  Verify return code: 18 (self signed certificate)
---
```

Dove nell'ultima riga dice che il certificato e' effettivamente autogenerato.

Ora se avvio la richiesta:

https://localhost

Ottengo:



## Your connection isn't private

Attackers might be trying to steal your information from **localhost** (for example, passwords, messages or credit cards).

NET::ERR\_CERT\_AUTHORITY\_INVALID

Hide advanced

Go back

This server couldn't prove that it's **localhost**; its security certificate is not trusted by your computer's operating system. This may be caused by a misconfiguration or an attacker intercepting your connection.

[Continue to localhost \(unsafe\)](#)

Se si vuole evitare che chrome non visualizzi il messaggio precedente:

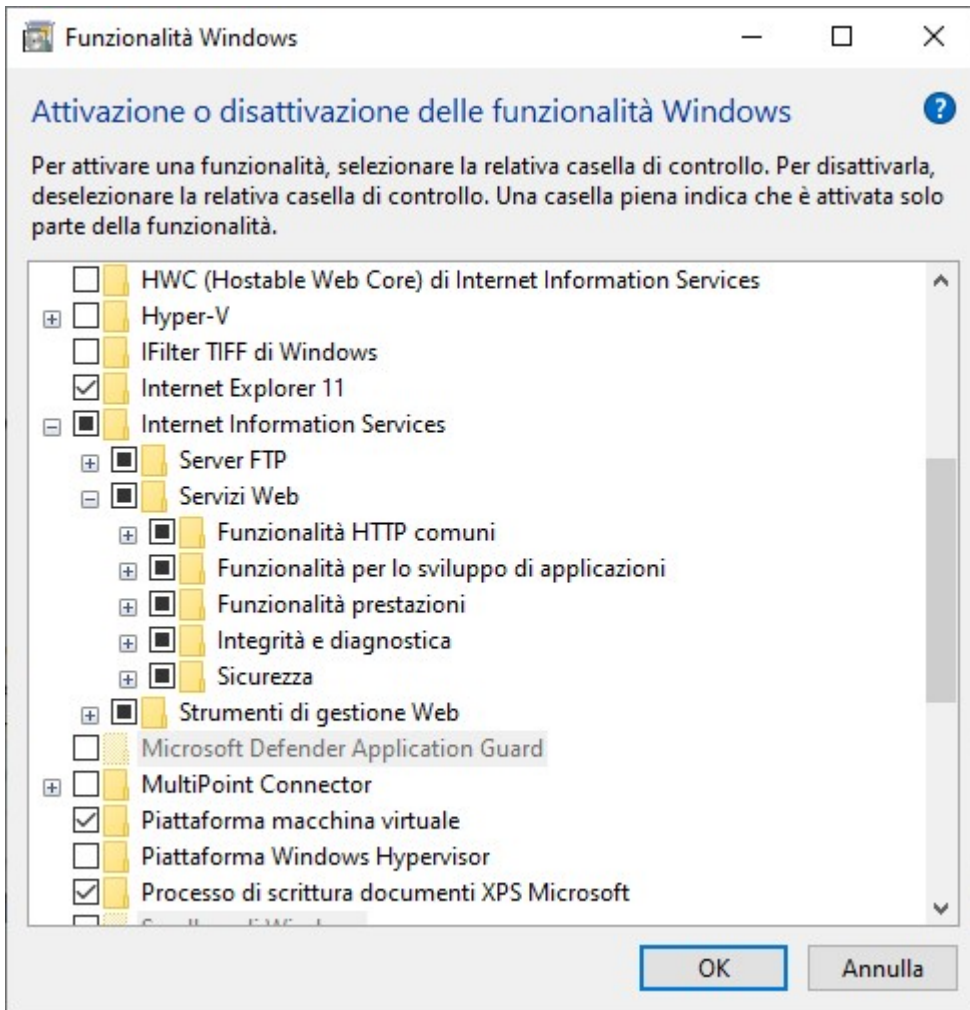
`chrome://flags/#allow-insecure-localhost`

e abilitare la voce:

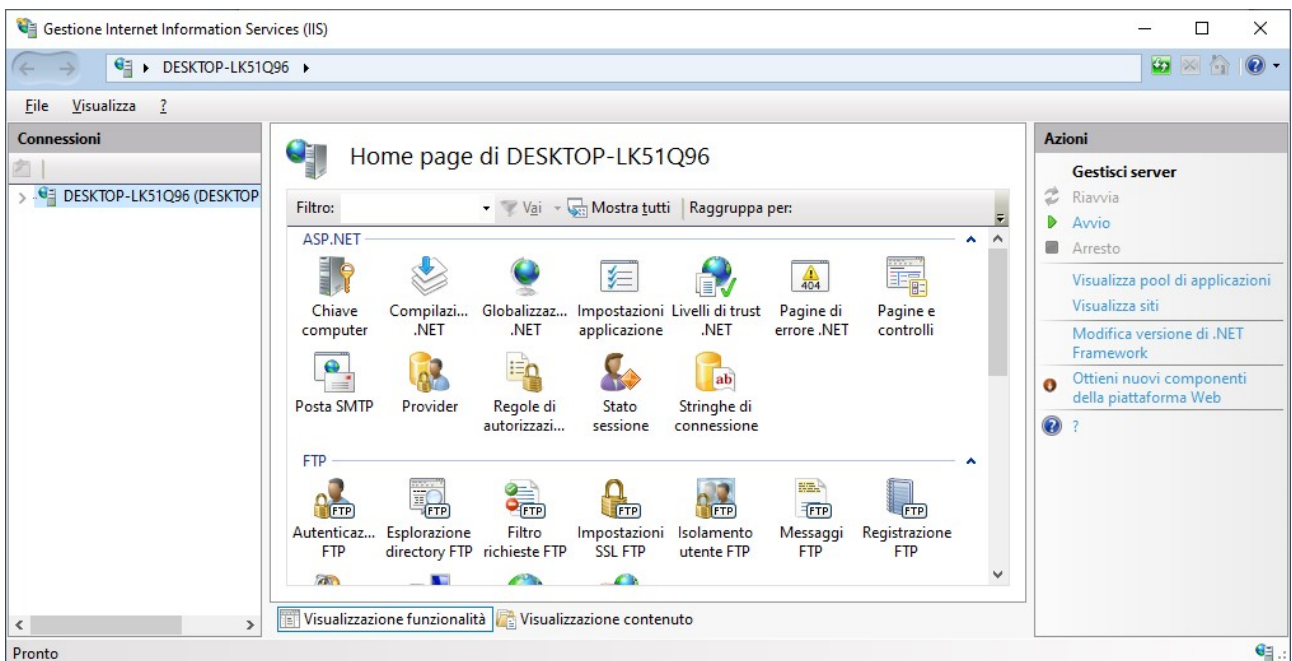
Allow invalid certificates for resources loaded from localhost

### **Il Web Server Microsoft Internet Information Services (IIS)[CENNI].**

Puo' essere installato, se si e' amministratori, tramite le "Funzionalita' di Windows" dal Pannello di controllo.



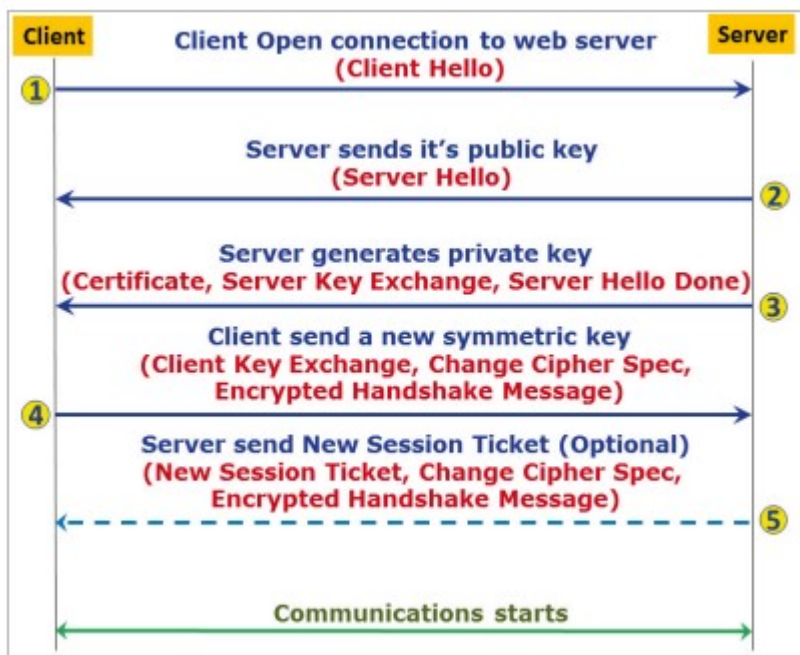
IIS si può gestire tramite l'applicazione Windows "Gestione Internet Information Services" avviabile dalla finestra cerca digitando IIS.



IIS usa il linguaggio lato server e' ASP.NET (i cui file di programma hanno estensione .aspx) basato su C#. Di seguito un frammento di file aspx:

```
<% @ Page Language="C#" %>
<%
foreach (string var in Request.ServerVariables)
{
    Response.Write(var + " " + Request[var] + "<br>");
}
%>
```

### IL PROTOCOLLO HTTPS (ssl/tls)



No.	Time	Source	Destination	Protocol	Length	Info
22	0.068116	192.168.1.52	192.168.1.57	TCP	54	443 → 55174 [FIN, ACK] Seq=1330 Ack=526 Win=130816 Len=0
23	0.068160	192.168.1.52	192.168.1.57	TCP	54	443 → 55172 [ACK] Seq=1331 Ack=526 Win=130816 Len=0
24	0.072485	192.168.1.57	192.168.1.52	TCP	54	55172 → 443 [ACK] Seq=526 Ack=1331 Win=64128 Len=0
25	0.072485	192.168.1.57	192.168.1.52	TCP	54	55174 → 443 [ACK] Seq=526 Ack=1331 Win=64128 Len=0
26	0.073817	192.168.1.57	192.168.1.52	TCP	54	55176 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0
27	0.073817	192.168.1.57	192.168.1.52	1	571	TLSv1.2 Client Hello
28	0.084807	192.168.1.52	192.168.1.57	2	1383	TLSv1.2 Server Hello, Certificate, Server Key Exchange, Server Hello Done
29	0.097178	192.168.1.57	192.168.1.52	TCP	54	55176 → 443 [ACK] Seq=518 Ack=1330 Win=64128 Len=0
30	0.097178	192.168.1.57	192.168.1.52	3	180	TLSv1.2 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
31	0.097178	192.168.1.57	192.168.1.52	TLSv1.2	777	Application Data
32	0.097379	192.168.1.52	192.168.1.57	TCP	54	443 → 55176 [ACK] Seq=1330 Ack=1367 Win=129792 Len=0
33	0.098747	192.168.1.52	192.168.1.57	4	312	TLSv1.2 New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
34	0.107779	192.168.1.52	192.168.1.57	TLSv1.2	527	Application Data
35	0.114912	192.168.1.57	192.168.1.52	TCP	54	55176 → 443 [ACK] Seq=1367 Ack=2061 Win=64128 Len=0
36	0.278686	192.168.1.57	192.168.1.52	TLSv1.2	676	Application Data
37	0.281028	192.168.1.52	192.168.1.57	TLSv1.2	1514	Application Data
38	0.281028	192.168.1.52	192.168.1.57	TCP	1514	443 → 55176 [ACK] Seq=3521 Ack=1080 Win=131328 Len=1460 [TCP segment of a reassembled

> Frame 27: 571 bytes on wire (4568 bits), 571 bytes captured (4568 bits) on interface \Device\NPF\_{2634871C-619E-44FD-810F-F3EE7243C6C9}, id 0  
 > Ethernet II, Src: LiteonTe\_0b:c9:31 (20:68:9d:0b:c9:31), Dst: LiteonTe\_8f:e2:59 (3c:a0:67:8f:e2:59)  
 > Internet Protocol Version 4, Src: 192.168.1.57, Dst: 192.168.1.52  
 v Transmission Control Protocol, Src Port: 55176, Dst Port: 443, Seq: 1, Ack: 1, Len: 517  
 Source Port: 55176  
 Destination Port: 443  
 [Stream index: 2]

1 In packet 160, the client sends a Client Hello message that starts the negotiation

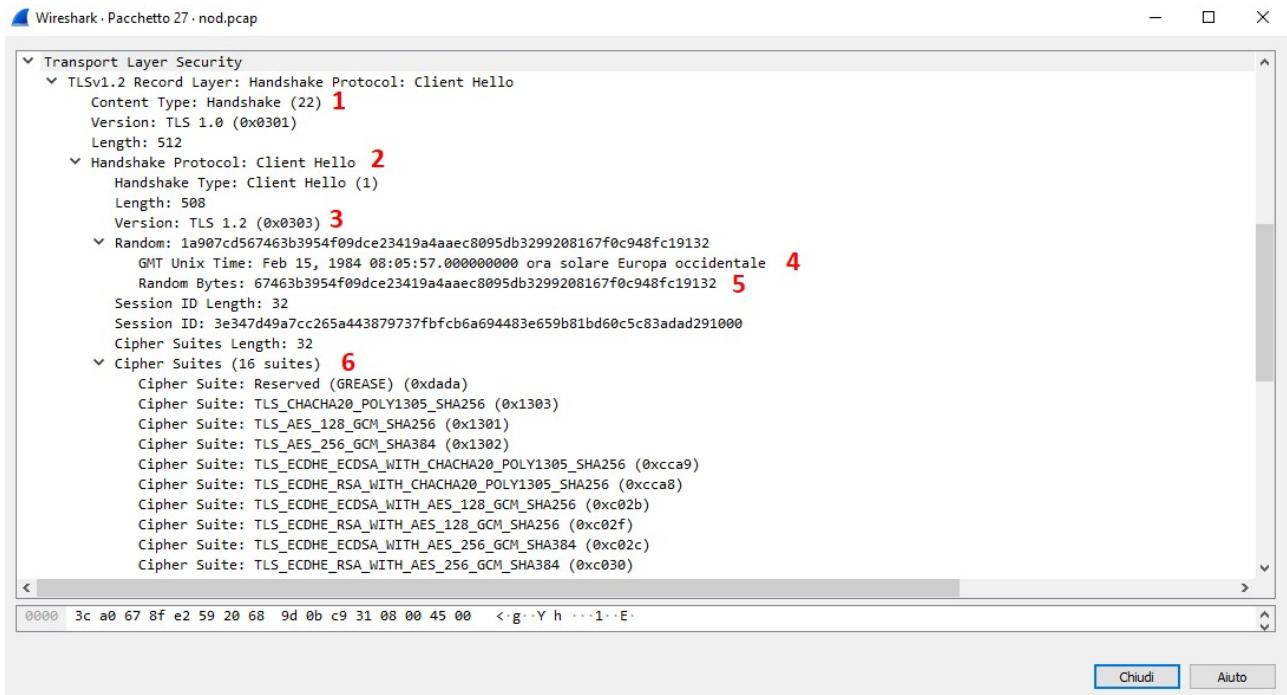


2 The server answers with a Server Hello message E The server sends a certificate to the client.

3 The client takes the certificate and generates a premaster key.

4 The server creates the master key, and the conversation begins

Nel passaggio 1, il pacchetto 27 è un messaggio Client Hello che è il primo pacchetto nell'handshake TLS .  
Alcuni dei parametri sono mostrati nello screenshot seguente:



1 mostra che il contenuto del pacchetto è un handshake (ssl.record.content\_type == 22).

2 mostra che il pacchetto è un messaggio Client Hello inviato dal client al server web. Questo messaggio avvia l'handshake.

3 mostra la versione SSL e TLS più alta supportata da il cliente.

4 mostra l'ora del client che verrà utilizzata nella chiave processo di generazione.

5 mostra i dati casuali generati dal client da utilizzare nel processo di generazione delle chiavi.

6 mostra le cifre supportate dal client. Le cifre sono elencati in ordine di preferenza.

7 mostra i metodi di compressione dei dati supportati dal cliente.

Come mostrato nella figura seguente, il pacchetto 28 è un messaggio Server Hello, che include i seguenti dettagli:



```
Wireshark · Pacchetto 28 · nod.pcap
> Frame 28: 1383 bytes on wire (11064 bits), 1383 bytes captured (11064 bits) on interface \Device\NPF_{2634871C-619E-44FD-810F-F3EE7243C6C9}, id 0
> Ethernet II, Src: LiteonTe_8f:e2:59 (3c:a0:67:8f:e2:59), Dst: LiteonTe_0b:c9:31 (20:68:9d:0b:c9:31)
> Internet Protocol Version 4, Src: 192.168.1.52, Dst: 192.168.1.57
> Transmission Control Protocol, Src Port: 443, Dst Port: 55176, Seq: 1, Ack: 518, Len: 1329
▼ Transport Layer Security
  ▼ TLSv1.2 Record Layer: Handshake Protocol: Server Hello
    Content Type: Handshake (22) 1
    Version: TLS 1.2 (0x0303) 2
    Length: 76
    ▼ Handshake Protocol: Server Hello
      Handshake Type: Server Hello (2) 3
      Length: 72
      Version: TLS 1.2 (0x0303)
      ▼ Random: 9d8a0f3a6e9da54fc08f82acdba5811d0a3dcde33365d242331d390a7da0da5f
        GMT Unix Time: Oct 3, 2053 04:19:38.000000000 ora legale Europa occidentale 4
        Random Bytes: 6e9da54fc08f82acdba5811d0a3dcde33365d242331d390a7da0da5f 5
      Session ID Length: 0
      Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030) 6
      Compression Method: null (0) 7
      Extensions Length: 32
      > Extension: renegotiation_info (len=1)
      > Extension: ec_point_formats (len=4)
      > Extension: session_ticket (len=0)
      > Extension: application_layer_protocol_negotiation (len=11)
        [JA3S Fullstring: 771,49200,65281-11-35-16]
        [JA3S: 2b33c1374db4ddf06942f92373c0b54b]
    ▼ TLSv1.2 Record Layer: Handshake Protocol: Certificate
      Content Type: Handshake (22)
      Version: TLS 1.2 (0x0303)
      Length: 896
      ▼ Handshake Protocol: Certificate
```

1 mostra che il contenuto del pacchetto è un handshake (ssl.record.tipo\_contenuto == 22).

2 mostra la versione TLS che verrà utilizzata in questa sessione.

3 mostra che il pacchetto è un messaggio Server Hello inviato da il server al client.

4 mostra l'ora del server utilizzata nel processo di generazione della chiave.

5 mostra i dati casuali generati dal server per utilizzare nel processo di generazione delle chiavi.

6 mostra la suite di crittografia da utilizzare in questa conversazione. Viene scelto dall'elenco delle cifre inviate dal client.

7 mostra il metodo di compressione dei dati che verrà utilizzato la sessione.

Il pacchetto successivo è la risposta del server che emette un certificato:

No.	Time	Source	Destination	Protocol	Length	Info
22	0.068116	192.168.1.52	192.168.1.57	TCP	54	443 → 55174 [FIN, ACK] Seq=1330 Ack=526 Win=130816 Len=0
23	0.068160	192.168.1.52	192.168.1.57	TCP	54	443 → 55172 [ACK] Seq=1331 Ack=526 Win=130816 Len=0
24	0.072485	192.168.1.57	192.168.1.52	TCP	54	55172 → 443 [ACK] Seq=526 Ack=1331 Win=64128 Len=0
25	0.072485	192.168.1.57	192.168.1.52	TCP	54	55174 → 443 [ACK] Seq=526 Ack=1331 Win=64128 Len=0
26	0.073817	192.168.1.57	192.168.1.52	TCP	54	55176 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0
27	0.073817	192.168.1.57	192.168.1.52	TLSv1.2	571	Client Hello
28	0.084807	192.168.1.52	192.168.1.57	TLSv1.2	1383	Server Hello, Certificate, Server Key Exchange, Server Hello Done
29	0.097178	192.168.1.57	192.168.1.52	TCP	54	55176 → 443 [ACK] Seq=518 Ack=1330 Win=64128 Len=0
30	0.097178	192.168.1.57	192.168.1.52	TLSv1.2	180	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
31	0.097178	192.168.1.57	192.168.1.52	TLSv1.2	777	Application Data
32	0.097379	192.168.1.52	192.168.1.57	TCP	54	443 → 55176 [ACK] Seq=1330 Ack=1367 Win=129792 Len=0
33	0.098747	192.168.1.52	192.168.1.57	TLSv1.2	312	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
34	0.107779	192.168.1.52	192.168.1.57	TLSv1.2	527	Application Data
35	0.114912	192.168.1.57	192.168.1.52	TCP	54	55176 → 443 [ACK] Seq=1367 Ack=2061 Win=64128 Len=0
36	0.278686	192.168.1.57	192.168.1.52	TLSv1.2	676	Application Data
37	0.281028	192.168.1.52	192.168.1.57	TLSv1.2	1514	Application Data
38	0.281028	192.168.1.52	192.168.1.57	TCP	1514	443 → 55176 [ACK] Seq=3521 Ack=1980 Win=131328 Len=1460 [TCP segment of a reassembled...

> Frame 28: 1383 bytes on wire (11064 bits), 1383 bytes captured (11064 bits) on interface \Device\NPF\_{2634871C-619E-44FD-810F-F3EE7243C6C9}, id 0  
 > Ethernet II, Src: LiteonTe\_0f:e2:59 (3c:a0:67:8f:e2:59), Dst: LiteonTe\_0b:c9:31 (20:68:9d:0b:c9:31)  
 > Internet Protocol Version 4, Src: 192.168.1.52, Dst: 192.168.1.57  
 > Transmission Control Protocol, Src Port: 443, Dst Port: 55176, Seq: 1, Ack: 518, Len: 1329  
 > Transport Layer Security  
   > TLSv1.2 Record Layer: Handshake Protocol: Server Hello  
   > TLSv1.2 Record Layer: Handshake Protocol: Certificate **1**  
   > TLSv1.2 Record Layer: Handshake Protocol: Server Key Exchange **2**  
   > TLSv1.2 Record Layer: Handshake Protocol: Server Hello Done **3**

## Dettagli:

Wireshark · Pacchetto 28 · nod.pcap

- TLSv1.2 Record Layer: Handshake Protocol: Certificate
  - Content Type: Handshake (22)
  - Version: TLS 1.2 (0x0303)
  - Length: 896
- Handshake Protocol: Certificate
  - Handshake Type: Certificate (11)
  - Length: 892
  - Certificates Length: 889
  - Certificates (889 bytes)
    - Certificate Length: 886
      - Certificate: 308203723082025a020900f80ce6973b19a94d300d06092a864886f70d01010b0500307b... (pkcs-9-at-emailAddress=my@acme.it,id-at-commonN)
        - signedCertificate
        - algorithmIdentifier (sha256WithRSAEncryption)
        - Padding: 0
        - encrypted: 60ae0a07012971bc26d8b859e49174dfbbd3ce06a11a73b8d8ba6ec515ba0d9c59189f24...
- TLSv1.2 Record Layer: Handshake Protocol: Server Key Exchange
  - Content Type: Handshake (22)
  - Version: TLS 1.2 (0x0303)
  - Length: 333
  - Handshake Protocol: Server Key Exchange
    - Handshake Type: Server Key Exchange (12)
    - Length: 329
      - EC Diffie-Hellman Server Params
- TLSv1.2 Record Layer: Handshake Protocol: Server Hello Done
  - Content Type: Handshake (22)
  - Version: TLS 1.2 (0x0303)
  - Length: 4
  - Handshake Protocol: Server Hello Done
    - Handshake Type: Server Hello Done (14)
    - Length: 0

0000 20 68 9d 0b c9 31 3c a0 67 8f e2 59 08 00 45 00 h...1< g..Y..E

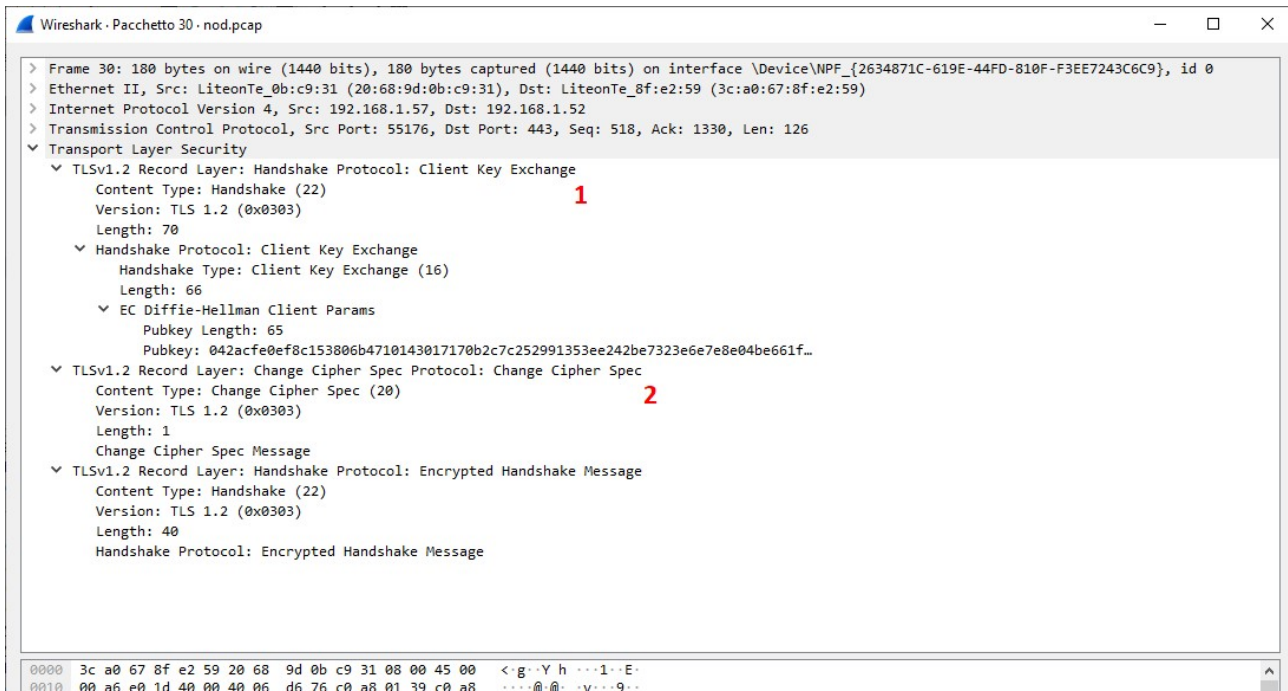
Chiudi    Aiuto

1 indica che il server invia il comando Certificate, che include il certificato del server. Navigando nel ramo si vedr' l'emittente del certificato, il tempo di validità, algoritmo e altri dati.

2 mostra che il server invia il comando Server Key Exchange (solitamente Difie-Hellman), inclusi i parametri richiesti (chiave pubblica, firma e così via).

3 mostra che il server invia il messaggio Server Hello Done. Questo comando indica che il server ha completato questa fase dell'handshake SSL. Il passo successivo è l'autenticazione del client.

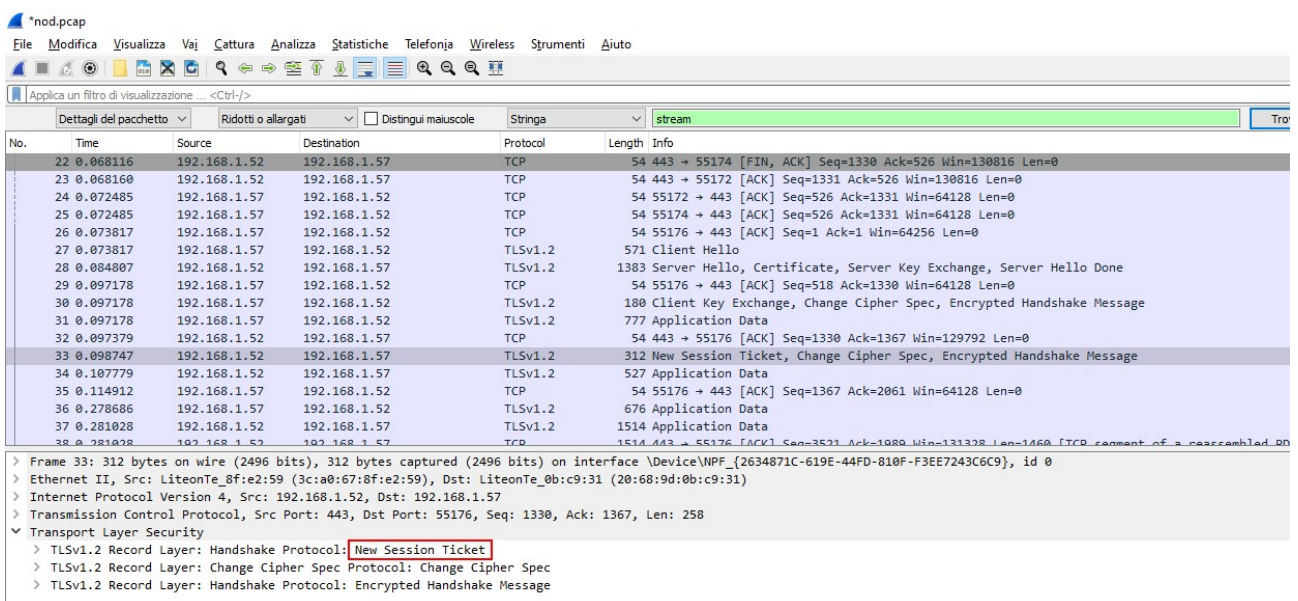
Il pacchetto successivo è la risposta emessa dal server cioè' un certificato.



1 mostra che il client invia il comando Client Key Exchange. Questo comando contiene il segreto premaster creato dal client ed è stato quindi crittografato utilizzando la chiave pubblica del server. Nella crittografia simmetrica le chiavi vengono generate dal client e dal server, in base ai dati scambiati nel file messaggi di saluto client e server.

2 mostra che il client invia la notifica Change Cipher Spec al server. Questo viene fatto per indicare che il cliente utilizzerà le nuove chiavi di sessione per l'hashing e la crittografia.

L'ultimo passaggio è quando il server invia un nuovo ticket di sessione al client come riportato nella figura seguente:



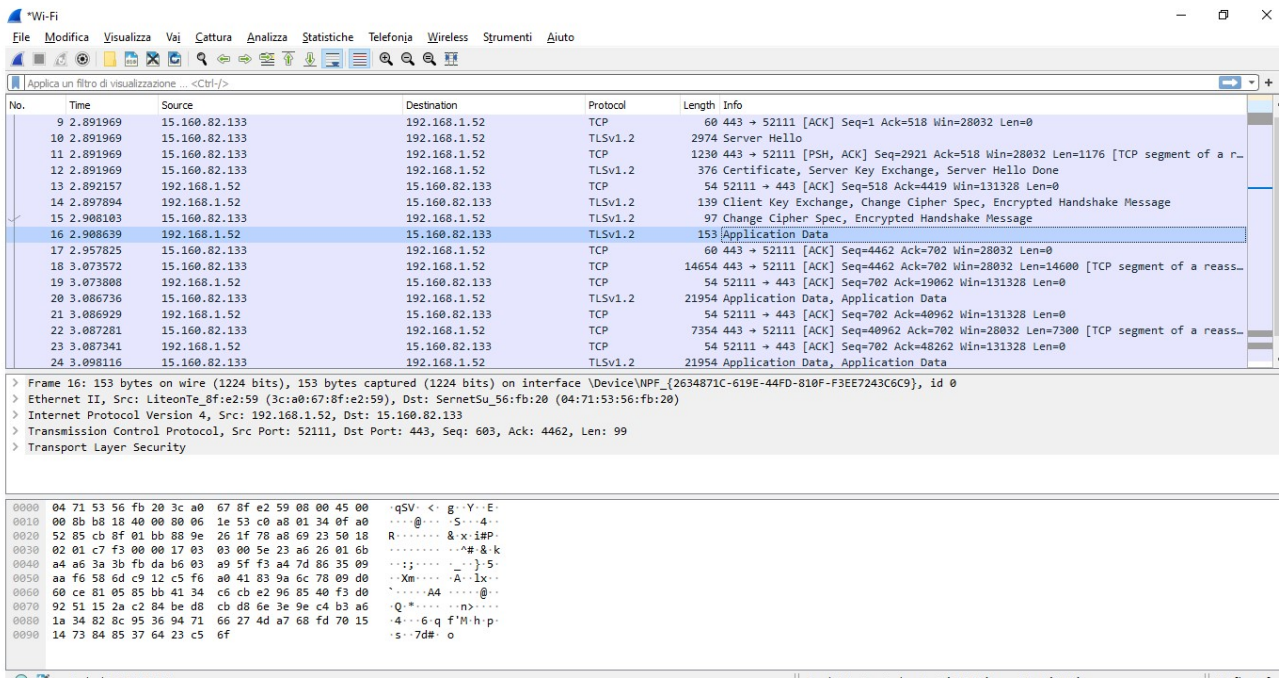
## DECODIFICARE TLS CON WIRESHARK



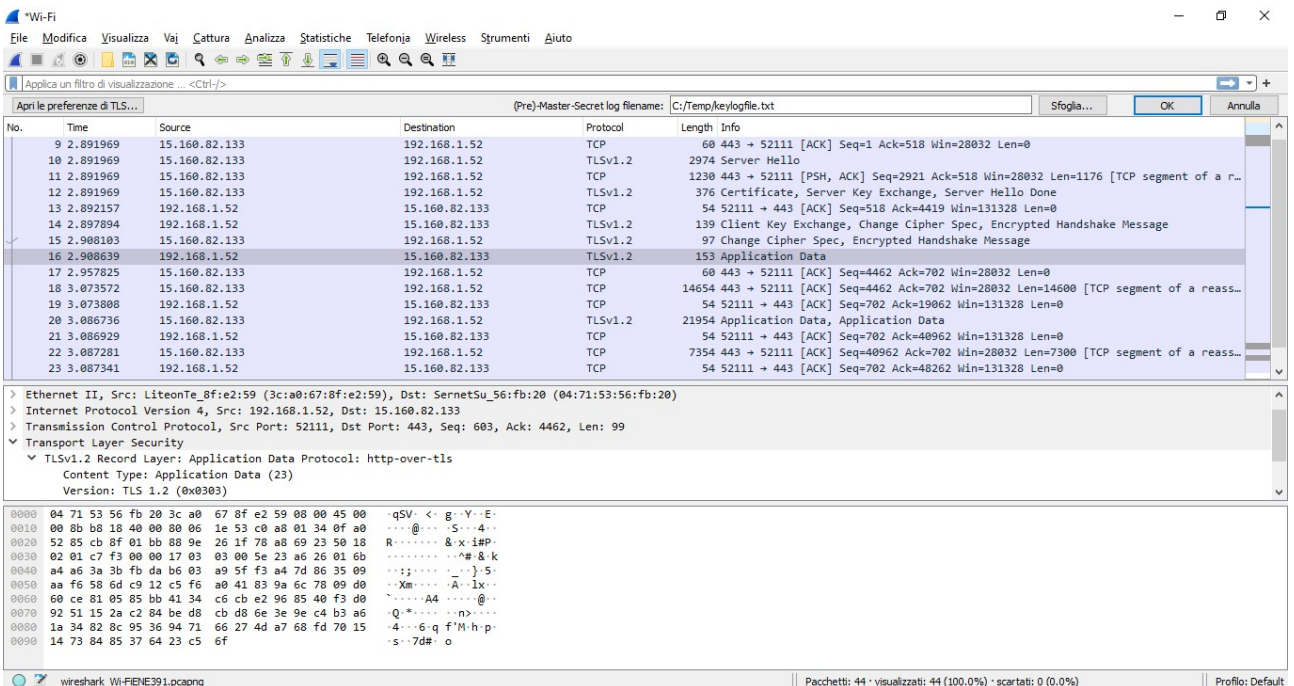
Impostiamo prima di tutto la variabile d'ambiente SSLKEYLOGFILE su un certo file a nostra scelta, ad esempio:

```
C:\>set SSLKEYLOGFILE=C:\temp\keylogfile.txt
```

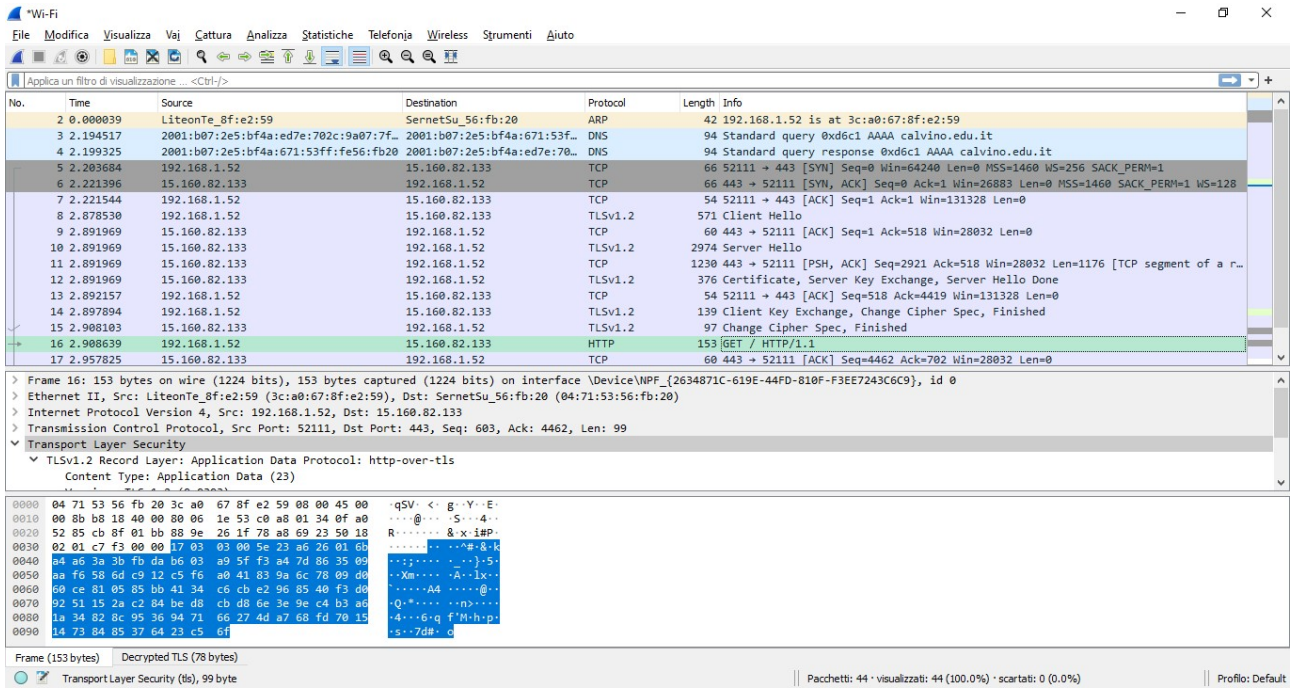
Scegliamo un URL che risponda al protocollo https come ad esempio: <https://calvino.edu.it> (possiamo anche utilizzare `curl -s https://calvino.edu.it`)



Si nota che il pacchetto 16 contiene i dati criptati ovviamente. Ma ora clicchiamo sempre sul pacchetto 16 e alla voce "Transport Layer Security" del pannello intermedio clicchiamo con il tasto destro del mouse e scegliamo "Preferenze del Protocollo" -> "(Pre)-Master-Secret log filename ...":



Inseriamo il path del file precedentemente scelto nella casella di testo in alto a destra: "Pre-Master-Secret log filename" e diamo OK



Il pacchetto 16 ora appare in chiaro come evidenziato anche dal dettaglio seguente:

