Práctica 3: HTTP + TLS

1. Introducción

Se denomina genéricamente servidor Web seguro a un servidor Web con soporte del protocolo HTTPS (HTTP por encima de SSL).

Nuestro objetivo será construir un servidor Web seguro a partir del servidor Web apache (http://www.apache.org). Apache es el servidor Web más utilizado en la red, en torno al 60% de los sitios Web utilizan apache. Hay dos formas de añadir soporte de HTTPS a apache:

- Utilizando apache_ssl.
- Utilizando mod_ssl.

Ambas formas proporcionan una interfaz al servidor Web para utilizar TLS y se basan en una implementación de dicho protocolo proporcionada por el paquete OpenSSL. Independientemente de si utilizamos una u otra estaremos obligados a instalar dicho paquete que podemos encontrar en http://www.openssl.org. Nosotros utilizaremos mos_ssl que no es más que un módulo de los disponibles para apache (como mos_php para poder utilizar PHP, por ejemplo) y que se configura mediante una serie de directivas que tendremos que incluir en la configuración del servidor.

2. Descipción

Los pasos a seguir son los siguientes:

- Instalar el servidor Web Apache.
- Instalar OpenSSL.
- Generar los certificados del servidor.
- Configurar el mod_ssl (Crear un Host Virtual).
- Generar los certificados del cliente.
- Comprobar el buen funcionamiento del sistema.

2.1 Configuración del servidor

Los pasos a seguir de cara a la instalación del servidor seguro son:

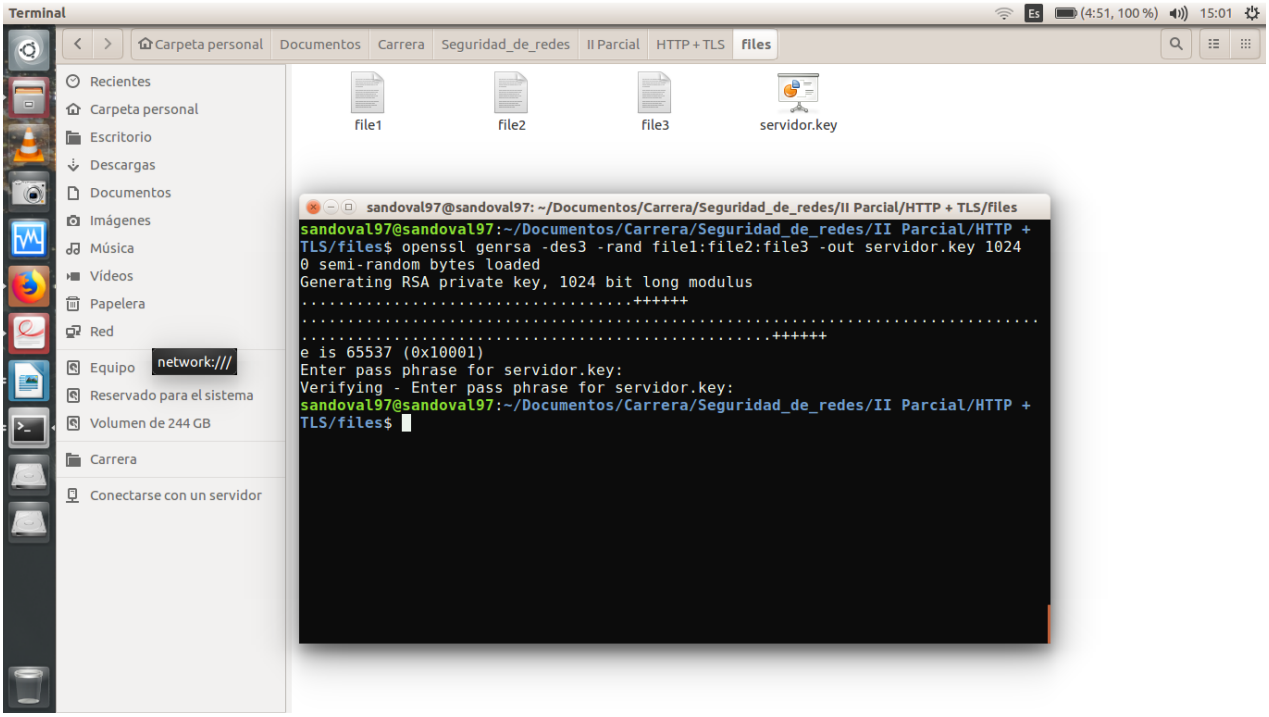
- Generación del certificado del servidor.
- Instalación y configuración del certificado del servidor.
- Configuración de apache.

2.2 Generación del certificado del servidor.

Generaremos un certificado propio. Este paso es uno de los más importantes y más sensibles a error. Vamos a generar el certificado que identificará al servidor seguro. Cuando solicitemos la creación de un certificado el programa (en este caso lo vamos a hacer con openssl) nos va a solicitar una serie de datos que son los correspondientes a los campos que tiene un certificado, junto con una clave para proteger el mismo. Aunque todos los campos son importantes puesto que dan la información que los clientes tendrán de nuestro servidor hay que ser especialmente cuidadosos con el campo CN. Cuando se nos solicite este campo hay que introducir no nuestro nombre, o el nombre de la compañía sino el FQDN (fully qualifield domain name) del servidor tal y como aparece en el DNS o en el archivo de resolución de nombres. Esto tiene que ser correcto porque los clientes comprobarán que el CN del certificado concuerde con la dirección que está siendo accedida.

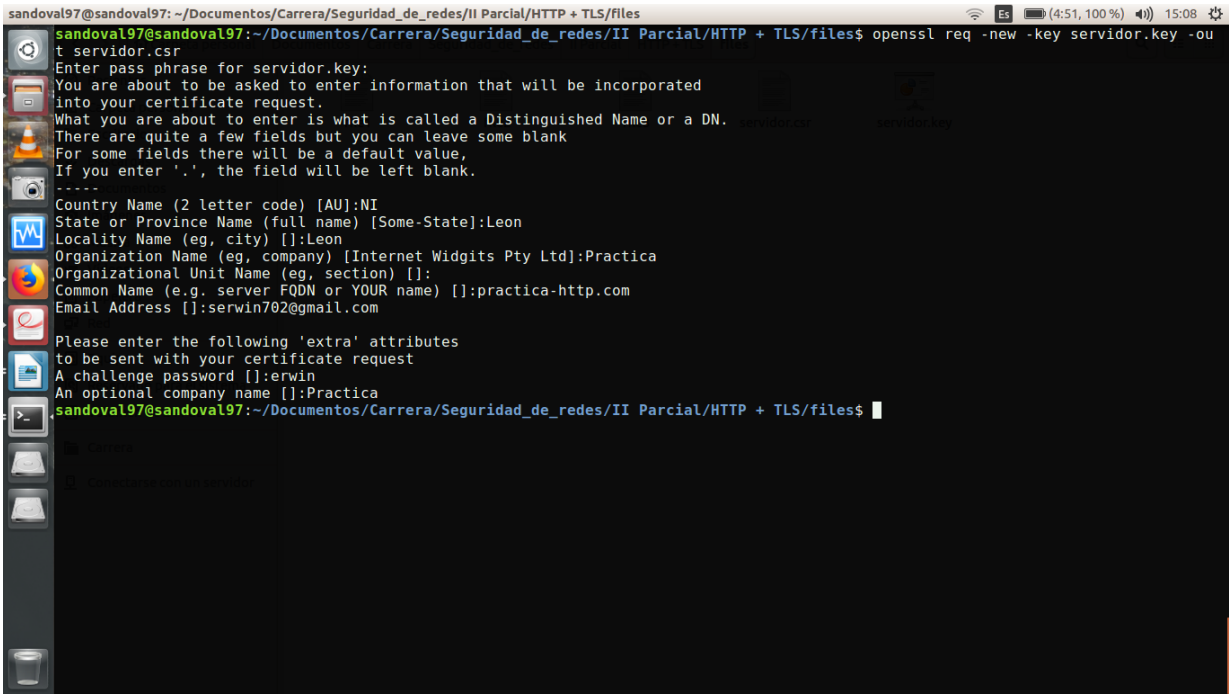
Primero vamos a generar el par de llaves utilizando el siguiente comando:
 # openssl genrsa –des3 –rand file1:flie2:file3 –out servidor.key 1024

lo que hace este comando es generar un par de claves usando el sistema criptografico RSA y el tipo de cifrado -des3, con el parametro -rand especificamos que cargue el archivo (o los archivos en el directorio) -out es el nombre que tendra nuestro archivo de salida y le decimos que tendra una longitud no menos de 1024 bits.



Bueno hasta este paso ya tenemos nuestro par de llaves nos falta generar el
Certificado:
 # openssl req –new –key servidor.key –out servidor.csr

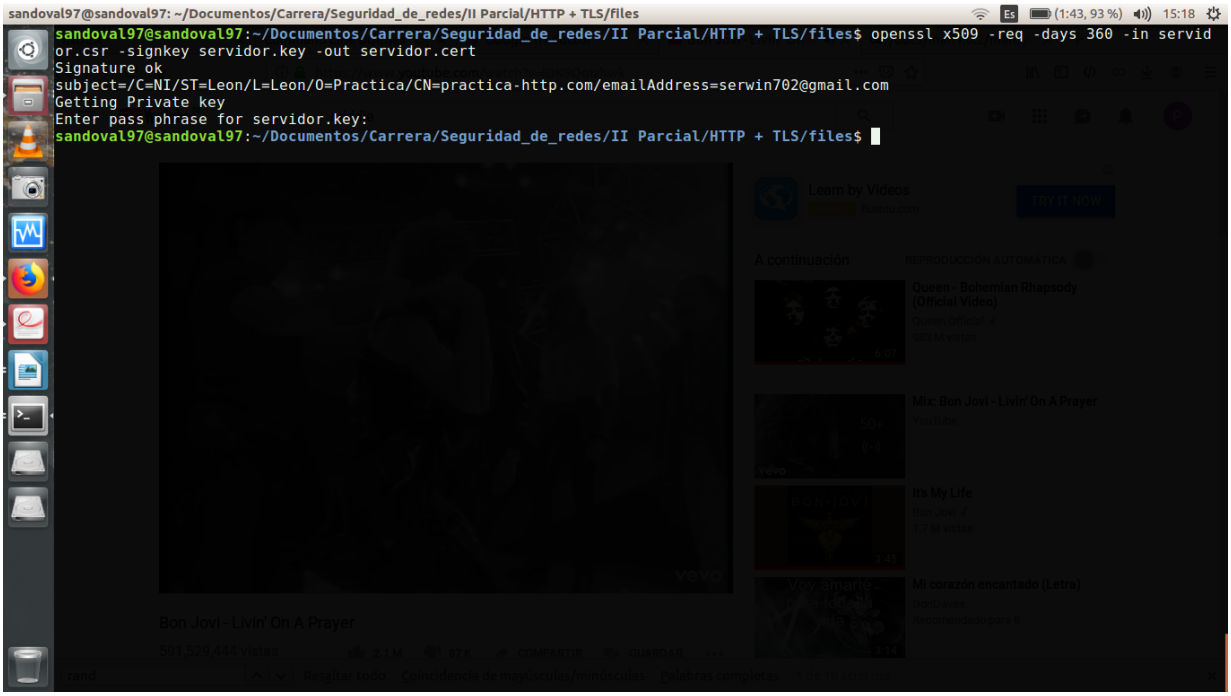
Con este comando decimos que queremos generar un certificado usando la opcion Certificate Request (req) y el parametro -new, con -key especificamos la clave con la que sera creado y -out para especificar el nombre de la salida de dicho comando.



El certificado está guardado en servidor.csr, bueno ahora vamos a crear un certificado firmado por nosotros mismos esto significa que podemos ser tanto el destinatario como el usuario del certificado. Aunque esto no es muy usual en las aplicaciones comerciales, si nos servirá para probar nuestra instalación.

 # openssl x509 –req –days 360 –in servidor.csr –signkey servidor.key –out servidor.cert.

el parametro x509 podemos mostrar información de certificados, convertir certificados en varios formularios y firmar solicitudes de certificados, al igual que el comando anterior usamos el parametro -req para esepecificar con -days los dias en que el certificado sera valido, especificamos el archivo de entrada que sera el certificado creado anteriormente y especificamos la clave con la que sera firmado y como con el comando anterior especificamos el nombre del archivo de salida.



2.3 Configuración.

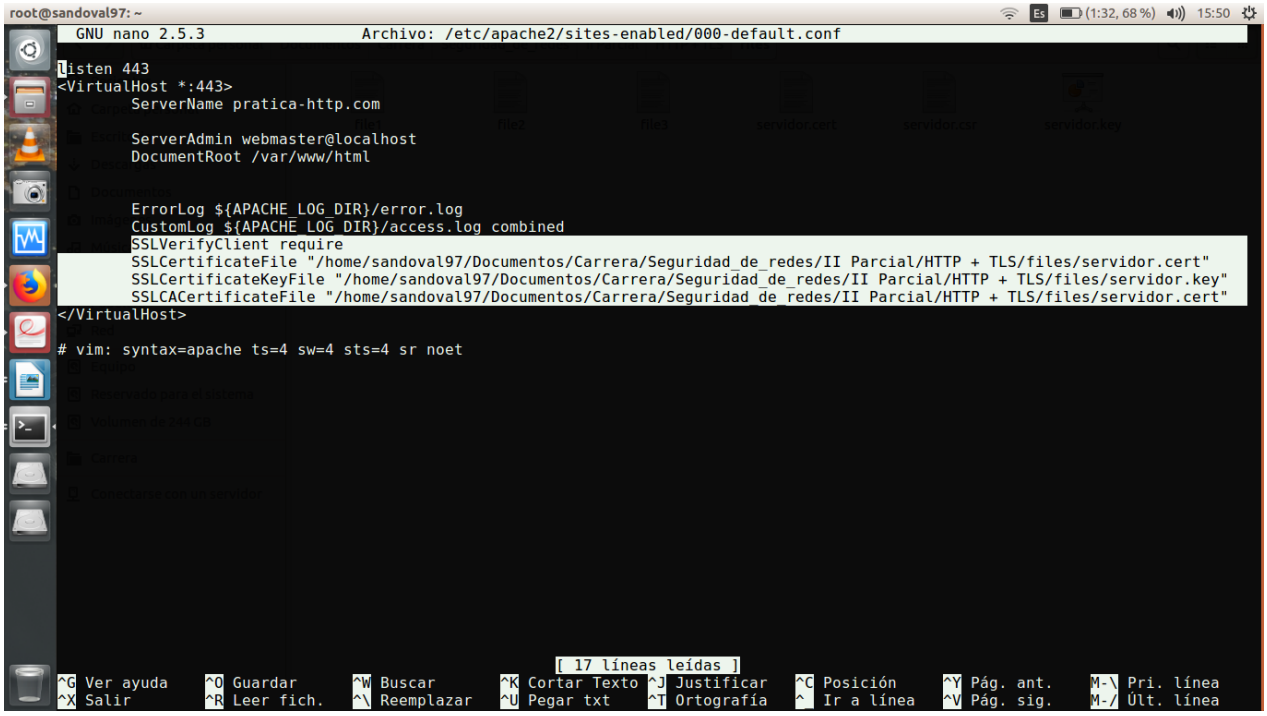
Una vez que tenemos instalado nuestro servidor apache nos queda la tarea de configurarlo para adecuarlo a nuestras necesidades concretas. En principio, sería interesante que sirviera páginas, en nuestro caso, tanto de forma segura como no segura, es decir, que pudiéramos descargar códigos utilizando tanto el protocolo HTTP como el protocolo HTTPS. Necesitamos, por tanto, que el servidor escuche en dos puertos distintos, el 80 para http y, en principio, el 443 para HTTPS. Para ello tendremos que crear un host virtual.

Por último otro aspecto interesante está relacionado con la autenticación de los participantes de la comunicación. Tal y como viene configurado por defecto el servidor apache seguro no requiere autenticación de cliente. El servidor envía su certificado al cliente para mostrar que él es quién realmente dice ser, pero no solicita al cliente que hago lo propio. Si obligamos a que el cliente nos envíe su certificado aseguraremos la autenticidad de las peticiones.

Todo esto lo tenemos que configurar en el servidor. La autenticación de cliente se regula mediante la variable SSLVerifyClient del fichero de configuración. Si ponemos esta variable a 0 (none) el servidor no solicitará al cliente que envíe su certificado durante la fase de negociación del protocolo TLS. Si la ponemos a 1 (optional) admitirá peticiones con o sin autentificación de cliente. Finalmente, un valor 2 (require) indicará al cliente que es obligatorio que presente su certificado.

Comenzamos definiendo el puerto donde estará ubicado así como activando el uso de SSL en el host. con SSLChiperSuite establecemos la lista de algoritmos criptográficos que se ofrecerán al cliente durante la negociación así como la prioridad de los mismos. Esto nos puede servir para obligar al cliente a usar determinado algoritmo para autenticarse, por ejemplo. Después viene la parte donde se indica donde reside el certificado (SSLCertificateFile) y la clave o claves privadas del servidor (SSLCertificateKeyFile).

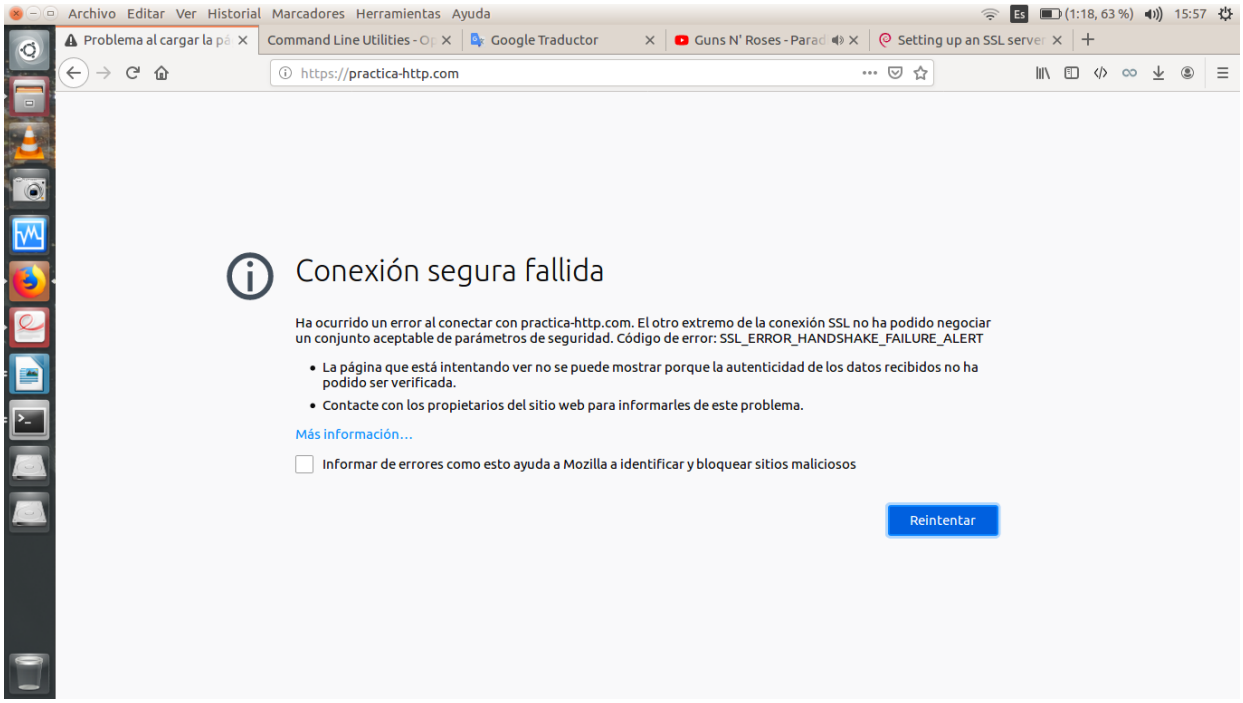
Posteriormente tenemos que indicar (SSLCACertificateFile) donde reside el certificado de la autoridad de certificación que permite probar los certificados que nos envían los clientes. En nuestro caso tendremos que poner aquí el certificado de la autoridad de certificación de nuestro dominio. Por último, tenemos los parámetros que ya hemos comentado acerca de la autenticación de cliente. Faltaría comentar la variable SSLVerifyDepth que especifica la longitud máxima de la cadena de verificación del certificado que presenta el cliente.



3. Pruebas

Para comprobar el correcto funcionamiento de nuestro servidor apache hemos de comprobar que permite el acceso al puerto 80 (HTTP) y al puerto 443 (HTTPS) con y sin autentificación del cliente. En primer lugar hemos de arrancar el servidor y comprobar que está escuchando en los puertos 80 y 443. Una vez que vemos que el servidor está ejecutándose probamos que se puede acceder al mismo utilizando un explorador Web como por ejemplo Firefox.

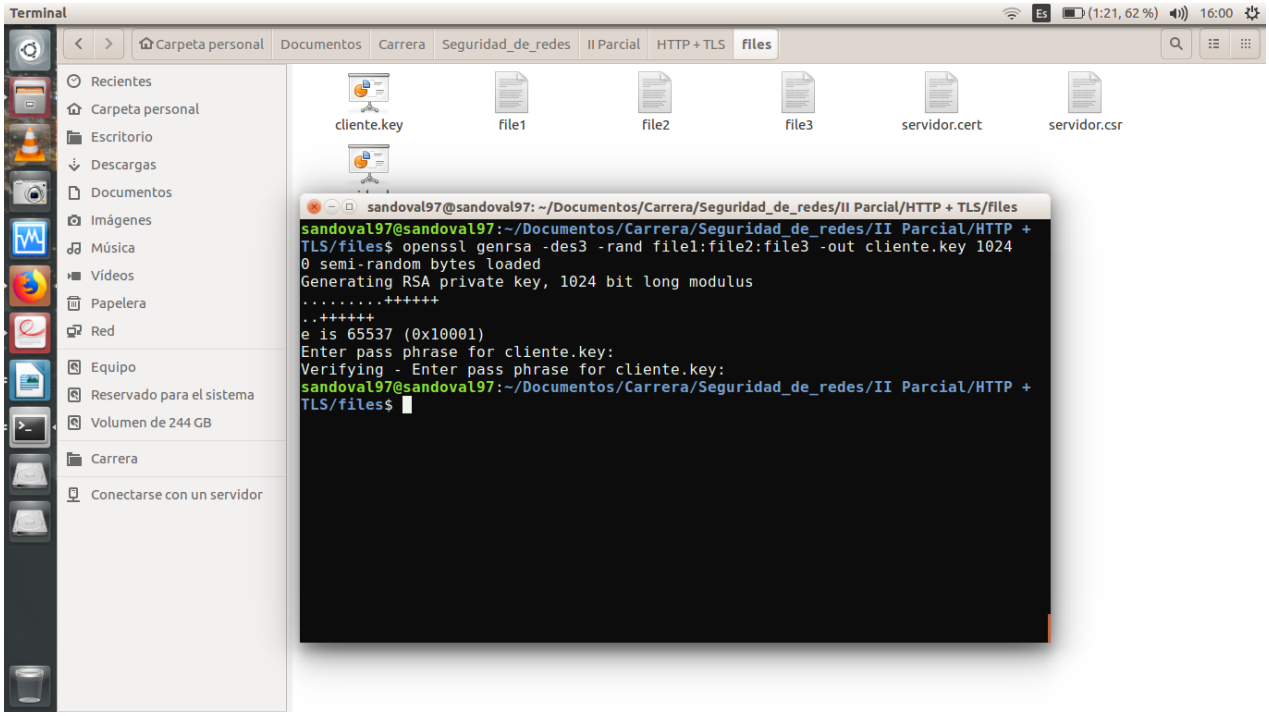
Para ello simplemente introducimos la dirección Web del servidor y observamos como aparece la página de inicio del servidor. Si accedemos al servidor seguro vemos que obtenemos una pantalla donde el servidor nos envía su certificado el cual podemos visualizar. Pero aún nos generará un error ya que configuramos el servidor para que pidiera autenticación para el cliente y aún no hemos generado el certificado del cliente.



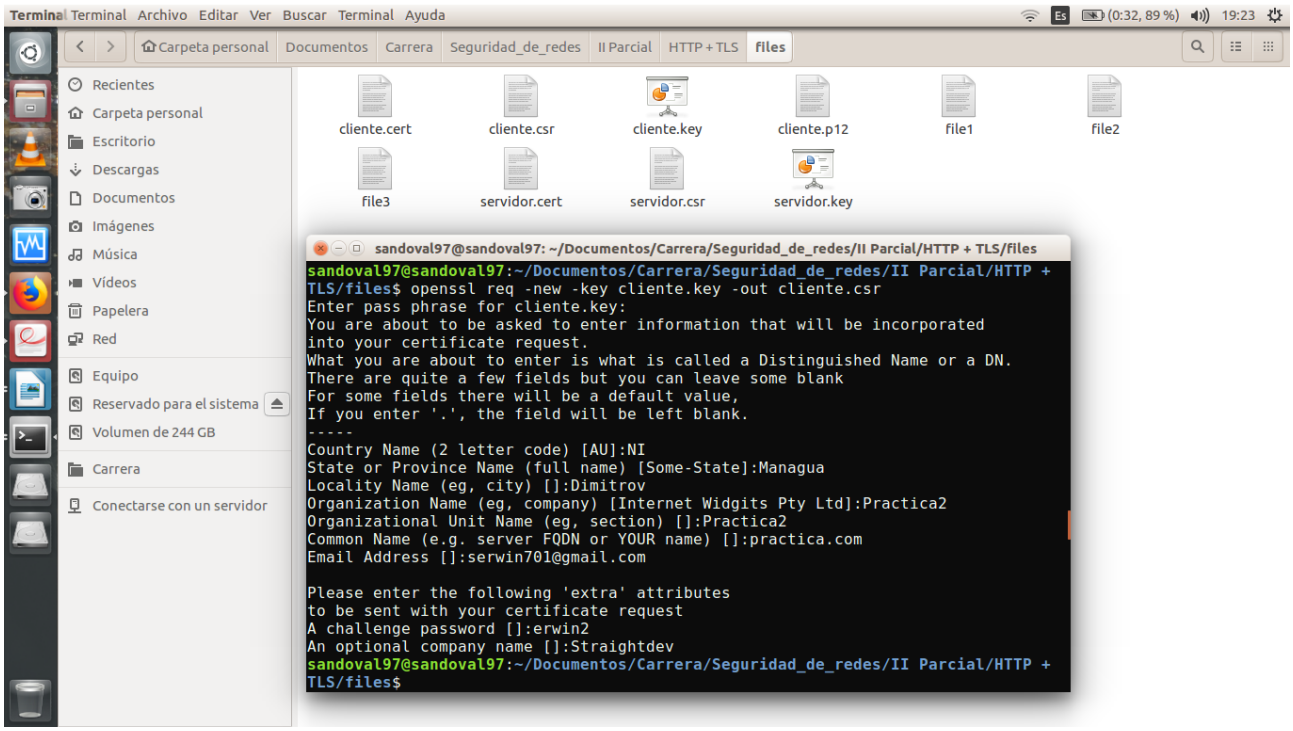
4. Acceso utilizando el navegador WEB.

Vamos a comprobar cómo generar un certificado e integrarlo en un navegador de manera que podamos utilizarlo para autenticarnos ante el servidor que solicita autenticación del cliente. El primer paso es generar el certificado del cliente tal y como hemos generado el certificado del servidor.

 # openssl genrsa -des3 -rand file1:file2:file3 -out cliente.key 1024

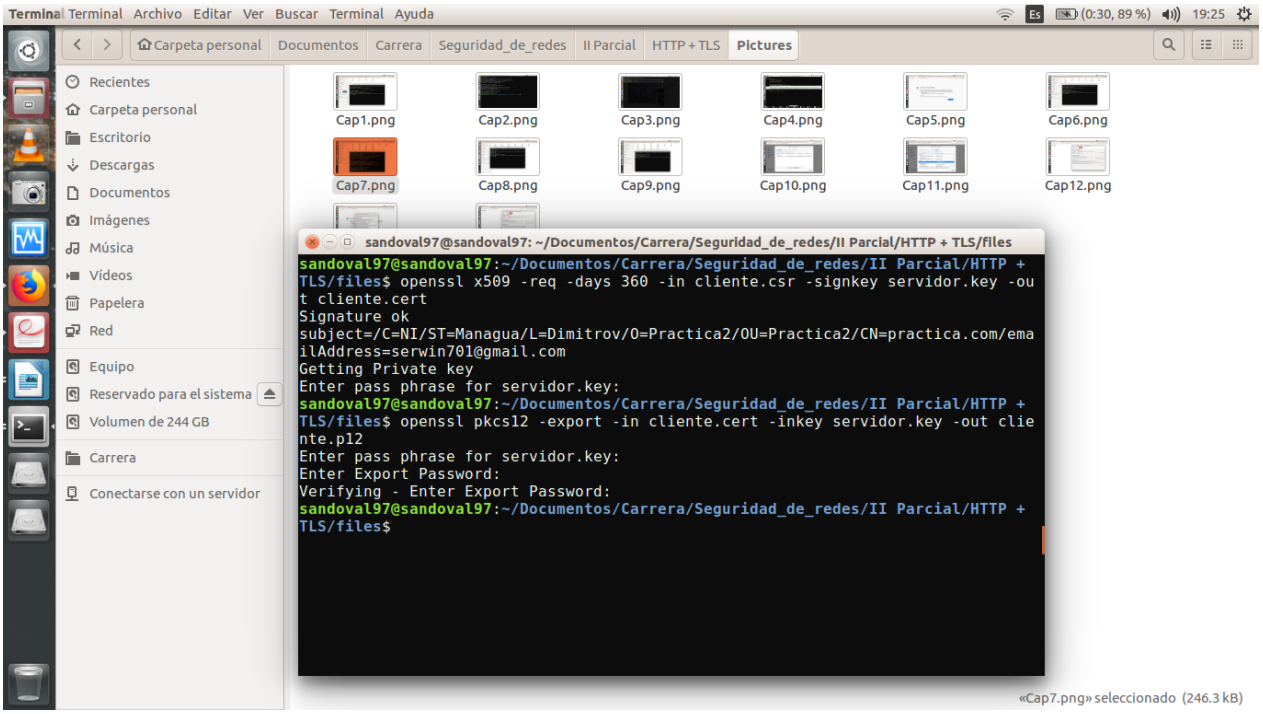


Una vez que tenemos nuestra pareja de claves, nos falta generar el certificado:
 # openssl req -new -key cliente.key -out cliente.csr



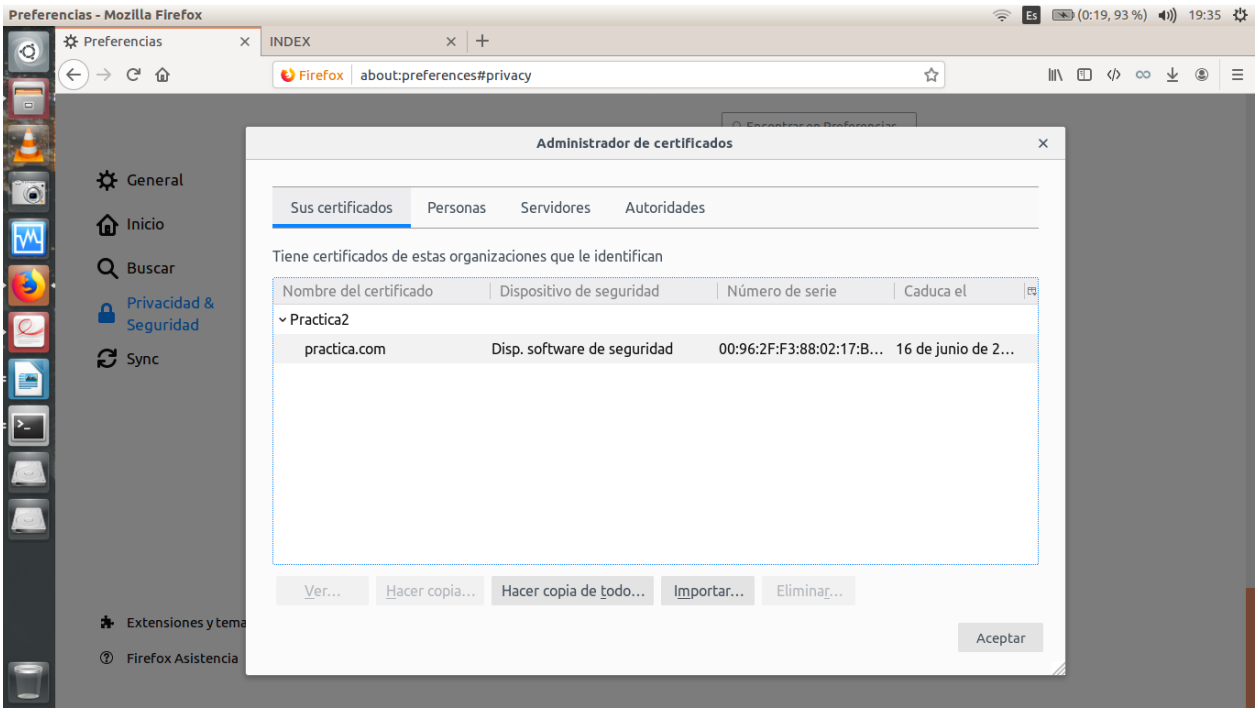
Una vez que tenemos nuestra petición de certificado (Certificate Signing Request) y nuestra pareja de claves, debemos de enviar el certificado a una autoridad de certificación para que lo firmara. En nuestro caso lo que vamos hacer es firmarlo utilizando la clave privada del servidor utilizando la siguiente orden:
 # openssl x509 -req -days 360 -in cliente.csr -signkey servidor.key -out cliente.cert

Tras este paso lo único que nos quedaría por hacer sería exportar el certificado en un formato tal que los navegadores lo puedan entender. Si probamos a importar el certificado tal y como aparece ahora, el navegador lo va aceptar en primera instancia pero luego no nos va a dejar utilizarlo. Para que pueda ser utilizado por los navegadores debemos exportarlo en el formato PKCS12. Para convertir del formato PEM que es el utilizado por OpenSSL al formato PKCS12 debemos utilizar la siguiente orden:
 # openssl pkcs12 -export -in cliente.cert -inkey servidor.key -out cliente.p12 -name
“Certificado”

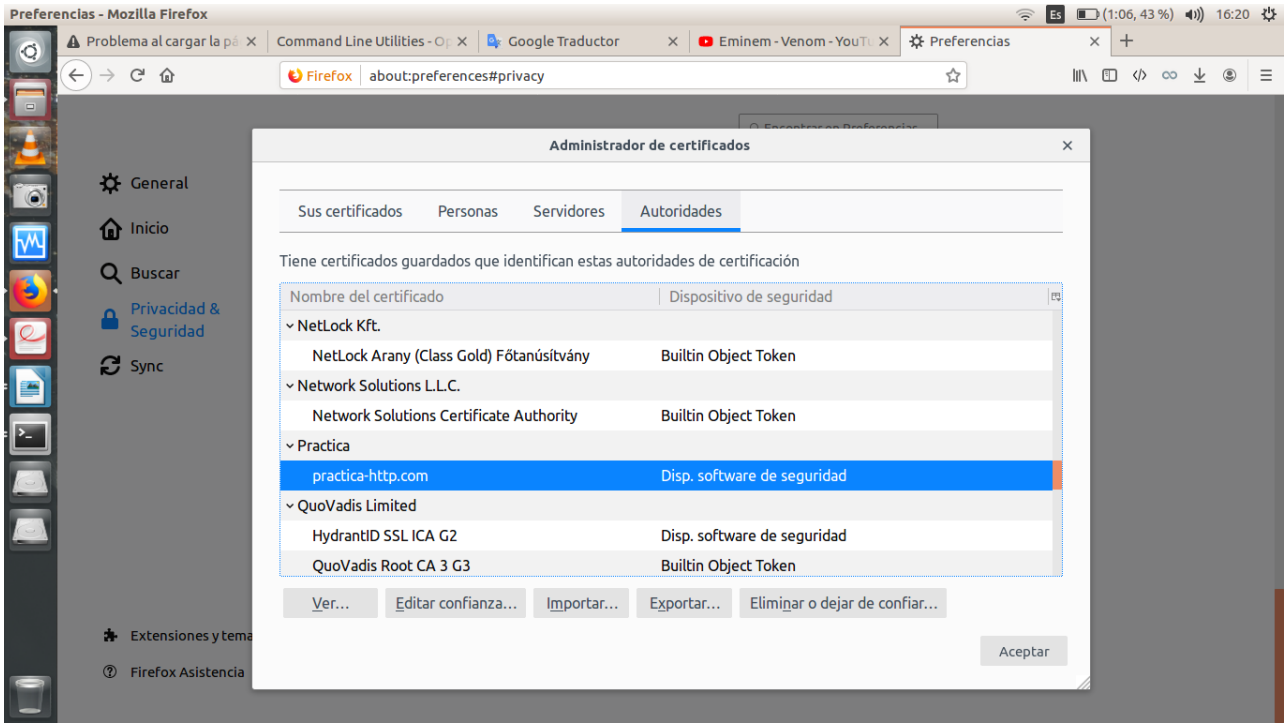


En criptografía, **PKCS # 12** define un formato de archivo de almacenamiento para almacenar muchos objetos de criptografía como un solo archivo. Normalmente se utiliza para agrupar una clave privada con su certificado X.509 o para agrupar a todos los miembros de una cadena de confianza.

Una vez que tenemos el certificado en formato PKCS12 lo importamos desde el navegador. En Firefox, la forma de importarlo es dirigirse a Edit -> Preferences -> Security -> View Certificates -> Your Certificates ->Import.



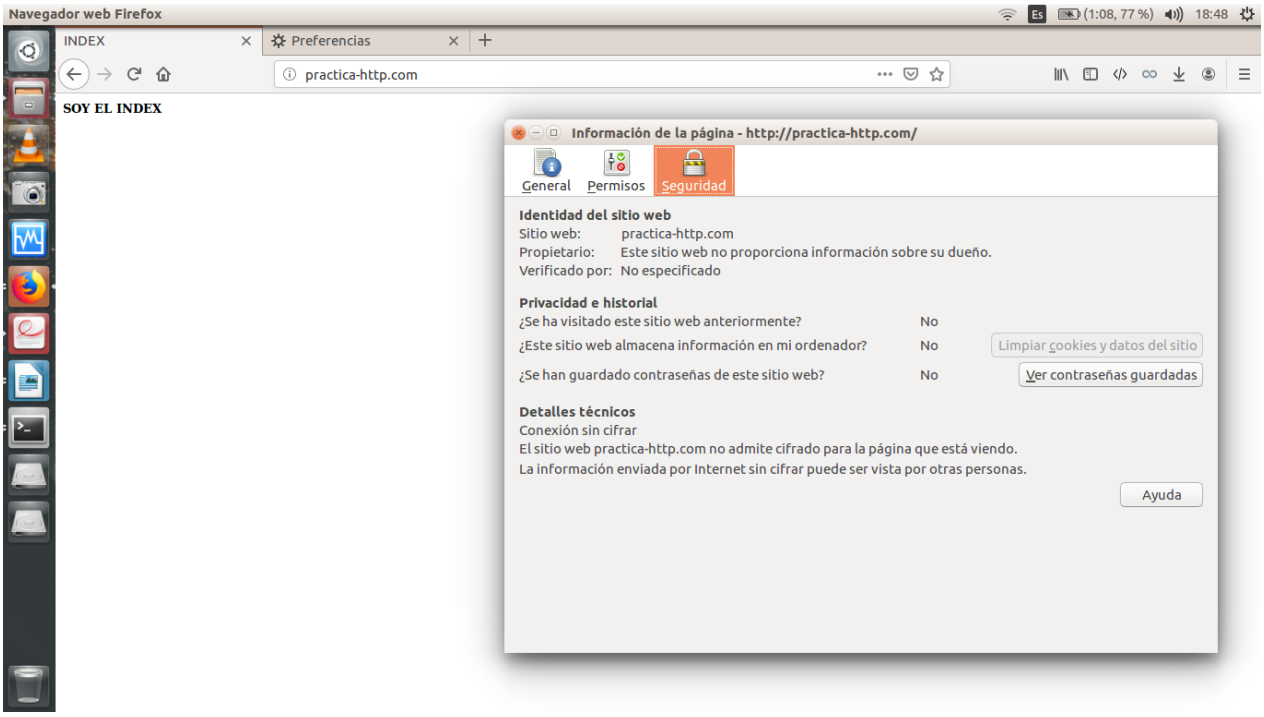
Por último debemos también importar el certificado de la autoridad de certificación que ha generado el certificado del cliente, En Firefox, la forma de importarlo es dirigirse a Edit -> Preferences -> Security -> View Certificates -> Authorities -> Import.



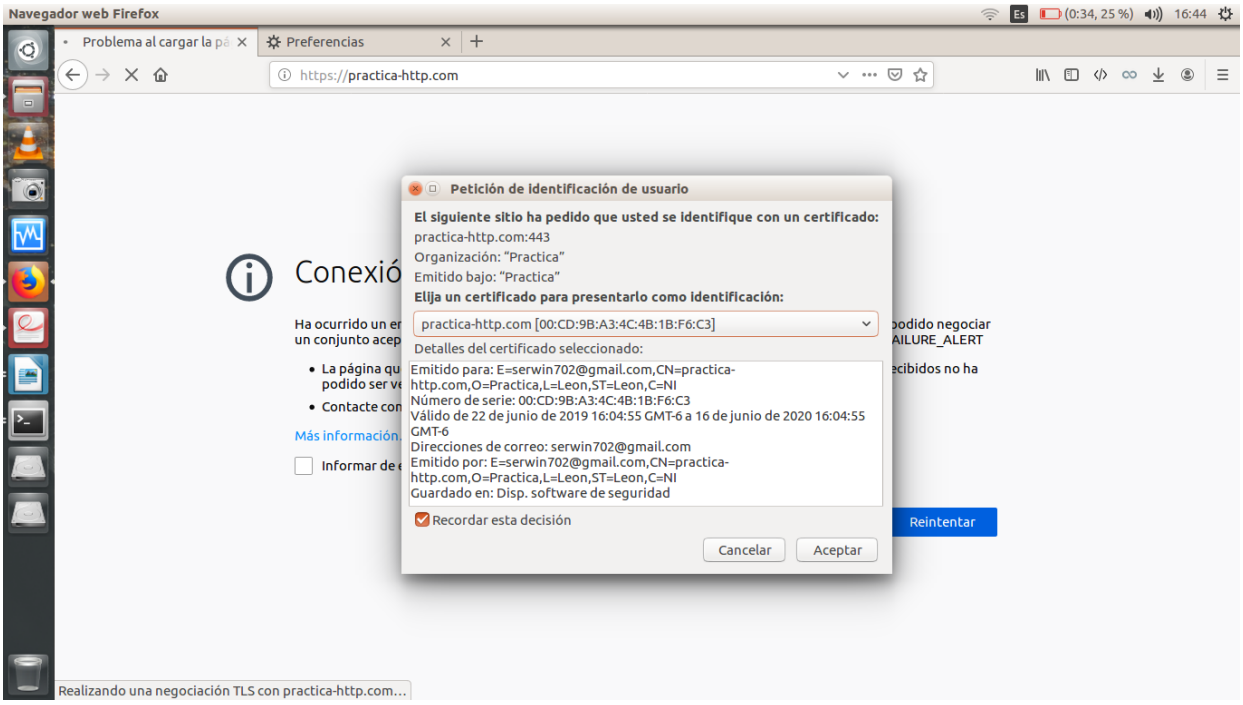
Una vez importado el certificado accedemos al servidor seguro.

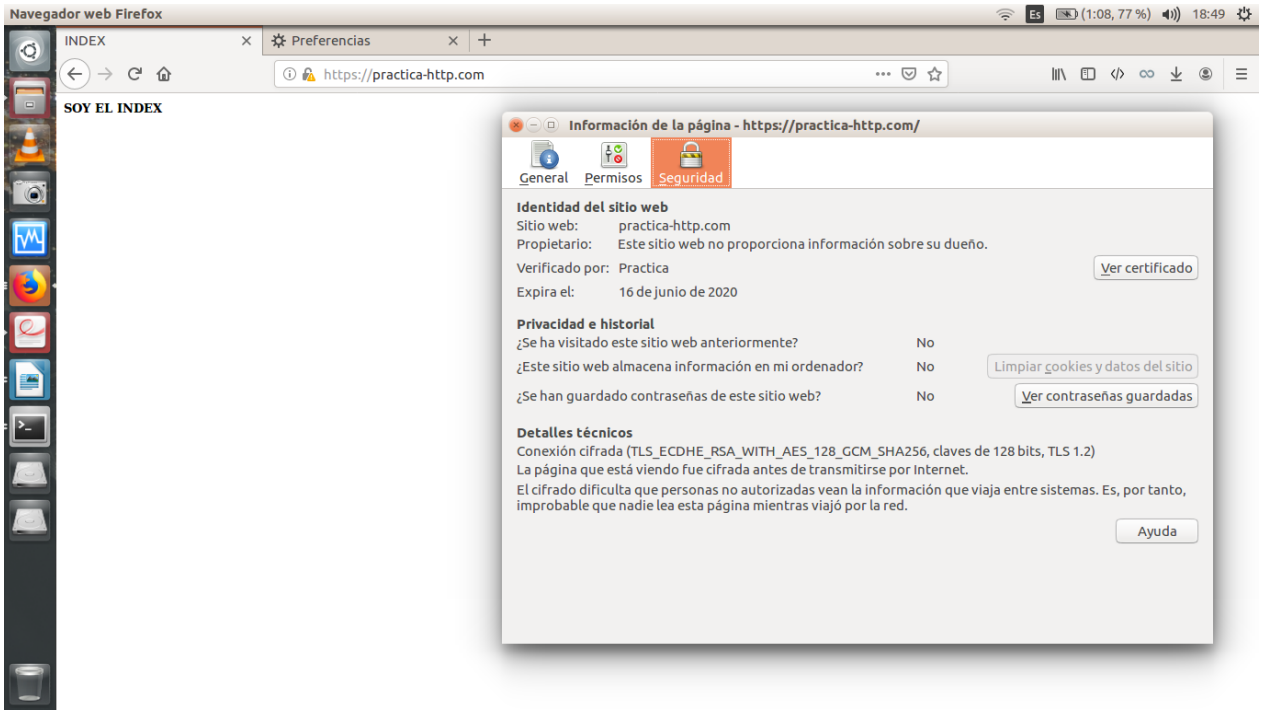
5. Verificaciones

1. Conseguir acceder al servidor Apache utilizando un navegador sin autenticación de cliente.



2. Conseguir acceder al servidor Apache utilizando un navegador con autenticación de cliente.




Navegador web Firefox
Es
(1:08, 77 %)
18:49
INDEX
Preferencias
https://practica-http.com
SOY EL INDEX
Información de la página - https://practica-http.com/
General
Permisos
Seguridad
Identidad del sitio web
Sitio web: practica-http.com
Propietario: Este sitio web no proporciona información sobre su dueño.
Verificado por: Practica
Ver certificado
Expira el: 16 de junio de 2020
Privacidad e historial
¿Se ha visitado este sitio web anteriormente? No
¿Este sitio web almacena información en mi ordenador? No
Limpiar cookies y datos del sitio
¿Se han guardado contraseñas de este sitio web? No
Ver contraseñas guardadas
Detalles técnicos
Conexión cifrada (TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, claves de 128 bits, TLS 1.2)
La página que está viendo fue cifrada antes de transmitirse por Internet.
El cifrado dificulta que personas no autorizadas vean la información que viaja entre sistemas. Es, por tanto, improbable que nadie lea esta página mientras viajó por la red.
Ayuda