INGENIERÍA DE SEGURIDAD US
E.T.S. Ingeniería Informática. Universidad de Sevilla
Avda Reina Mercedes s/n. 41012 Sevilla
Tlf/Fax 954 557 139 **E-mail** lsi@lsi.us.es **Web**
www.lsi.us.es

PAI-2. COMUNICACIÓN SEGURA DE INFORMACIÓN EN UNA APLICACIÓN BASADA EN SSL/TLS

INTRODUCCIÓN

Una determinada empresa trata de implantar la Política de Seguridad **Bring your Own Device (BYOD)** seguro, que consiste en que los empleados utilicen sus propios dispositivos para realizar su trabajo, pudiendo tener acceso a recursos de la empresa tales como correos electrónicos, bases de datos y archivos en servidores corporativos. Para la transmisión de todos estos elementos es fundamental la implementación de canales de comunicación seguros. En proyectos anteriores hemos visto cómo **proceder al aseguramiento de la información de las empresas y organizaciones** mediante tecnologías de **aseguramiento de la integridad usando hashing seguros.** Todos estos elementos son fundamentales para la implementación de canales de comunicación seguros. Como se ha visto, una de las maneras para crear canales de comunicación es la utilización de ***Sockets***. Un Socket no es más que la especificación de una dirección IP y un puerto. La información puede transmitirse **en "claro" o cifrada** dependiendo del tipo de ***Socket*** que se utilice o de los algoritmos que se utilicen con la información a transmitir.

La política de seguridad de la organización nos dice con respecto a la seguridad en las transmisiones de información cliente-servidor:

*"… deberían ser **confidenciales e íntegras** y **además autenticadas**."*

Ya se han visto en proyectos anteriores varias formas para hacer la información integra, pero no confidencial y autenticada a través del canal de comunicaciones. Por ello en este proyecto se usarán sockets del tipo ***Secure Sockets Layers (SSL),*** como se observa en la Figura 1.

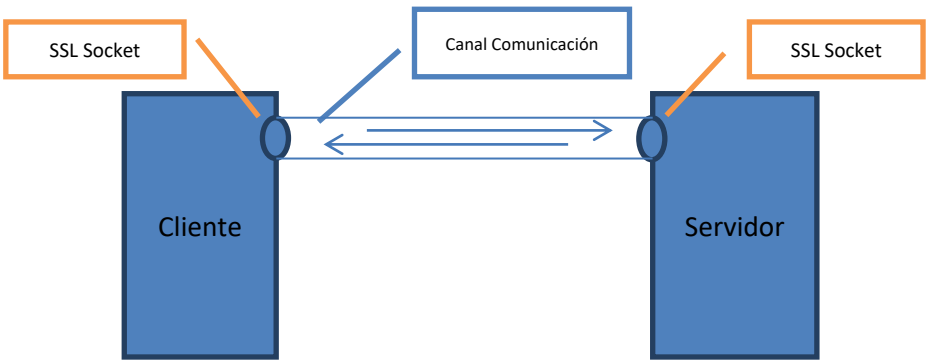

Figura 1: Canal de Comunicación Segura con SSL

SSL/TLS es un protocolo de comunicación seguro (implementa los requisitos de **autenticidad, confidencialidad e integridad**). Este protocolo utiliza una infraestructura basada en almacenes de claves y certificados. Estos canales de comunicación segura pueden servir para el cumplimiento de forma técnica a lo dispuesto en el artículo 104 sobre ***Telecomunicaciones Seguras*** del Real Decreto 1720/2007 que desarrolla la Ley Orgánica de Protección de Datos (LOPD) que dice:

"Cuando, conforme al artículo 81.3 ***deban implantarse las medidas de seguridad de nivel alto, la transmisión de datos de carácter personal a través de redes públicas o redes inalámbricas*** *de comunicaciones electrónicas* ***se realizará cifrando dichos datos o bien utilizando cualquier otro mecanismo que garantice que la información no sea inteligible ni manipulada por terceros****."*

Y también se dice en el Artículo 93.3 sobre Identificación y Autenticación:

"***Cuando el mecanismo de autenticación se base en la existencia de contraseñas existirá un procedimiento de asignación, distribución y almacenamiento que garantice su confidencialidad e integridad. "***

Esta política está bien soportada por la ***Open Security Architecture (OSA)*** donde se define el patrón de seguridad ***SP-008: Public Web Server Pattern*** (aunque en este caso no es un servidor Web***)***, donde para una infraestructura con un servidor público, se describe el control ***SC-09 Transmision confidentiality:***

*" **Control**: The information system protects the confidentiality of transmitted information."*

Objetivos

1. Desarrollar/seleccionar como llevar a la práctica los canales de comunicación segura para la transmisión de credenciales (usuario, contraseñas) y un mensaje con el Protocolo SSL/TLS (autenticidad, confidencialidad e integridad).
2. Utilizar una herramienta de análisis de tráfico que permita comprobar la confidencialidad e integridad de los canales de comunicaciones seguros.
3. Analizar la seguridad de los ***Cipher Suites*** usados y el funcionamiento de estos dentro del protocolo SSL/TLS en sus diferentes versiones.

Tareas para desarrollar

Tarea 1. Generación de la infraestructura Cliente-Servidor

En primer lugar, vamos a crear una infraestructura cliente-servidor para la **comprobación de login/password** de usuarios y un mensaje **secreto** mediante el uso de sockets seguros. Veremos un ejemplo específico usando la infraestructura en el lenguaje Java, para lo que seguiremos los siguientes pasos:

A. Creación de un keyStore – Compartición de claves

De acuerdo con la especificación del protocolo SSL/TLS se necesita para la autenticación de los servidores de los correspondientes certificados. En Java se puede realizar esto mediante la creación de un almacén o repositorio de certificados de seguridad para el protocolo SSL. Java viene provisto de una herramienta que permite crear de manera sencilla un almacén de certificados. Para crear el almacén de certificados ejecutamos en consola y con permisos de administración el siguiente comando:

```
keytool -genkeypair -keystore c:\SSLStore -alias SSLCertificate -keyalg RSA -keysize
2048
```

Keytool hará una serie de preguntas para crear la creación del keyStore. Con esto ya tendremos en la ruta C:\ un archivo SSLStore, que será el almacén de certificados que usaran nuestros Sockets cliente y servidor. De manera similar se podría realizar en un sistema basado en Linux.

Nota: Para usuarios de Windows, ***keytool*** se encuentra en la carpeta bin del JDK instalado en el sistema. Por lo tanto, para usar ***keytool*** desde línea de comandos sin tener que entrar en la ruta debemos configurar las variables de entorno JAVA_HOME y PATH con la ruta hacia la carpeta bin de la JDK de Java.

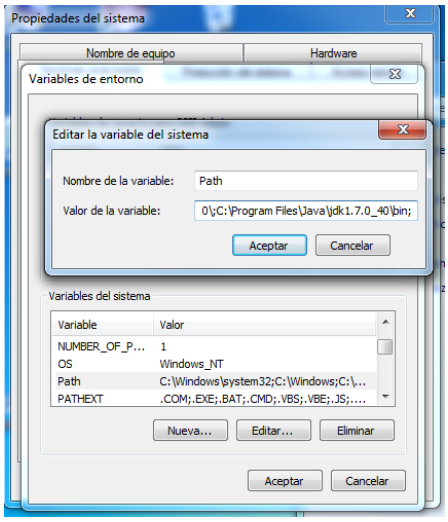

Figura 2: Configuración variable de entorno

Observe toda la información que se le pide al crear el keyStore **¿Para qué cree que puede ser necesario esta información?** Recoja todas sus reflexiones y el trabajo realizado para el informe y en su presentación.

B. Generación de un Socket TLS Servidor

Para utilizar comunicaciones seguras usando SSL/TLS entre cliente-servidor, utilizaremos Socket SSL/TLS. Crearemos una clase ***BYODServer.java*** con el mismo código que la clase LoginServer.java de la CAI-2 pero creando un SSL Socket Server en lugar de un Server Socket:

```
SSLServerSocketFactory socketFactory = (SSLServerSocketFactory)
                                        SSLServerSocketFactory.getDefault();
serverSocket = (SSLServerSocket) socketFactory.createServerSocket(7070);
```

El código se puede ejecutar en Eclipse, pero tenemos que tener en cuenta que son dos procesos (servidor y cliente) diferentes con diferentes parámetros, por tanto, tendremos que tener activada la correspondiente configuración de Eclipse. O también podemos no ejecutar directamente este código sobre Eclipse, y hacerlo en el entorno de ejecución del sistema operativo especificando los procesos y los parámetros a la JVM de dónde se encuentra el almacén de claves y la clave de este como veremos en el apartado D.

C. Generación de un Socket TLS Cliente

Crearemos una clase ***BYODCliente.java*** con la única diferencia con respecto al código de la CAI-2 en la creación de los **Sockets**, en este caso utilizamos las factorías ***SSLSocketFactory*** tal y como se muestra a continuación:

```
SSLSocketFactory socketFactory = (SSLSocketFactory)  SSLSocketFactory.getDefault();
SSLSocket socket = (SSLSocket) socketFactory.createSocket("localhost", 7070);
```

Notar que la IP y el puerto de conexión deben coincidir con el puerto e IP del servidor.

D. Compilar y ejecutar código cliente y servidor desde consola

1. Desde una consola con permisos de administración y en el directorio donde se encuentre la clase Java del cliente o servidor, ejecutaremos el siguiente comando:

   javac BYODServer.java

   Esto va a generar un archivo .class con el código interpretable por la JVM de Java, y que utilizaremos desde la misma línea de comandos para su ejecución.

2. Para ejecutar las clases una vez compiladas simplemente debemos de ejecutar el siguiente comando donde:

   - **BYODServer:**
     java –Djavax.net.ssl.keyStore=C:\SSLStore –Djavax.net.ssl.keyStorePassword=PASSWORD BYODServer
   - **BYODCliente:**
     java –Djavax.net.ssl.trustStore=C:\SSLStore –Djavax.net.ssl.
     trustStorePassword=PASSWORD BYODCliente

3. Compruebe que, si un empleado escribe en la correspondiente interfaz su **username**, su **contraseña**, y un **mensaje**, estos se envian sin problemas de seguridad y el servidor tras comprobar que es un usuario y password correcto, le informa al cliente que su mensaje SECRETO ha sido almacenado correctamente, en caso contrario le indicará que su mensaje no se ha almacenado.

Tarea 2. Análisis de tráfico de red en comunicaciones (sniffers)

Una manera de inspeccionar la información que fluye entre los Sockets es usar herramientas para analizar tráfico de red o sniffers. Los sniffers se interponen en los canales de comunicación y capturan toda la información que fluye por ellos. Existen sniffers muy sofisticados como *Wireshark* que permiten analizar múltiples protocolos, y otros sniffers más simples, por ejemplo, tcpdump que sólo capturan la información de red y hacen un volcado de la información de los paquetes en un archivo dump.

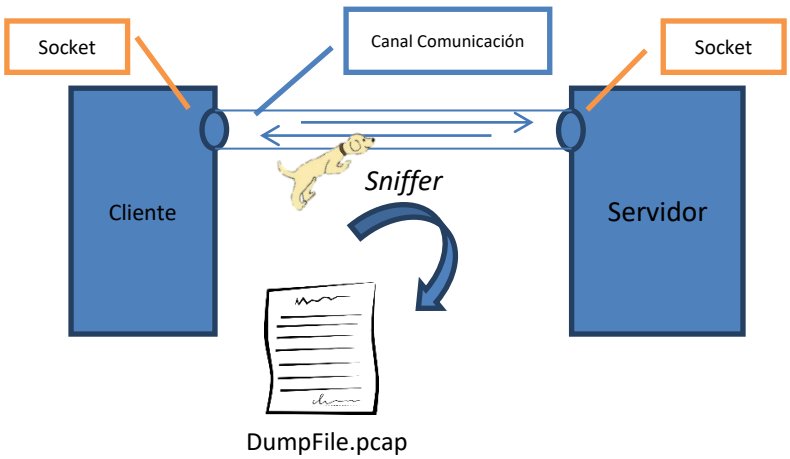

Figura 3: Monitorización del tráfico de red

En este proyecto si estamos en Windows, podríamos usar *RawCap* un sniffer para Windows que se utiliza desde línea de comandos, y/o *Wireshark. P*ara inspeccionar el archivo de salida de RawCap podremos usar Wireshark. En Linux no es necesario usar *RawCap*, podría usarse directamente Wireshark o un sniffer como *tcpdump* junto con *Wireshark.*

RawCap

RawCap nos hará dos preguntas: (1) la interfaz queremos escuchar; y (2) ruta y nombre del archivo donde queremos volcar la información (con extensión pcap). En nuestro caso vamos a capturar el tráfico en *loopback* o *localhost*, ya que el canal de comunicación se estable en nuestra propia máquina. *RawCap* interceptará todos los paquetes de información y los escribirá en el archivo especificado. En la siguiente imagen vemos a *RawCap* en marcha escuchando en la interfaz loopback, y la ruta (c:\ dumpfile.pcap) del fichero donde escribirá la información.

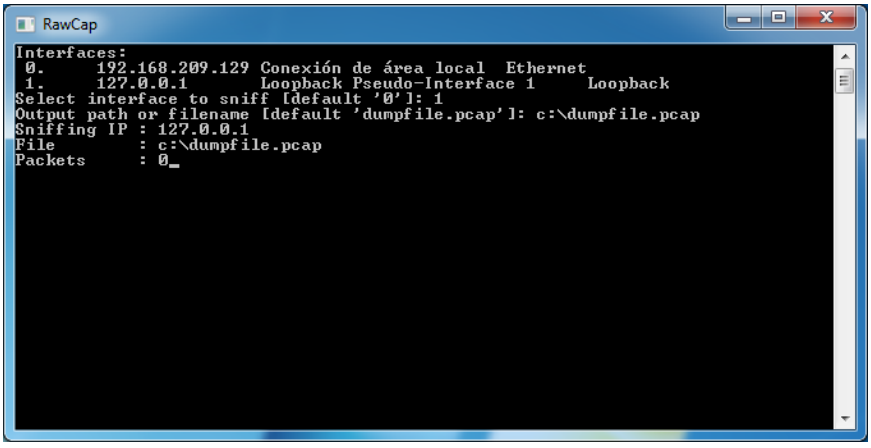

Figura 4: RawCap capturando información en la interfaz localhost

Wireshark

Una vez capturado el tráfico con ***RawCap*,** y obtenido el archivo .pcap se puede abrir con la herramienta ***Wireshark*** de tal manera que podemos observar el intercambio de paquetes que se ha producido en la interfaz seleccionada.

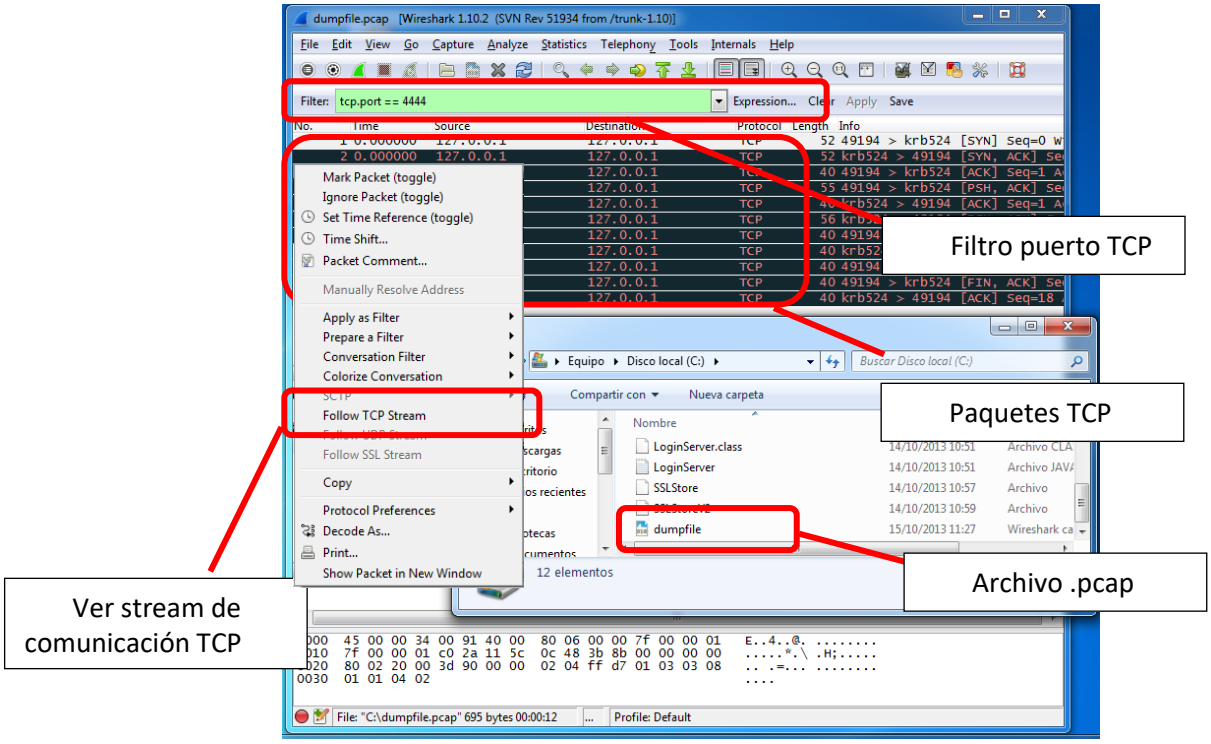

Figura 5: Configuración de Wireshark

El sniffer ha podido capturar múltiples paquetes que pasan por la interfaz loopback, en nuestro caso vamos a filtrar los paquetes capturados para observar sólo aquellos que utilicen el puerto 4444 como se observa en la Figura 2. Si utilizamos la opción de *"Follow TCP Stream"*

podemos observar todos los datos que se transmiten en la conexión de los Sockets que hemos llevado a cabo.

Haga uso de un analizador de tráfico de red (podría ser Wireshark, RawCap para Windows, ...) para observar la información en el canal de comunicación. **¿Podemos obtener los datos de nombre de usuario, contraseña o el mensaje secreto?** Además, haga una reflexión acerca de que está ocurriendo con el keyStore **¿se está utilizando el mismo keyStore? ¿Qué ocurriría si utilizamos dos máquinas distintas, donde estaría el keyStore?** Muestre en su informe y en la presentación todas las conclusiones acerca de esta tarea.

Tarea 3. Selección e implementación del conjunto de cipher suite más adecuado para la seguridad con SSL/TLS

Uno de los aspectos más relevantes para disponer de un **nivel de seguridad "fuerte" (strong en inglés)** en las comunicaciones SSL/TLS sería poder escoger la correspondiente ***Cipher Suite*** por parte de los clientes para realizar las comunicaciones seguras con el servidor. Se pretende entonces **descubrir a través del tráfico generado en las comunicaciones (capturado por Wireshark) la *Cipher Suite* que ha utilizado** por defecto cuando ha usado los ***Socket SSL*** en la última tarea (ejemplos de posibles suites ***TLS_RSA_WITH_AES_128_CBC_SHA, TLS_DHE_RSA_WITH_AES_128_CBC_SHA, TLS_DHE_DSS_WITH_AES_128_CBC_SHA, SSL_RSA_WITH_3DES_EDE_CBC_SHA***, etc.).

Haga pruebas de comunicación con TLS1.2 y TLS1.3. Compare las tramas que se pueden observar usando Wireshark, entre el uso de TLS1.2 y TLS1.3, ¿qué pasos se observan en el intercambio inicial o handshake? ¿qué ciphersuite se ha seleccionado en cada uno?. Presente en el informe el trabajo realizado para descubrir lo que se pide y analice el grado de seguridad que tiene esta ***Cipher Suite* y cómo la podríamos cambiar si lo consideramos poco segura e implemente dicho cambio en la solución**.

Normas del entregable

- Cada grupo debe entregar a través de la Plataforma de Enseñanza Virtual y en la **actividad** preparada para ello un archivo zip, nombrado ***PAI2-EquipodeTrabajoX*.zip**, que deberá contener al menos los ficheros siguientes:
  - ✓ **Documento en formato pdf que contenga un informe/resumen del proyecto** con los detalles más importantes de las decisiones, soluciones adoptadas y/o implementaciones desarrolladas, así como el resultado y análisis de las pruebas realizadas (máximo 15 páginas).
  - ✓ **Código fuente de las posibles implementaciones y/o scripts desarrollados y/o configuraciones y/o logs del analizador.**
- El plazo de entrega de dicho proyecto finaliza el **día 23 de noviembre a las 8:30 horas.**
- Los proyectos entregados fuera del plazo establecidos serán considerados inadecuados por el cliente y por tanto entrarán en penalización por cada día de retraso entrega de 15% del total, hasta agotarse los puntos.
- **El cliente no se aceptará envíos realizados por email, ni mensajes internos de la enseñanza virtual, ni correo interno de la enseñanza virtual.**

Métricas de valoración

Para facilitar el desarrollo de los equipos de trabajo el cliente ha decidido listar las métricas que se tendrán en cuenta para valorar los entregables de cada grupo de trabajo:

- **Documento (30%)**
    - Calidad del informe presentado y justificaciones
- **Solución aportada (70%)**
    - Cumplimiento de requisitos establecidos
    - Calidad del proyecto entregado
    - Complejidad de la solución
    - Respuesta al conjunto de preguntas planteadas
    - Calidad de pruebas realizadas y resultados