PAI2. COMUNICACIÓN SEGURA DE INFORMACIÓN EN UNA APLICACIÓN BASADA EN SSL/TLS

Escuela Técnica Superior de
Ingeniería Informática

Índice

Alcance y Objetivos del Proyecto

En este proyecto, se nos ha introducido a un tipo de comunicación cifrada que garantiza la confidencialidad y la integridad de los datos en las transmisiones. Estas comunicaciones suponen una mejora con respecto a la CAI 2, donde se nos planteaba garantizar la integridad de los datos pero no se trató la confidencialidad de la información, la cual se puede considerar el aspecto más importante hoy en día,

Garantizar la confidencialidad de los datos es una de las primeras tareas con las que las empresas tienen que lidiar, ya que una filtración de datos personales supone una flagrante violación del Reglamento General de Protección de Datos, RGPD, donde se menciona expresamente en la consideración 83.

Por tanto los objetivos de este proyecto son investigar las conexiones cifradas, y realizar las pruebas que demuestren su funcionamiento.

Recursos para el Proyecto

Recursos Humanos

El equipo de seguridad para esta asignatura se compone de los siguientes miembros:

- Barragán Candel, Marina - estudiante de Ing. Informática – Tecnología Informática, en la mención de Tecnologías de la información
- Calcedo Vázquez, Ignacio - estudiante de Ing. Informática – Tecnología Informática, en la mención de Sistemas de Información
- Polo Domínguez, Jorge - estudiante de Ing. Informática – Ingeniería de Computadores
- Sala Mascort, Jaime Emilio - estudiante de Ing. Informática – Tecnología Informática, en la mención de Computación

El equipo se organizará mediante la herramienta de Github, bajo el repositorio público SSII

Recursos Tecnológicos

Dado el carácter de la asignatura y la facilidad del equipo con el uso de Python 3, se hará uso de este lenguaje para programar los distintos scripts necesarios así como cualquier adaptación de las herramientas recomendadas.

Estudio Inicial

En primera instancia, debemos investigar sobre las conexiones cifradas SSL/TLS, analizando los distintos tipos que existen y el nivel de seguridad que proporcionan .

Una conexión SSL es un protocolo de socket seguro, sin embargo, se ha visto comprometido por POODLE desde el 2014 por tanto no se recomienda el uso de ninguna de sus versiones por ello se va a optar por TLS, también llamado SSL/TLS,

SSL/TLS es otro protocolo que securiza la capa de transporte de distintas formas por un lado garantiza la integridad mediante HMAC con un cifrado superior a SHA256, la confidencialidad mediante AES, o Rijndael, bajo distintos modos, CBC, CCM, GCM, y por último el intercambio de claves se puede realizar mediante Diffie-Hellman o RSA, sin embargo, debido a Heartbleed se recomienda el uso de Diffie-Hellman que asegura *Forward secrecy,* donde se asegura que si una clave se ha visto comprometida, el resto no puede serlo.

Las actividades que se han desarrollado en el proyecto han sido las siguientes:

Generación de la Infraestructura Cliente-Servidor

Para la realización de las pruebas, debemos desarrollar una infraestructura cliente-servidor donde se simule la conexión entre una entidad bancaria y un cliente desde una aplicación, para ello, necesitamos la generación de los certificados, y los sockets del servidor y el cliente.

Generación de los certificados mediante OpenSSL

Para el correcto funcionamiento de TLS, debemos hacer uso de certificados, idealmente estarían firmados por una entidad certificadora, como puede ser la Fabrica Nacional de Moneda y Timbre, sin embargo para nuestro caso nos vale con un certificado autofirmado.

Mediante OpenSSL podemos crear nuestro propio certificado autofirmado, para ello introduciremos el siguiente comando:

```
openssl req -x509 -newkey rsa:4096 -keyout key.pem -out certificate.pem -days 365
```

Que nos llevará al proceso de configuración del certificado, en la siguiente imagen se muestra la creación de uno.

```
Generating a RSA private key
.....................................................................
.....................................+++++
writing new private key to 'key.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
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) [AU]:ES
State or Province Name (full name) [Some-State]:Sevilla
Locality Name (eg, city) []:Sevilla
Organization Name (eg, company) [Internet Widgits Pty Ltd]:SSII
Organizational Unit Name (eg, section) []:Grupo 10
Common Name (e.g. server FQDN or YOUR name) []:Jaime Sala
Email Address []:jaisalmas@alum.us.es
```

Se nos han pedido una serie de datos, que deberían certificar la autenticidad de nuestra persona o servidor, pero como es autofirmado, realmente podríamos poner lo que quisiéramos, en este caso se ha puesto información relevante al grupo y la asignatura.

Creación de un Socket TLS Servidor

Para hacer uso comunicaciones seguras usando SSL/TLS entre cliente-servidor, utilizaremos OpenSSL. Crearemos un script ServerSideSSLSecureSocket.py que iniciará un servidor en localhost en un puerto determinado que estará a la espera de un cliente para realizar el intercambio de datos.

A continuación, debemos realizar la parte del cliente, y ya que se nos ha pedido en un principio una aplicación, hemos realizado una aplicación de escritorio haciendo uso de Tkinter trasladando el flujo de la aplicación proporcionada en Java.

Ejecución del código

Para ejecutar el código, simplemente hemos tenido que realizar los siguientes comandos en consola:

1. Para hacer funcionar el servidor necesitamos hacer `python3 ServerSideSSLSecureSocket.py`, siempre teniendo en cuenta que los certificados y clave se encuentran en la misma ruta.
2. Para hacer funcionar el cliente repetimos el mismo proceso, ejecutamos en este caso lo siguiente, `python3 ClientSideSSLSecureSocket.py`.
3. Por último solo debemos hacer uso de la aplicación como se describe en el Anexo I.

Análisis de Tráfico de Red mediante Sniffers

Por último, debemos comprobar que efectivamente este tráfico está cifrado correctamente, para ello tenemos que hacer uso de los sniffers que son herramientas muy versátiles que nos permiten analizar el tráfico de la red, capturando las tramas que recibe nuestro ordenador.

Existen distintos sniffers ya sea Rawcap, Wireshark, tcpdump, etc. Debido a nuestra familiarización con Wireshark en la asignatura Redes de Computadores se optará por dicha herramienta.

Entonces procedemos a la realización del análisis, abrimos la herramienta de Wireshark y la ponemos a escuchar en la interfaz de loopback de nuestro ordenador, puede requerir privilegios de superusuario en Linux, y como filtro usamos tls para distinguir el tráfico que nos interesa.

Una vez configurada la herramienta, procedemos con la simulación de la comunicación. Obteniendo los siguientes resultados.

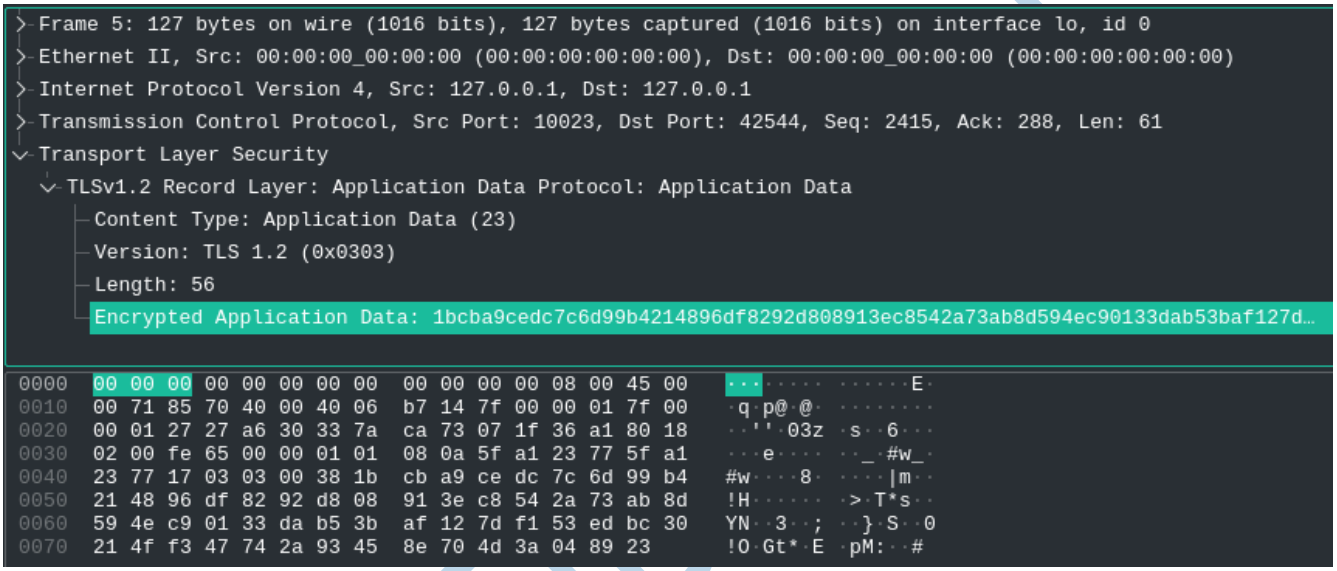
No.	Time	Source	Src Port	Destination	Dst Port	Protocol	Lengtl	Info
1	0.000000000	127.0.0.1	42544	127.0.0.1	10023	TLSv1.2	260	Client Hello
2	0.011982853	127.0.0.1	10023	127.0.0.1	42544	TLSv1.2	2254	Server Hello, Certificate, Server Key Exchange, Server Hello Done
3	0.012750638	127.0.0.1	42544	127.0.0.1	10023	TLSv1.2	159	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
4	0.012995305	127.0.0.1	10023	127.0.0.1	42544	TLSv1.2	292	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
5	0.013033446	127.0.0.1	10023	127.0.0.1	42544	TLSv1.2	127	Application Data
6	0.013261367	127.0.0.1	42544	127.0.0.1	10023	TLSv1.2	104	Application Data
7	0.013409116	127.0.0.1	10023	127.0.0.1	42544	TLSv1.2	136	Application Data
8	0.013659530	127.0.0.1	42544	127.0.0.1	10023	TLSv1.2	104	Application Data
9	0.013813088	127.0.0.1	10023	127.0.0.1	42544	TLSv1.2	149	Application Data
10	0.013977512	127.0.0.1	42544	127.0.0.1	10023	TLSv1.2	105	Application Data
11	0.014121268	127.0.0.1	10023	127.0.0.1	42544	TLSv1.2	131	Application Data
12	0.014269106	127.0.0.1	42544	127.0.0.1	10023	TLSv1.2	97	Application Data
13	0.014375804	127.0.0.1	10023	127.0.0.1	42544	TLSv1.2	149	Application Data
14	0.014484968	127.0.0.1	42544	127.0.0.1	10023	TLSv1.2	105	Application Data
15	0.014594053	127.0.0.1	10023	127.0.0.1	42544	TLSv1.2	131	Application Data
16	0.014763706	127.0.0.1	42544	127.0.0.1	10023	TLSv1.2	97	Application Data
17	0.014895368	127.0.0.1	10023	127.0.0.1	42544	TLSv1.2	130	Application Data

Donde se puede observar que se han realizado los siguientes pasos:

1. Client Hello, donde el cliente inicia la conexión con el servidor, dando una lista de la versión TLS que soporta el cliente y un identificador, además de una lista de Cipher Suites.
2. Server Hello, respuesta al cliente donde el servidor elige la versión más alta de TLS que admite el cliente y la Cipher suite que se va a utilizar, junto con el certificado y una clave pública, además de la firma digital que garantiza la autenticidad.

3. Client Key Exchange y Handshake, el cliente solo tiene que enviar el valor público del método escogido y acepta el método de cifrado y dando un aviso que su siguiente mensaje irá cifrado, enviando así un sumario de las operaciones realizadas cifradas, de modo que si el servidor no puede leerlo significa que ha ocurrido algo extraño y cortarían la conexión.
4. Handshake del servidor, el servidor se pone de acuerdo con el cliente para cifrar su conexión y avisa de igual forma al cliente que su siguiente mensaje irá cifrado y envía el resumen de operaciones igual que hizo el cliente.

Una vez finalizado este protocolo, podemos ver como está cifrada su conexión en la siguiente imagen, donde nos son irreconocibles los datos enviados.



Informe Final

Actualmente, ya se está empezando a usar una versión TLS más moderna, TLS 1.3, que ha introducido mejoras significativas en el proceso. En primer lugar, acorta el *handshake* entre el cliente y servidor a dos mensajes, *client-hello* y *server-hello*, ya que se han suprimido varias opciones, entre ellas la negociación de la versión de TLS, evitando así los ataques de downgrade, y limitando los *ciphers suites*, eliminando los más inseguros, con esto ya no son necesarias las negociaciones, el primer mensaje envía al servidor todo lo necesario para cifrar y el servidor envía la respuesta y el mensaje cifrado inmediatamente. En segundo lugar, disminuye el *Round Trip Time* a idealmente a 0, dado el caso en el que el cliente ya haya tenido una conexión reciente anterior, enviando al servidor una información básica para que recalcule el cifrado y enviando la petición cifrada a su vez, sin embargo, esto acarrea un problema y es que haría al servidor susceptible de un ataque de Replay, pero la IETF, *Internet Engineering Task Force*, ha delegado este problema a los servidores, haciendo que estos sean los que tengan que hacer frente a esto.

Los cambios a realizar para implementar este cambio en python son minúsculos, ya que solo habría que eliminar la especificación de uso de TLS 1.2, ya que por defecto escoge la máxima versión disponible.

Después de realizar los cambios vemos la siguiente captura.

No.	Time	Source	Src Port	Destination	Dst Port	Protocol	Lengtl	Info
1	0.000000000	127.0.0.1	53546	127.0.0.1	7070	TLSv1.3	583	Client Hello
2	0.045903773	127.0.0.1	7070	127.0.0.1	53546	TLSv1.3	2391	Server Hello, Change Cipher Spec, Application D
3	0.048750343	127.0.0.1	53546	127.0.0.1	7070	TLSv1.3	146	Change Cipher Spec, Application Data
4	0.049357192	127.0.0.1	7070	127.0.0.1	53546	TLSv1.3	321	Application Data
5	0.091094131	127.0.0.1	7070	127.0.0.1	53546	TLSv1.3	375	Application Data, Application Data
6	30.303106370	127.0.0.1	53546	127.0.0.1	7070	TLSv1.3	97	Application Data
7	30.303490597	127.0.0.1	7070	127.0.0.1	53546	TLSv1.3	129	Application Data
8	41.711080378	127.0.0.1	53546	127.0.0.1	7070	TLSv1.3	101	Application Data
9	41.711385653	127.0.0.1	7070	127.0.0.1	53546	TLSv1.3	142	Application Data
10	47.122593189	127.0.0.1	53546	127.0.0.1	7070	TLSv1.3	99	Application Data
11	47.124074396	127.0.0.1	7070	127.0.0.1	53546	TLSv1.3	124	Application Data
12	48.771586929	127.0.0.1	53546	127.0.0.1	7070	TLSv1.3	90	Application Data
13	48.771957825	127.0.0.1	7070	127.0.0.1	53546	TLSv1.3	142	Application Data
14	53.238926212	127.0.0.1	53546	127.0.0.1	7070	TLSv1.3	100	Application Data
15	53.239684028	127.0.0.1	7070	127.0.0.1	53546	TLSv1.3	124	Application Data
16	54.925079286	127.0.0.1	53546	127.0.0.1	7070	TLSv1.3	90	Application Data
17	54.926198595	127.0.0.1	7070	127.0.0.1	53546	TLSv1.3	123	Application Data

Como se puede observar, el proceso de conexión es más rápido ya que se ha reducido el número de mensajes a intercambiar hasta obtener el cifrado de conexión.

Anexo I: Manual de Uso de la Aplicación Bancaria

Para hacer uso de la aplicación bancaria, tras haber creado el certificado del cliente, simplemente tenemos que correr el script y se nos presentará el siguiente flujo: