PAI2-G8

13/03/2023

Pablo Mateos Angulo

Pablo Aurelio Sanchez Valenzuela

Abstract

En este documento se describe el software desarrollado en java utilizando la librería crypto capaz de realizar envíos de mensajes de texto entre un cliente y un servidor protegiendo la integridad con un hash para los ataques del tipo man in the middle y un nonce incluido en el envío del mensaje para cubrir de ataques replay.

1.Objetivos

Se propone a los Security Teams de INSEGUS alcanzar los objetivos siguientes:
1. Desarrollar/Seleccionar el verificador de integridad para los mensajes de transferencia bancaria que se transmiten a través de las redes públicas evitando los ataques de man-in-the-middle y de replay (tanto en el servidor como en el cliente).
2. Desplegar un verificador de integridad en los sistemas cliente/servidor para llevar a cabo la realización de la verificación de forma práctica de los mensajes transmitidos entre un servidor y un cliente.

2.Propuesta

Aquí tenemos un esquema para ver cómo funciona el orden de ejecución de un sistema cliente servidor.

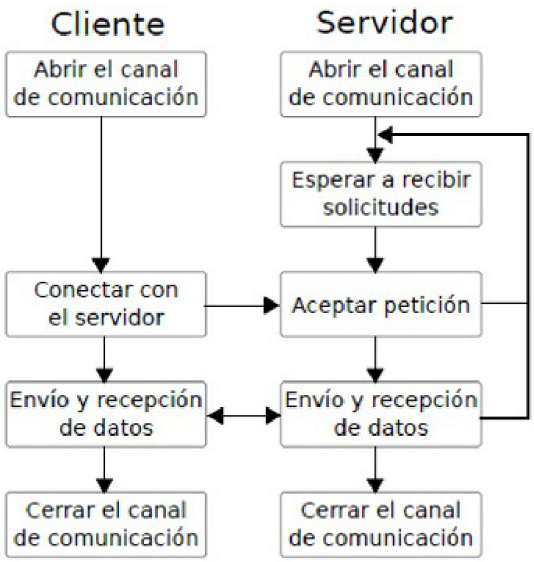

Figura 1. [1]

Se debe proteger este sistema de ataques Replay y Man in the Middle.

Un ataque de MITM es un tipo de ciberataque en el que los criminales interceptan una conversación o una transferencia de datos existente, ya sea escuchando o haciéndose pasar por un participante[2]. Aquí tenemos un ejemplo gráfico.<Figura 2>

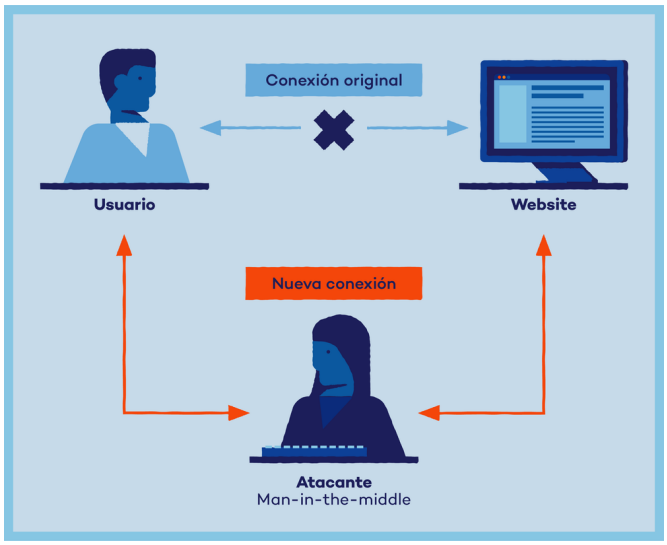

Figura 2.Ataque Man in the Middle. [2]

Un Ataque Replay o 'ataque de repetición' es un tipo de ataque de red en el que un pirata informático detecta una transmisión de datos y obtiene acceso a datos confidenciales actuando como el remitente original y enviando una comunicación a su destino original [3].Ejemplo de este tipo de ataque.<Figura 3>

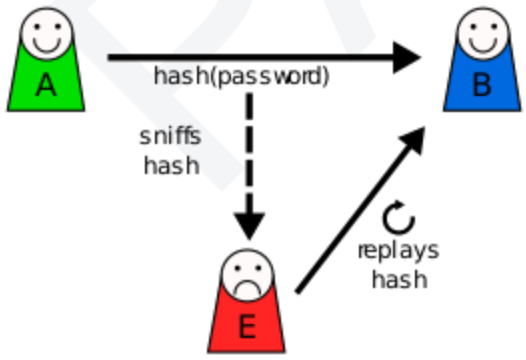

Figura 3.Ataque Replay. [4]

Para evitar ataques de replay se utiliza un nonce que tiene formato LocalDateTime, en el cuál se utiliza la fecha y hora actual para el envío de mensajes.Si este cuando llega al servidor se corresponde con el actual se verifica la integridad de transmisión del mensaje y el sistema muestra un mensaje de bienvenida ,sino el servidor muestra que el mensaje está repetido .

Para evitar ataques MITM se utiliza un cifrado hash HMAC256 en el que se incluye el mensaje y la clave secreta que funciona como cifrado simétrico. De tal forma que el mensaje y la clave para comprobar que es el cliente original el que lo envía están seguros contra posibles ataques en los que se intente hacer pasar un cliente falso por el cliente verdadero.

3.Experimentación

Para probar que el sistema funciona correctamente, se ha ejecutado en dos ordenadores que usan la misma red wifi, uno con el rol de cliente y otro con el rol de servidor el esquema mencionado anteriormente.

Se ha definido un usuario y contraseña por defecto de tal manera que se pide cuando se conecta con el servidor<Figura 4> y <Figura 5> .Además también se ha definido una clave por defecto que tienen de antemano tanto el cliente como el servidor que se usa para la realización del hashMac.

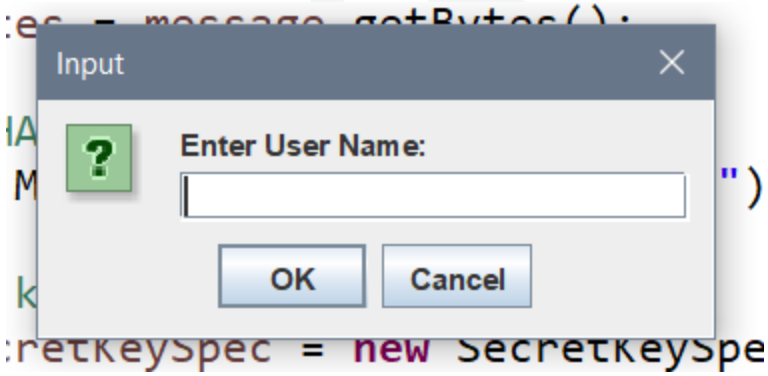

Figura 4.Insertar un nombre de usuario.

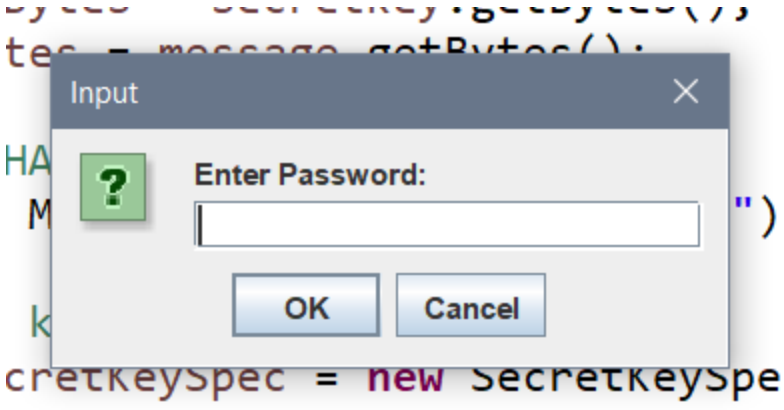

Figura 5.Insertar una contraseña válida.

Posteriormente se solicita el mensaje a enviar <Figura 6>

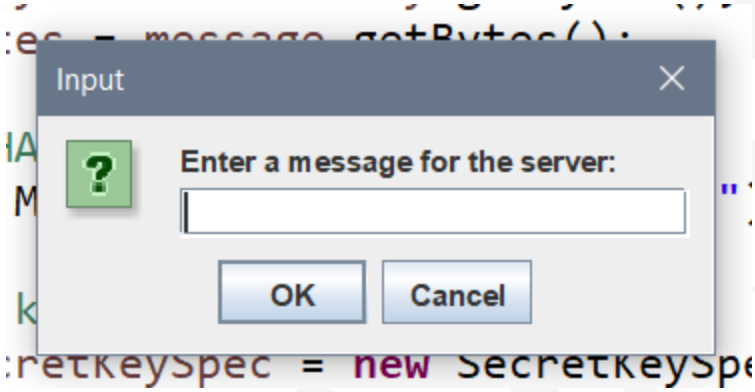

Figura 6.Introducir un mensaje

Este mensaje se combina con el nonce (que se obtiene automáticamente con un LocalDate.now() que incluye, año, mes, día, hora, minuto, segundo e incluso fracciones de segundo) se hashea y se envía al servidor tanto el mensaje con el nonce como el hash correspondiente al mensaje+nonce. Si cuando esto llega al servidor, el servidor es capaz de con la clave simétrica y el mensaje original (utiliza un .split(“,”) para separar el mensaje original del nonce) reproducir el hash que le ha llegado, pasará a comprobar la unicidad del nonce del mensaje (segunda parte del .split()) comparándolo con una lista de nonces ya utilizados y en caso de pasar ambas comprobaciones se retornará un mensaje que saluda de vuelta.<Figura 7> y si el mensaje era cualquier cosa que no fuera “hola” nos responderá con <Figura 8>.

En caso que se realizara un envío con un nonce repetido (replay attack) se mostraría por pantalla del que haya realizado el envío que ese mensaje ya ha aparecido antes <Figura 9> (un poco ambiguo que se refiera a mensaje, pero al ser replay entendemos que todo el mensaje será idéntico ya que consiste en capturar el mensaje en tráfico y enviarlo de nuevo. No hay problema si el texto es el mismo siempre que el nonce haya cambiado, que

lo hace automáticamente al calcular la hora actual del envío, así que esta alerta solo aparecería en envíos fraudulentos)

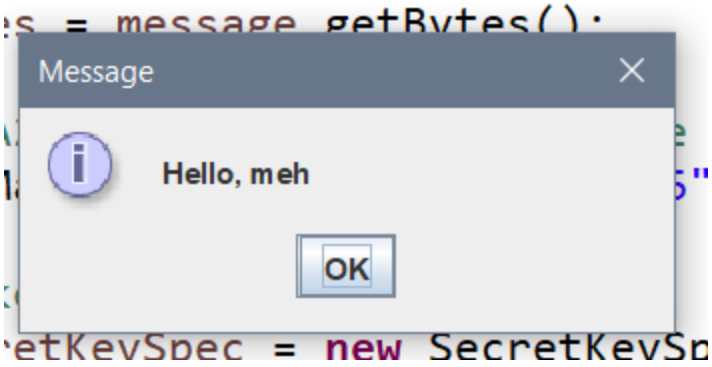

Figura 7.Mensaje de bienvenida al usuario.

Figura 8.Respuesta del servidor cuando el mensaje del cliente no es “hola”.

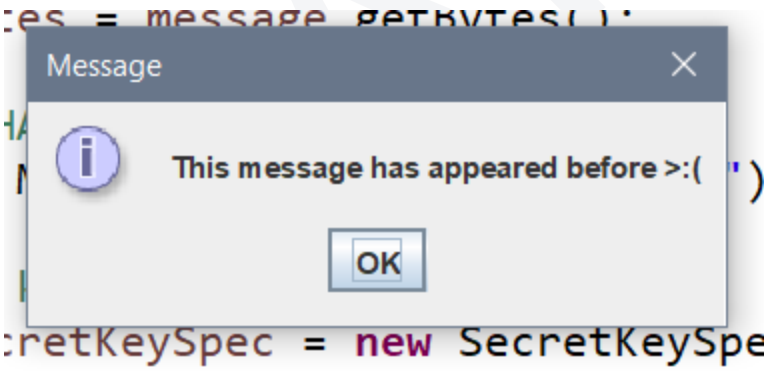

Figura 9.Ataque de replay.

Figura 10.Fallo de integridad.

Eficiencia de la solución propuesta y el reparto de claves

La solución propuesta tiene la ventaja y el inconveniente de ser un sistema extremadamente simple, se puede echar en cara que un reparto de claves simétricas que depende de que tanto cliente como servidor se pongan de acuerdo en una clave a utilizar o se la pasen por pen es algo poco deseable, pero de esta manera se puede asegurar también que es imposible de obtener si no se roba directamente de donde estas personas tengan el pen o se escuche la conversación donde la acuerdan.

En eficiencia bruta, aunque no tenemos tiempos exactos de ejecución podemos ver un código que ha evitado bucles en el envío y recepción, únicamente teniendo uno para hacer que el servidor se mantenga abierto a la espera de mensajes y otro para la conversión de binario a hexadecimal que tenemos planteada. Fuera de estos dos bucles (uno de los cuales no tiene impacto sobre el rendimiento) la aplicación corre en la librería crypto de java y algunos ifs, que en tiempos de ejecución son relativamente eficientes, con lo que concluimos, que pese a la falta de pruebas de velocidad, para una aplicación de intercambio de mensajes somos suficientemente eficientes como para que el cliente no note problemas a la hora de comunicarse de manera normal con el servidor.

4.Referencias

1.Esquema funcionamiento cliente servidor | Download Scientific Diagram (researchgate.net). [1]

2.Cómo prevenir los ciberataques "man in the middle" | Conversia (consultoria-conversia.es). [2]

3.¿Qué es un Ataque Replay? - Bitnovo Blog [3]

4.¿Qué es el ataque de reply? ¿Cómo prevenir el ataque de reply con VPN? (itopvpn.com)[4]

5.Código para montar un sistema cliente servidor en Java sacado de las diapositivas de la asignatura

Respecto a código se han utilizado muchas páginas para ver el funcionamiento de sistemas similares, principalmente se ha buscado en:

6.https://www.baeldung.com

7.https://www.w3schools.com

8.https://stackoverflow.com

9.https://www.geeksforgeeks.org

y para probar rápidamente fragmentos de código se ha empleado:

10.https://www.jdoodle.com/online-java-compiler/