UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

DEPARTAMENTO DE ELECTRÓNICA

ELO 322: REDES DE COMPUTADORES I

ENVÍO DE E-MAIL POR MEDIO DE SMTP

Alumnos	
Ariel Mancilla G.	2521040 - 9
Daniel Spataris J.	2521029 - 8
Profesor	
Agustín J. González	

Fecha: 26 de Junio

INTRODUCCIÓN

En el presente informe se dará a conocer un mecanismo para el envío de e-mail por medio de SMTP, utilizando programación JAVA. A través de este informe se logrará mostrar al lector en forma práctica el uso del protocolo SMTP perteneciente a la capa de aplicación, se demostrará la vulnerabilidad en algunos servidores de correo frente al uso indiscriminado de SPAM en los recipientes de sus usuarios, se dará a conocer la facilidad de usurpar nombres y correos de otros usuarios a la hora de enviar un e-mail, se mostrará un programa en JAVA con el cual se podrá hacer envío de un e-mail por medio de SMTP.

Es importante hacer un llamado al lector sobre el uso de esta información, puesto que este trabajo nos invita a conocer la vulnerabilidad de algunos servidores de correo a la hora de pensar en la seguridad de los correos, por tanto, se deja hecha la invitación a trabajar por disminuir la vulnerabilidad expuesta por medio del presente informe, en vez de sacar partido malicioso de ella.

SMTP

SMTP (Simple Mail Transfer Protocol), es un protocolo entre servidores de correo, para el envío de mensajes e-mail, el cual pertenece a la capa de aplicación. SMTP se basa en el modelo cliente-servidor, donde un cliente envía un mensaje a un destinatario.

ENVÍO DE E-MAIL POR MEDIO DE SMTP

Programación en JAVA

Para realizar el envío de un e-mail por medio de SMTP, lo haremos por intermedio de JAVA, que es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. Cabe destacar que por intermedio de Java, se muestra una mejora sobre el envío del e-mail por medio de SMTP, ya que es posible también realizar el envío manualmente, es decir, paso a paso, utilizando TELNET (Telecomunication Network) y tecleando por pantalla los datos requeridos a medida que el servidor va cumpliendo las tareas anteriores.

Programa Java para el envío de e-mail por SMTP

En primer lugar es necesario tener un editor de archivos .java. En este caso usaremos el editor de texto de Windows y guardaremos el archivo como .java, con el fin de demostrar la sencillez de utilizar este mecanismo.

Al comenzar la creación de nuestro programa Java, es necesario importar las librerías que se van a utilizar durante el proceso, con el fin de acudir a ellas cuando sea

necesario, evitando crear un programa más engorroso y complicado. También se define el archivo .class a crear, luego de compilar nuestro programa.

```
import java.net.*;
import java.io.*;
import java.util.*;
import java.text.*;

class ELO322 {
 public static void main( String[] args ) {
```

Teniendo lo anterior resuelto, para simplicidad al momento de hacer cambios en los datos del servidor, puerto (25 para SMTP), emisor y receptor del e-mail, creamos las siguientes cadenas y el entero puerto:

```
String servidor = "XXX";
String usuario_receptor = "ariel@XXX.cl";
String usuario_emisor = "alumnos@usm.cl";
int puerto = 25;
```

Por cuestiones de seguridad, y para prevenir que el servidor facilitado para hacer las pruebas de este proyecto, incremente la recepción de SPAM (correo basura), habrá información que no será especificada durante el desarrollo de este informe (por ejemplo, el nombre del servidor).

A continuación se establece la conexión por intermedio de un socket, especificando el servidor y puerto SMTP.

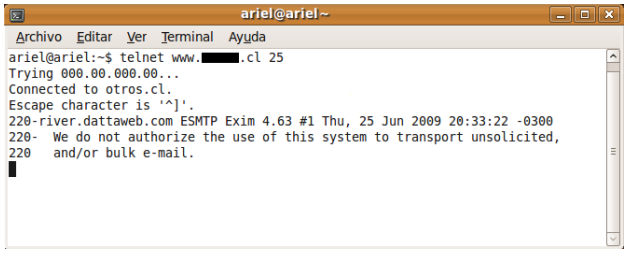
```
try {
  Socket socket = new Socket( servidor,puerto );
```

Es necesario también establecer tanto el canal de entrada, como el de salida para la comunicación.

```
BufferedReader entrada = new BufferedReader(
 new InputStreamReader( socket.getInputStream() ) );
PrintWriter salida = new PrintWriter(
 new OutputStreamWriter( socket.getOutputStream() ),true );
```

Para el siguiente paso, es necesario saber cuantas líneas de respuestas nos dará el servidor luego de completados los pasos anteriores, ya que no necesariamente todos los servidores responden la misma cantidad de líneas, y debemos saber cuando el servidor ha terminado de enviar datos, y se encuentra a la espera de los datos enviados por el cliente (en este caso, nosotros). Para lo anterior, ocupamos el comando telnet, el nombre del servidor, y el puerto por el cual nos estamos comunicando. Esto se puede realizar tanto en Windows como en Linux (en este caso se utiliza Linux para mostrar la facilidad con ambos sistemas operativos).

Generándonos como respuesta lo siguiente:



De la figura anterior podemos ver que las líneas que andamos buscando comienzan con el número 220, dato con el cual podemos observar que serán 3 las líneas de respuesta del servidor.

Teniendo en cuenta lo obtenido en el paso anterior, podemos continuar con nuestro programa en java, escribiendo el siguiente código que estará a la espera de las 3 líneas que enviará el servidor con el cual se establece la conexión.

```
System.out.println( entrada.readLine() );
System.out.println( entrada.readLine() );
System.out.println( entrada.readLine() );
```

Luego que el programa detecte las 3 líneas que espera del servidor, continuará con la ejecución del resto de las líneas de código.

A continuación se necesita enviar el emisor y el receptor del e-mail, esperando las respectivas respuestas del servidor, sobre el ingreso de los datos. Es importante mencionar que el receptor debe ser una cuenta perteneciente al servidor, y que el emisor, no necesariamente tiene que tener una cuenta en el servidor (es más, dado que no hay una verificación de estos datos, podrían ser inventados o no necesariamente pertenecer a quien los está utilizando).

```
salida.println( "mail from: "+ usuario_emisor );
System.out.println( entrada.readLine() );
salida.println( "rcpt to: "+ usuario_receptor );
System.out.println( entrada.readLine() );
```

Posteriormente se habilita el llenado de la estructura del e-mail, enviando la palabra “data” al servidor y esperando la respectiva respuesta del servidor.

```
salida.println( "data" );
System.out.println( entrada.readLine() );
```

Se crea la cadena timeStamp, donde se guardara información de la fecha, la cual será necesaria en la información que debe llevar el e-mail.

```
String timeStamp = (new SimpleDateFormat("EEE, dd MMM yyyy HH:mm:ss z")).format(new Date());
```

Se procede a llenar el encabezado, donde se especificará from, to, subject y date (de, para, asunto y fecha).

```
salida.println( "from: ELO322 <ELO.322@elo.utfsm.cl>" );
salida.println( "to: Redes de Computadores I <ELO.322@usm.cl>" );
salida.println( "subject: Envío de e-mail por medio de SMTP");
salida.println( "date: "+ timeStamp );
```

Es importante notar que los datos incluidos en dicho encabezado, no necesariamente, para efectos funcionales, deben ser reales o pertenecer a quien los está utilizando.

Luego se especifica el mensaje que se quiere enviar, en este caso enviaremos el siguiente mensaje “Se hace envío de mail mediante SMTP”.

```
salida.println( ".Se hace envío de mail mediante SMTP.");
```

Posteriormente se finaliza el mensaje enviando “.” (un punto) al servidor, y se espera su respectiva respuesta:

```
salida.println( "." );
System.out.println(entrada.readLine());
```

Luego de finalizado el envío de nuestro e-mail, procedemos a cerrar el socket. También se especifica unas líneas de código para que en caso de no lograr la comunicación con el servidor, se imprima por pantalla “No se pudo conectar con el servidor”, y así saber que el motivo del no envío del e-mail fue por esta causa.

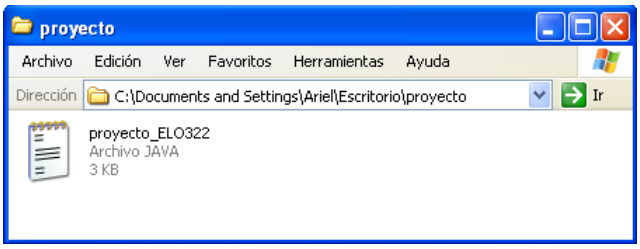
```
    socket.close();
   } catch( UnknownHostException e ) {
    e.printStackTrace();
    System.out.println(
     "No se pudo conectar al servidor indicado." );
   } catch( IOException e ) {
    e.printStackTrace();
   }
  }
}
```

Luego se procede a guardar el programa con extensión .java.

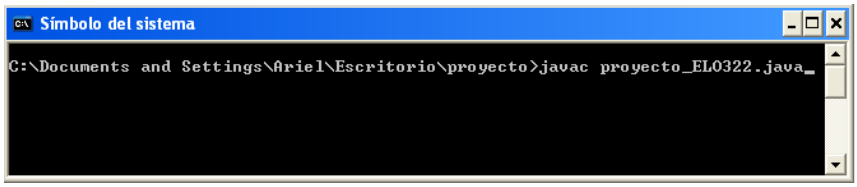
Envío de e-mail por medio de SMTP usando programación JAVA

Luego de crear nuestro archivo con extensión .java, debemos contar con algún programa que compile nuestro archivo .java. En nuestro caso hemos usado JAVA EE 5 SDK de Sun Microsystems (Esto queda a criterio del lector) y configurado la variable de ambiente PATH.

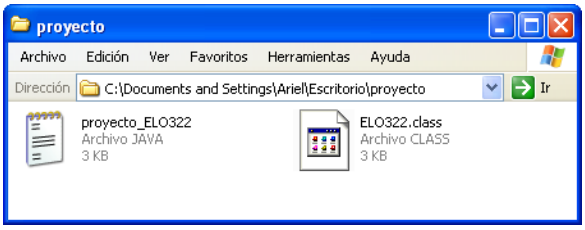
Localizamos el archivo y su ruta, para poder acceder en ella desde la consola de Windows.



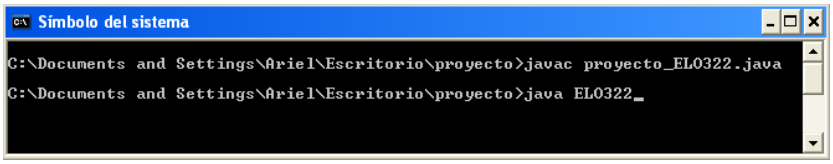
Estando en la consola de windows, nos posicionamos en la carpeta donde esta el archivo y compilamos el archivo .java creado anteriormente.



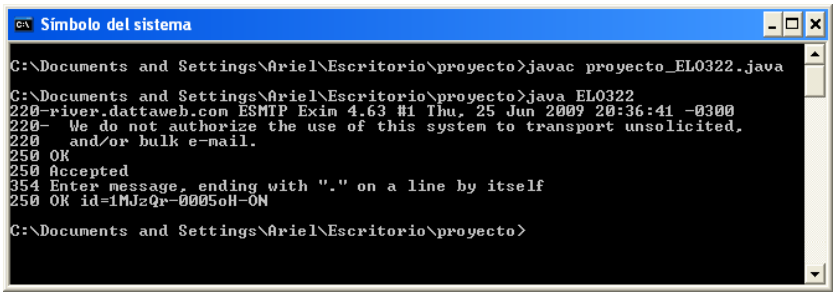
En la carpeta donde se encuentra el archivo .java, se creará un archivo .class con el nombre especificado en el programa.



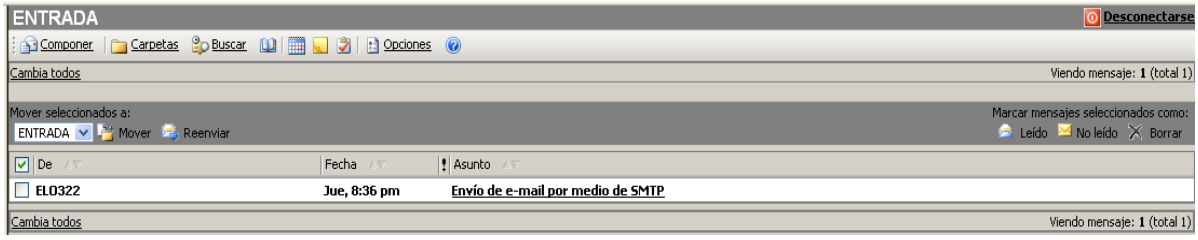
Posteriormente procedemos a ejecutar nuestro archivo .class creado recientemente.



Con lo cual se comenzará a desplegar por pantalla la comunicación entre el cliente y el servidor.

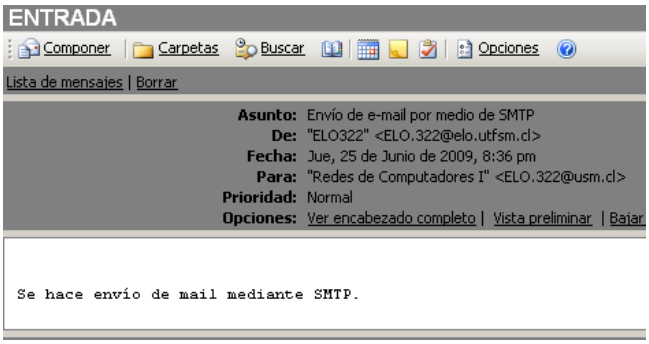


Luego de realizar lo anterior, se verifica que el correo efectivamente haya sido enviando, por tanto, entramos a la bandeja de entrada del correo (en este caso nos hemos enviado un e-mail a nuestra cuenta, con el fin de verificarlo).



Como era de esperarse, el correo se encuentra en nuestra bandeja de entrada, mostrando el from (de), el date (fecha) y subject (asunto), que anteriormente se había configurado en el programa.

Posteriormente revisamos el e-mail recibido:



Y corroboramos que todos los datos especificados en el programa java estén de acuerdo a lo que se esperaba, logrando así enviar un e-mail por medio de SMTP, utilizando programación Java.

CONCLUSIÓN

El envío de e-mail por este medio sólo verifica que el receptor se encuentre en la base de datos del servidor, por tanto lo vuelve vulnerable para ser ocupado en el envío de SPAM, usurpación de nombre y usurpación de correos electrónicos, ya que sólo basta con colocar el nombre y el sistema no corrobora la validez de dicha información, tal como se hizo en los ejemplos, donde se ocupa el nombre de usuarios del servidor de la USM, cuyos nombres y correos no necesariamente existen.

Es posible probar con distintos mecanismos para el envío mediante SMTP, en este informe se observó el envío sin autenticación, sin embargo, existen servidores que tienen un grado más alto de seguridad exigiendo la autentificación, la cual puede ser realizada por medio del comando “perl”.

Por otra parte, se expone un programa en Java, que facilita el envío de e-mail, pudiendo ser mejorado ofreciendo una interfaz gráfica donde el usuario ingrese los datos y el resto sea parte de la programación.

REFERENCIAS

Simple Mail Transfer Protocol (SMTP). Obtenido el 15 de mayo de 2009 en http://es.wikipedia.org/wiki/SMTP.

Cliente SMTP. Obtenido el 19 de junio de 2009 en http://www.itapizaco.edu.mx/paginas/JavaTut/froufe/parte20/cap20-11.html.

ANEXO

Programa utilizado

```
import java.net.*;
import java.io.*;
import java.util.*;
import java.text.*;

class ELO322 {
 public static void main( String[] args ) {

   // se establecen las cadenas con información del
   // servidor, usuario receptor y emisor del e-mail
   // y puerto de conexión SMTP.
   String servidor = "XXX";                          // nombre del servidor.
   String usuario_receptor = "ariel@XXX.cl";   // correo del receptor e-mail.
   String usuario_emisor = "alumnos@usm.cl"; //correo del emisor del e-mail.
   int puerto = 25;                                  // 25 es el puerto para SMTP.

   try {

    // Se establece conexión abriendo un socket,
    // especificando el servidor y puerto SMTP.
    Socket socket = new Socket( servidor,puerto );

    // Se consigue el canal de entrada.
    BufferedReader entrada = new BufferedReader(
     new InputStreamReader( socket.getInputStream() ) );

    // Se consigue el canal de salida.
    PrintWriter salida = new PrintWriter(
     new OutputStreamWriter( socket.getOutputStream() ),true );

    // Se consigue captar las líneas de respuestas enviadas por
    // el servidor luego de establecer la conexión.
    System.out.println( entrada.readLine() );
    System.out.println( entrada.readLine() );
    System.out.println( entrada.readLine() );

    // Se inicia la conversación con el servidor.

    // Se establece emisor y receptor del correo.
    salida.println( "mail from: "+ usuario_emisor ); // remitente
    System.out.println( entrada.readLine() );
    salida.println( "rcpt to: "+ usuario_receptor );   // destinatario
    System.out.println( entrada.readLine() );

    // Se habilita el llenado de la estructura del e-mail.
    salida.println( "data" );
    System.out.println( entrada.readLine() );

    // Se establece una cadena para guardar información horaria.
    String timeStamp = (new SimpleDateFormat("EEE, dd MMM yyyy HH:mm:ss z")).format(new
Date());

    // Se llena el encabezado del mensaje.
    salida.println( "from: ELO322 <ELO.322@elo.utfsm.cl>" );              // de:
    salida.println( "to: Redes de Computadores I <ELO.322@usm.cl>" );   // para:
    salida.println( "subject: Envío de e-mail por medio de SMTP");          // asunto:
    salida.println( "date: "+ timeStamp );                                          // fecha:

    // Se escribe el mensaje a enviar.
    salida.println( ".Se hace envío de mail mediante SMTP.");

    // Se finaliza el mensaje.
    salida.println( "." );
    System.out.println(entrada.readLine());

    // Se cierra el socket
    socket.close();
   } catch( UnknownHostException e ) {
    e.printStackTrace();
    System.out.println(
      "No se pudo conectar al servidor indicado." );  // mensaje que se arroja cuando no se logra
   } catch( IOException e ) {                          // la conexión con el servidor.
    e.printStackTrace();
   }
 }
}
```