**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. VERIFICADORES DE INTEGRIDAD EN LA TRANSMISIÓN PUNTO-PUNTO PARA ENTIDAD FINANCIERA

## *Introducción*

Existen numerosas API en los lenguajes de programación, comandos en los sistemas operativos y herramientas comerciales/software libre que se pueden utilizar para comprobar la integridad de los datos en el almacenamiento mediante la correspondiente generación de resúmenes de mensajes (message digests o checksums). En este **Proyecto de Aseguramiento de la Información** usaremos técnicas para poder verificar **la integridad de datos en la transmisión por redes públicas como Internet y evitar los diferentes tipos de posibles ataques.**

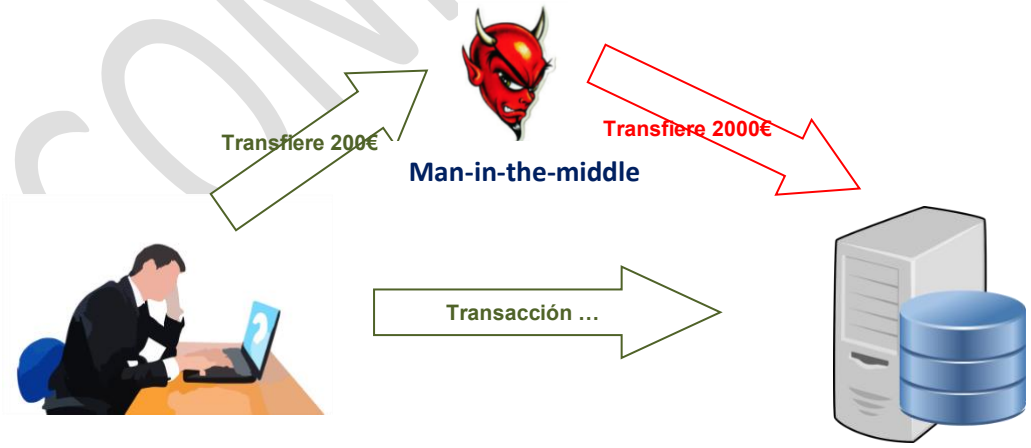
En este PAI, una entidad financiera nos ha comunicado que en su **Política de Seguridad sobre la Integridad de las Transmisiones** establece que:

*"En todas las transferencias bancarias **por medios electrónicos no seguros se debe conservar la integridad** y confidencialidad de las comunicaciones"*

Por ejemplo, para esta entidad el mensaje que se envía entre el cliente y el servidor de la organización financiera tiene el siguiente formato:

"***Cuenta origen, Cuenta destino, Cantidad transferida***"

Como primer paso para cumplir esta parte de la Política de Seguridad, la organización se propone **dar respuesta solamente al requisito de la integridad de la información transmitida.**

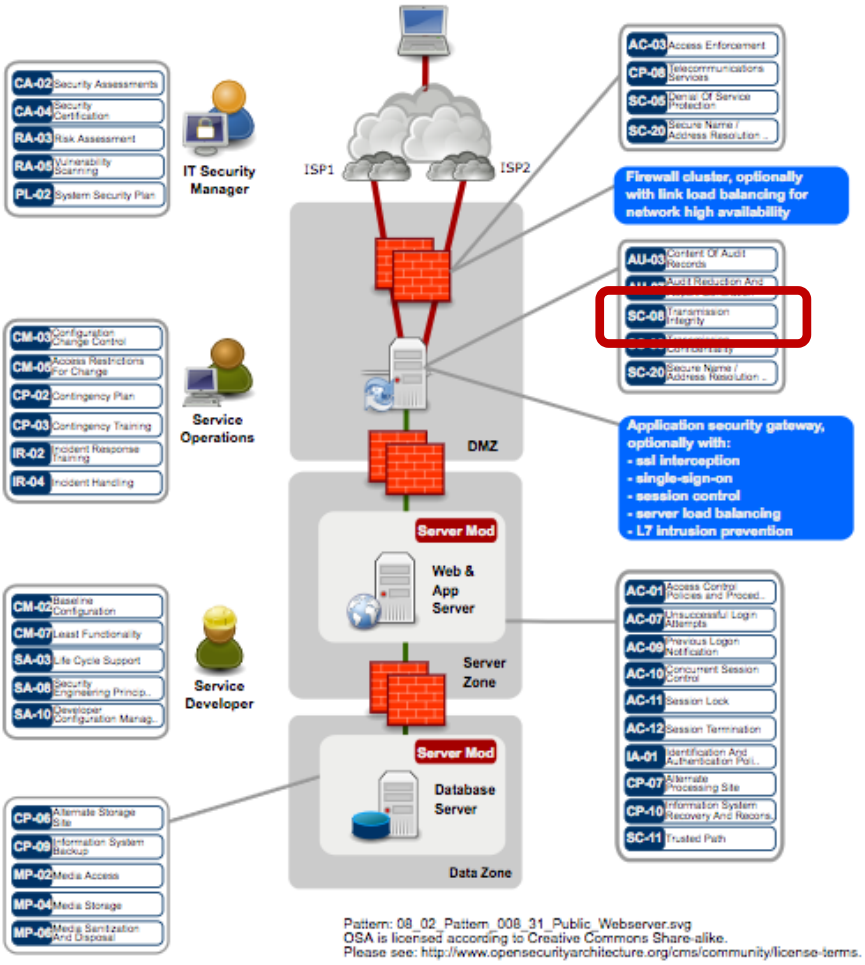



Esta política está bien soportada por la ***Open Security Architecture (OSA)*** donde se define el patrón de seguridad ***SP-008: Public Web Server Pattern***, donde para una infraestructura con un servidor público, se describe el control ***SC-08 Transmision integrity:***

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

## SP-008: Public Web Server Pattern

Diagram:




Pattern: 08_02_Pattern_008_31_Public_Webserver.svg
OSA is licensed according to Creative Commons Share-alike.
Please see: http://www.opensecurityarchitecture.org/cms/community/license-terms.

## *Objetivo del proyecto*

Por tanto, 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.**

## *Propuesta de la Dirección - Recomendaciones*

Para llevar a cabo estos objetivos se debe tener en cuenta que:

- **Input del sistema:** El **mensaje (Cuenta Origen, Cuenta Destino, Cantidad)** a verificar la integridad en su transmisión, nombre del **algoritmo** que se usará para verificar la integridad, **clave utilizada por el cliente y el servidor**.

- **Output del sistema:** Indicación en el cliente y servidor si se ha conservado la integridad o no se ha conservado. La salida podría ser presentada en una ventana al emisor del mensaje y en el servidor dejar constancia en **un fichero de logs de los mensajes que no han llegado de forma íntegra.**

**El KPI que hay que informar al Gobierno de la Seguridad de la Información de la organización financiera será la ratio de mensajes que se envían de forma íntegra/número total de mensajes enviados entre los usuarios y la entidad financiera.**

**Verificación de integridad en transmisión de información.**

Se puede realizar en cualquier lenguaje de programación o mediante cualquier herramienta de terceros. Pero **como prueba de concepto** en Java se podría hacer **mediante un Servidor que compruebe la integridad de la siguiente manera.**

# Servidor

```
public class IntegrityVerifierServer {

  private ServerSocket serverSocket;
   ……..
 // Constructor del Servidor
  public IntegrityVerifierServer() throws Exception  {
   // ServerSocketFactory para construir los ServerSockets
   ServerSocketFactory socketFactory = ( ServerSocketFactory ) ServerSocketFactory.getDefault();
    // Creación de un objeto ServerSocket escuchando peticiones en el puerto 7070
   serverSocket = (ServerSocket )  socketFactory.createServerSocket(7070);
  }

  // Ejecución del servidor para escuchar peticiones de los clientes
 private void runServer(){
     while (true) {
        // Espera las peticiones del cliente para comprobar mensaje/MAC
        try {
             System.err.println( "Esperando conexiones de clientes...");
            Socket socket =  ( Socket ) serverSocket.accept();
             // Abre un BufferedReader para leer los datos del cliente
             BufferedReader input = new BufferedReader(new InputStreamReader(socket.getInputStream()));
           // Abre un PrintWriter para enviar datos al cliente
             PrintWriter output = new PrintWriter(new OutputStreamWriter(socket.getOutputStream() ) );
            // Se lee del cliente el mensaje y el macdelMensajeEnviado
            String mensaje = input.readLine();
           // A continuación habría que calcular el mac del MensajeEnviado que podría ser
            String macdelMensajeEnviado = input.readLine();
            //mac del MensajeCalculado
              ……..
      if (macMensajeEnviado.equals(macdelMensajeCalculado)) {
          output.println( "Mensaje enviado integro " );
       }
       else {
          output.println( "Mensaje enviado no integro." );
       }
```

```
      output.close();
      input.close();
      socket.close();
      }
      catch ( IOException ioException ) {   ioException.printStackTrace(); }
       }
    }
```

```
// Programa principal
public static void main( String args[] ) throws Exception
  {
    IntegrityVerifierServer server = new IntegrityVerifierServer();
    server.runServer();
  }
}
```

## Cliente

Y **la parte cliente que envía el mensaje a través de Internet** podría ser en el lenguaje de programación Java:

```
import java.io.*;
import javax.swing.*;
import javax.net.*;

public class IntegrityVerifierClient {
    // Constructor que abre una conexión Socket para enviar mensaje/MAC al servidor
   public IntegrityVerifierClient(){
     try {
          SocketFactory socketFactory =  ( SocketFactory ) SocketFactory.getDefault();
          Socket socket = ( Socket ) socketFactory.createSocket("localhost", 7070 );
          // Crea un PrintWriter para enviar mensaje/MAC al servidor
           PrintWriter output = new PrintWriter(new OutputStreamWriter( socket.getOutputStream() ) );
           String userName = JOptionPane.showInputDialog(null,"Introduzca su mensaje:" );
           // Envío del mensaje al servidor
           output.println(mensaje );
           // Habría que calcular el correspondiente MAC con la clave compartida por servidor/cliente
           output.println(macdelMensaje);
          // Importante para que el mensaje se envíe
          output.flush();
          // Crea un objeto BufferedReader para leer la respuesta del servidor
         BufferedReader input = new BufferedReader(new InputStreamReader( socket.getInputStream ()) );
        // Lee la respuesta del servidor
         String respuesta = input.readLine();
         // Muestra la respuesta al cliente
         JOptionPane.showMessageDialog( null, respuesta);

        // Se cierra la conexion
        output.close();
        input.close();
        socket.close();
    } // end try
     catch ( IOException ioException ) {
            ioException.printStackTrace();
     }

      // Salida de la aplicacion
     finally {
       System.exit( 0 );
     }
   }
```

```
// ejecución del cliente de verificación de la integridad
  public static void main( String args[]){
    new IntegrityVerifierClient();
  }
}
```

Finalmente, y como punto crucial se debe detallar el procedimiento que ha llevado a cabo para **que el cliente y servidor tengan la misma clave para hacer la comprobación de la integridad y discuta la eficiencia de dicho procedimiento.**

## *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-SecTeam*Num*.zip (donde *Num* es el número del Security Team)**, 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 10 páginas).
  - ✓ **Código fuente de las posibles implementaciones o scripts desarrollados o configuraciones establecidas en herramientas ya disponibles.**
- El plazo de entrega de dicho proyecto finaliza el **día 20 de marzo a las 15:30 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 5% 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. Toda entrega realizada por estos medios conllevará una penalización en la entrega del 5%.**

## *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:

- **Resumen (30%)**
  - Tamaño del informe
  - Calidad del informe aportado y justificaciones
  - Calidad de pruebas presentadas y resultados
- **Solución aportada (70%)**
  - Cumplimiento de requisitos establecidos
  - Calidad del código entregado
  - Complejidad de la solución para evitar ataques MiTM y Replay
  - **Eficiencia de la solución propuesta y el reparto de claves**
  - Recolección de métricas e informes al Gobierno de la Seguridad de la Información
  - Pruebas realizadas