ITEM 06 – PROTOCOLO HTTPS

Manejo de conexiones seguras

María Victoria Calbet González
Marta Ramírez González
David Romero Esparraga
Jesús Ortiz Calleja
Guillermo Alcalá Gamero
Juan Carlos Utrilla Martín

Contenido

1. Introducción 3
2. Protocolo HTTPS 4
2.1. Características técnicas 4
2.2. Capas de red 4
3. Procedimientos 5
3.1. Verificando el uso de HTTPS: 5
3.2. Configurando el servidor Apache Tomcat 5
3.3. Configurando la aplicación web 8
4. Fuentes 9

1. Introducción

La LOPD requiere datos personales o datos que estén regulados por un contrato para ser manejados y comunicados de forma segura. Manejar datos de forma segura requiere que nuestra aplicación maneje los datos de cada cliente de forma independiente de los datos y que mantenga seguras las computadoras en las que almacena y procesa. Las comunicaciones seguras requieren la configuración del servicio Tomcat para que utilice el protocolo HTTPS. (Tenga en cuenta que no es la versión de desarrollo que utiliza Eclipse o por medio de Maven en la configuración de desarrollo, sino el servicio Tomcat que utiliza sus configuraciones previas o de producción. Para obtener un A+ debe:

1. Actualizar su plantilla de proyecto a la versión 1.14 para que pueda manejar comunicaciones seguras. Tenga en cuenta que se espera que use esta plantilla para los próximos proyectos, incluso si no opta por ganar un A+ en los próximos entregables.

2. Use una nueva plantilla para producir la nueva versión del proyecto “Acme-Rendezvous” que use el protocolo HTTPS cuando corresponda.

3. Escriba un informe en el que explique a los profesores que tienen que hacer para verificar tu proyecto. Proporcione archivos de configuración previa a la producción y ejecutar su proyecto utilizando comunicaciones seguras.

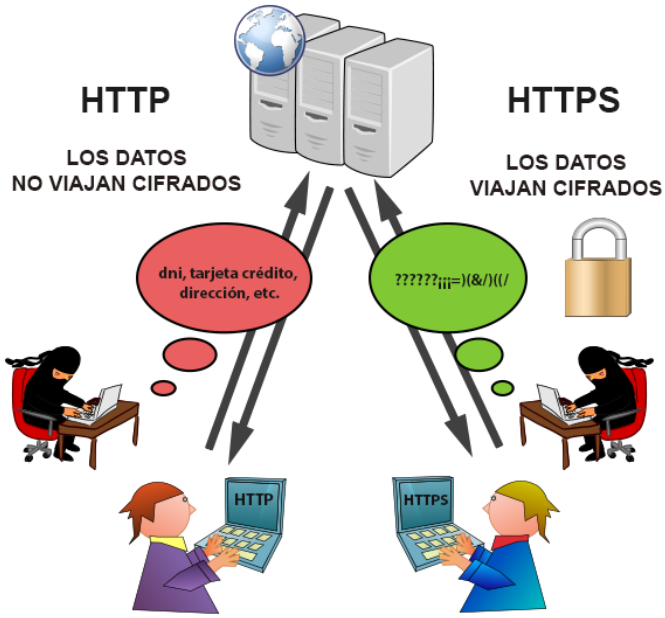
2. Protocolo HTTPS

HTTPS (Hypertext Transfer Protocol Secure) es un protocolo de la capa de la capa de aplicación basado en el protocolo HTTP, destinado a la transferencia segura de datos de datos de Hipertexto, es decir, es la versión segura de HTTP.

2.1. Características técnicas

El protocolo HTTPS utiliza un cifrado basado en SSL/TLS para crear un canal cifrado (cuyo nivel de cifrado depende del servidor remoto y del navegador utilizado por el cliente) para el tráfico de información sensible que el protocolo HTTP. De esta forma, se consigue que la información sensible no pueda ser usada por un atacante que haya conseguido interceptar la transferencia de datos de la conexión, ya que lo único que obtendrá será un flujo de datos cifrados que le resultará imposible de descifrar.

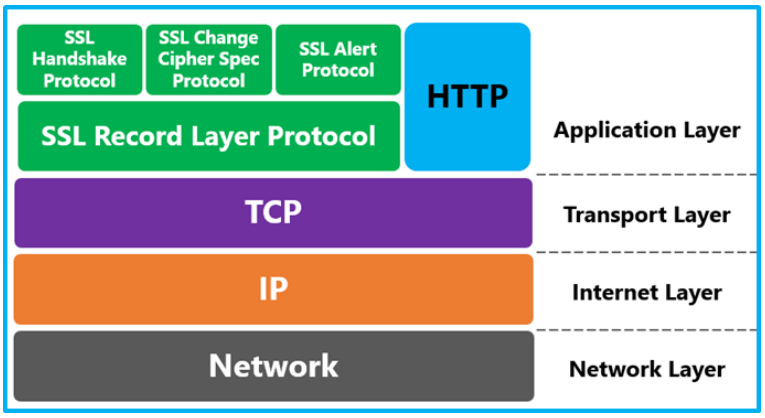
El puerto estándar para este protocolo es el 443.



HTTP es inseguro y está sujeto a ataques man-in-the-middle y eavesdropping que pueden permitir al atacante obtener acceso a cuentas de un sitio web e información confidencial. HTTPS está diseñado para resistir esos ataques y ser más seguro.

2.2. Capas de red

HTTP opera en la capa más alta del modelo OSI, la capa de aplicación; pero el protocolo de seguridad opera en una subcapa más baja, cifrando un mensaje HTTP previo a la transmisión y descifrando un mensaje una vez recibido. Estrictamente hablando, HTTPS no es un protocolo separado, pero prefiere el uso del HTTP ordinario sobre SSL o una conexión con seguridad en la capa de transporte sobre TLS.



3. Procedimientos

3.1. Verificando el uso de HTTPS:

Para verificar el uso de HTTPS utilizaremos el entorno de Pre-producción. Para verificar que el proyecto funciona correctamente y que utiliza dicho protocolo, introduciremos la siguiente URL en el navegador:

https://www.acme.com

3.2. Configurando el servidor Apache Tomcat

Para configurar el servidor Apache Tomcat para el uso de HTTPS debemos de seguir los siguientes pasos:

- **Paso 1**: los sistemas Java usan un almacén de certificados propio o ***keystore*** que es la base de datos de las claves privadas y sus certificados asociados. Para manipular este almacén, Java cuenta con la utiliza ***keytool*** que se encuentra distribuida entre los binarios del **entorno de desarrollo Java**.

  Utilizaremos *keytool* para la generación de pares de claves, ya que automáticamente almacena **la clave pública dentro de un certificado autofirmado**. Para generar el *keystore* con el certificado autofirmado, ejecutamos los siguientes comandos:

```
cd %JAVA_HOME%/bin
keytool.exe -genkey -alias tomcat -keyalg RSA
```

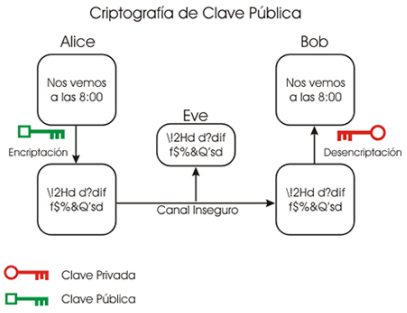
Nota: %JAVA_HOME% es una variable de entorno de Windows que indica el directorio donde se encuentra instalado Java Development Kit (JDK). En nuestro caso, se encuentra en:

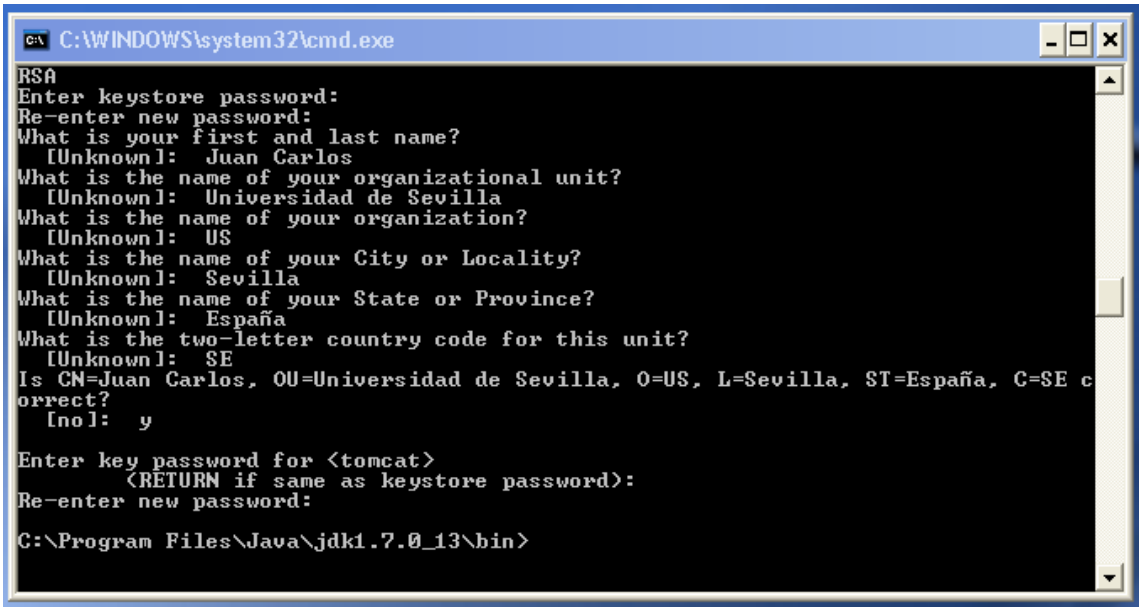
```
C:/Program Files/Java/jdk1.7.0_13
```

Acto seguido, la consola de comandos nos realizará una serie de preguntas las cuales debemos ir respondiendo. Si todos los datos son correctos, al final introducimos un 'y'.

keytool nos solicitará dos claves:

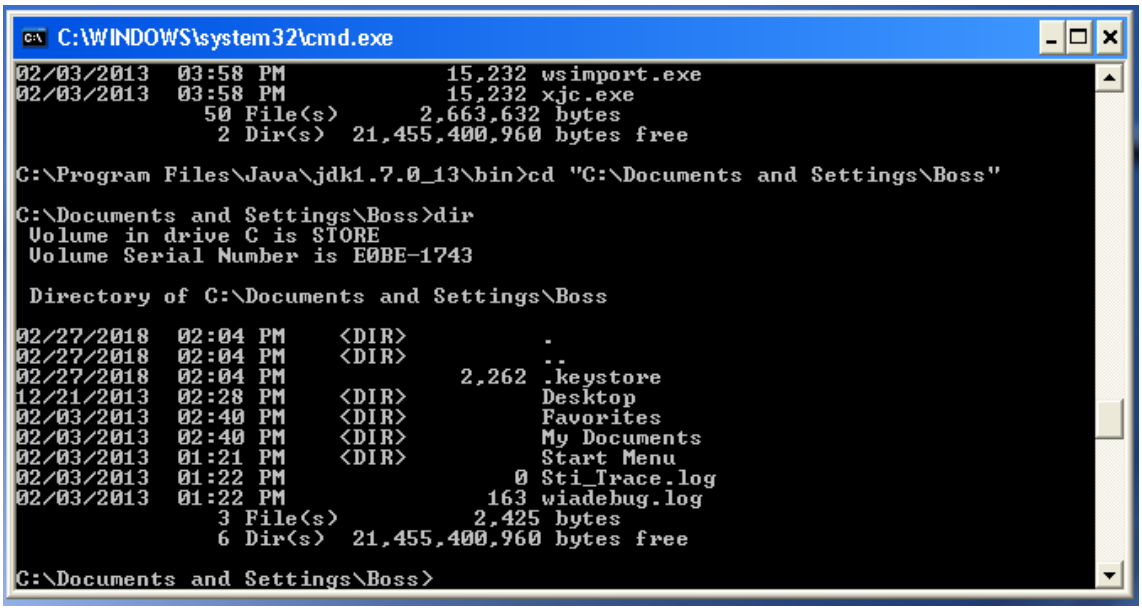
- **Clave pública**: cuando se quiere enviar un mensaje, el emisor busca la clave pública con esa clave.
- **Clave privada**: una vez que el mensaje cifrado llega al receptor, este se ocupa de descrifrarlo con su clave privada





- **Paso 2**: como resultado, se generará un fichero ".keystore" en la ruta:

```
C:\Documents and Settings\Boss
```



- **Paso 3**: ahora nos centraremos en configurar el servidor de aplicaciones Apache Tomcat sobre SSL necesaria para el protocolo HTTPS. Para ello, se debe editar la siguiente configuración que habilitará el puerto para que indiquemos para todo el tráfico HTTPS en el fichero *server.xml* ubicado en la siguiente ruta:

```
%CATALINA_BASE%/conf/server.xml
```

> **Nota**: %CATALINA_BASE % es una variable de entorno de Windows que indica el directorio donde se encuentra instalado el servidor de aplicaciones Apache Tomcat. En nuestro caso, se encuentra en:
>
> `C:/Program Files/Apache Software Foundation/Tomcat 7.0`

Añadimos:

```
<Connector port="8443"
      protocol="org.apache.coyote.http11.Http11NioProtocol"
      SSLEnabled="true"
      maxThreads="150" scheme="https" secure="true"
      disableUploadTimeout="true"
      clientAuth="false" sslProtocol="TLS"
      enableLookups="false" acceptCount="100"
      keystoreFile="C:\Documents and Settings\Boss\.keystore"
      keystorePass="nuestraContraseña"/>
```

- **Paso 4:** para verificar los cambios realizados, reiniciamos el servidor y escribimos:

https://localhost:8443/manager

> **Nota**: como es lógico la anterior URL http://localhost/manager ha dejado de funcionar.

> **Problema**: dependiendo del navegador, puede mostrar un error de validación debido a que se está accediendo a través de una IP cuando **un certificado debe estar ligado siempre a un dominio**.
>
> 
>
>
> **Solución**: esto desaparecerá cuando se configure el dominio a quien pertenece el certificado Apache Tomcat.

Si hemos seguido todos los pasos, nuestro servidor de aplicaciones estará preparado para realizar conexiones seguras mediante HTTPS.

3.3. Configurando la aplicación web

Para configurar una aplicación web para el uso de HTTPS debemos de seguir los siguientes pasos:

- **Paso 1**: modificaremos el fichero *web.xml* de la aplicación, en este caso, "*Acme-Rendezvous*":

```
<security-constraint>
        <web-resource-collection>
                <web-resource-name>Acme-Rendezvous</web-resource-name>
                <url-pattern>/*</url-pattern>
        </web-resource-collection>
        <user-data-constraint>
                <transport-guarantee>CONFIDENTIAL</transport-guarantee>
        </user-data-constraint>
```

```
</security-constraint>
```

CONFIDENTIAL se utiliza para asegurarnos de que nuestra aplicación web trabajará con SSL.

Nota: es necesario reiniciar el servidor Apache Tomcat para que los cambios funcionen.

- **Paso 2**: exportamos el proyecto war, preparamos los scripts y lo subimos a la url:

https://localhost:8443/manager

4. Fuentes

- **Información sobre el protocolo HTTPS**: https://es.wikipedia.org/wiki/Protocolo_seguro_de_transferencia_de_hipertexto
- **Transport Layer Security (SSL y TLS)**: https://es.wikipedia.org/wiki/Transport_Layer_Security
- **Imagen – HTTP vs HTTPS**: http://pablitholadera.blogspot.com.es/2015/08/la-diferencia-entre-el-http-y-https.html
- **Imagen – Capas de red de HTTPS**: https://blogs.msdn.microsoft.com/kaushal/2013/08/02/ssl-handshake-and-https-bindings-on-iis/
- **SSL/TLS How to – Apache Tomcat 7.0**: https://tomcat.apache.org/tomcat-7.0-doc/ssl-howto.html
- **Instalar Tomcat e importar el certificado SSL existente**: https://raiolanetworks.es/blog/instalar-tomcat-e-importar-certificado-ssl-existente/
- **A Simple Step-By-Step Guide To Apache Tomcat SSL Configuration**: https://www.mulesoft.com/tcat/tomcat-ssl
- **RSA:** https://es.wikipedia.org/wiki/RSA