Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
Curso de Redes I CI 4835
Trimestre Enero-Marzo 2012

Segundo Proyecto (10 %)

El objetivo general de este proyecto es desarrollar una aplicación cliente/servidor, a varios niveles, describiendo el protocolo de comunicación.

Objetivos específicos:

- Programar aplicaciones distribuidas en Lenguaje Java usando *sockets*
- Diseñar un protocolo para la comunicación de procesos usando XML.
- Desarrollar una arquitectura que sea tolerante a fallas de los servidores.
- Llevar estadísticas de consultas en servidores.

Enunciado

Actualmente los requerimientos de seguridad de muchas instituciones exigen el uso de certificación digital. Este mecanismo requiere que cada usuario cree una clave pública y una clave privada. La clave privada es bien resguardada por el usuario y la clave pública debe ser expuesta para que otros puedan utilizarla y enviar mensajes cifrados al propietario de dicha clave. No obstante, la clave pública, por estar expuesta, debe ser avalada por un tercero conocido como autoridad de certificación.

Una autoridad de certificación es un ente, público o privado, que certifica los datos del propietario de la clave pública y le da su aval firmándola digitalmente. Así todos los datos del propietario: nombre, institución, clave pública, etc. Junto con la firma de la autoridad de certificación, son incluidos dentro de una plantilla que sigue el formato X.509. Este estándar es respetado por todas las autoridades de certificación, navegadores Web y cualquier aplicación cliente/servidor.

Para este proyecto se pretende que Uds desarrollen una aplicación cliente/servidor que permita recuperar certificados digitales que están almacenados en repositorios. Inicialmente el cliente desconoce en que repositorio está el o los certificado(s) solicitado(s), pero tiene información del propietario que le permitirá a nuestra aplicación localizarlo entre los distintos repositorios de certificados.

En primer lugar deben desarrollar una aplicación cliente (que llamaremos simplemente `clicert`) que se conectará con un buscador (que denominaremos `buscert`) el cual conoce los servidores (`sercert`) los cuales almacenan los certificados digitales de una organización. Estos servidores deben inscribirse ante `buscert` al levantarse (iniciar su ejecución) ya que conocen su localidad. Los clientes de la aplicación también conocen la localidad de `buscert` y pueden hacer sus consultas por cualquiera de los campos que contengan los certificados. Las consultas pueden ser por varios campos que estarán conectados con un AND.

El lenguaje de consultas debe estar especificado en XML y queda de parte de Uds. definir las etiquetas que estimen necesarias y convenientes. También deben definir un protocolo que especificarán mediante diagramas de secuencia.

La sintaxis de la invocación de cada uno de los programas antes descritos es la siguiente:

Programa clCert

```
clicert –d <directorio de certificados descargados> -h <nombre de dominio, o dirección ip de buscert> [-p <puerto de buscCert>]
```

`<directorio de certificados descargados>`: contiene el camino absoluto o relativo del directorio donde estarán los certificados digitales descargados por el cliente

`<nombre de dominio , o dirección ip de buscert >`: máquina donde se encuentra corriendo el buscador

<puerto>: puerto de escucha del buscador `buscert`. Al ser opcional si no se indica ningún puerto el buscador deberá estar montado sobre el puerto 4000.

Una vez que esté corriendo `clicert` puede recibir por teclado los patrones de búsqueda, uno por línea. Se debe indicar los nombres de los campos, separados por blancos o tabuladores, y justo después su valor todos conectados por un AND lógico (no colocado explícitamente). Ejemplo:

```
Issuer: Thawte Subject: www.freesoft.org
```

`clicert` podrá recibir tantas búsquedas como el cliente lo desee y sólo se detendrá al colocar dos saltos de línea consecutivos.

El estándar usado por los certificados (X509) respeta las normas ASN.1 y especifica los siguientes campos:

- Versión
- Número de serie
- ID del algoritmo
- Emisor
- Validez
- Sujeto
- Información de la clave pública
- Varios campos opcionales ...
- Algoritmo usado para firmar
- Firma digital del certificado

Ejemplo de un viejo certificado de `www.freesoft.org` firmado por una autoridad de certificación ya desaparecida (en realidad adquirida por `Verisign`) conocida como `Thawte`:

```
Certificate:
   Data:
       Version: 1 (0x0)
       Serial Number: 7829 (0x1e95)
       Signature Algorithm: md5WithRSAEncryption
       Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
               OU=Certification Services Division,
               CN=Thawte Server CA/Email=server-certs@thawte.com
       Validity
           Not Before: Jul  9 16:04:02 1998 GMT
           Not After : Jul  9 16:04:02 1999 GMT
       Subject: C=US, ST=Maryland, L=Pasadena, O=Brent Baccala,
                OU=FreeSoft, CN=www.freesoft.org/Email=baccala@freesoft.org
       Subject Public Key Info:
           Public Key Algorithm: rsaEncryption
           RSA Public Key: (1024 bit)
               Modulus (1024 bit):
                   00:b4:31:98:0a:c4:bc:62:c1:88:aa:dc:b0:c8:bb:
                   33:35:19:d5:0c:64:b9:3d:41:b2:96:fc:f3:31:e1:
                   66:36:d0:8e:56:12:44:ba:75:eb:e8:1c:9c:5b:66:
                   70:33:52:14:c9:ec:4f:91:51:70:39:de:53:85:17:
                   16:94:6e:ee:f4:d5:6f:d5:ca:b3:47:5e:1b:0c:7b:
                   c5:cc:2b:6b:c1:90:c3:16:31:0d:bf:7a:c7:47:77:
                   8f:a0:21:c7:4c:d0:16:65:00:c1:0f:d7:b8:80:e3:
                   d2:75:6b:c1:ea:9e:5c:5c:ea:7d:c1:a1:10:bc:b8:
                   e8:35:1c:9e:27:52:7e:41:8f
               Exponent: 65537 (0x10001)
   Signature Algorithm: md5WithRSAEncryption
       93:5f:8f:5f:c5:af:bf:0a:ab:a5:6d:fb:24:5f:b6:59:5d:9d:
```

```
92:2e:4a:1b:8b:ac:7d:99:17:5d:cd:19:f6:ad:ef:63:2f:92:
ab:2f:4b:cf:0a:13:90:ee:2c:0e:43:03:be:f6:ea:8e:9c:67:
d0:a2:40:03:f7:ef:6a:15:09:79:a9:46:ed:b7:16:1b:41:72:
0d:19:aa:ad:dd:9a:df:ab:97:50:65:f5:5e:85:a6:ef:19:d1:
5a:de:9d:ea:63:cd:cb:cc:6d:5d:01:85:b5:6d:c8:f3:d9:f7:
8f:0e:fc:ba:1f:34:e9:96:6e:6c:cf:f2:ef:9b:bf:de:b5:22:
68:9f
```

Le recomendamos que sólo use un formato de certificado y sugerimos `base64` (extensión `PEM`). Hay convertidores para pasar de un formato a otro. Para generar sus certificados de prueba pueden usar `openssl` o `gpg` y posteriormente hacerlos firmar con una autoridad de prueba. Wikipedia presenta una breve descripción de los certificados X509 pero para más detalles consultar el RFC5280.

Programa buscert

```
buscert [-p <puerto>]
```

`<puerto>` corresponde al número del puerto de escucha de peticiones de `buscert`. El puerto por defecto, si esta opción no se especifica, será el puerto 4000.

.Este servidor debe llevar las siguientes estadísticas:

(1) Cantidad de certificados consultados en total
(2) Cantidad de clientes que solicitaron un certificado particular
(3) Cantidad de certificados requeridos por repositorio
(4) Lista de todos los certificados solicitados

Toda esta información se puede solicitar por pantalla desde la cónsola de `buscCert` con los siguientes comandos:

`clicert [IP del cliente]`	opción (1)
`solcli <sujeto a quien se emitió el certificado>`	opción (2)
`cerreq`	opción (3)
`liscert`	opción (4)

Programa servCert

```
sercert [-p <puerto>] –d <directorio que almacena los certificados> -h <nombre de dominio de buscert> -b <puerto de escucha de buscert>
```

`<puerto>`: corresponde al número del puerto de escucha de peticiones de `sercert`. El puerto por defecto, si esta opción no se especifica, será el puerto 5000.

`<directorio que almacena los certificados>`: contiene el camino absoluto o relativo del directorio donde estarán los certificados del repositorio

`<puerto de escucha de buscert>`: puerto de escucha del buscador que, de no ser especificado, será el 4000.

La respuesta, de `sercert` debe enviar todo el certificado digital. Además, como se mencionó al principio, cada repositorio `sercert` deben inscribirse ante el buscador indicando el nombre de dominio del repositorio y el puerto por donde está escuchando. Por último, en general, cualquiera de los servidores puede caerse por lo que los clientes deben detectar estos fallos y enviar mensajes que den un diagnóstico. En consecuencia los repositorios recibirán mensajes periódicos de `buscert` para detectar si están levantados.

Entrega del Proyecto

La entrega se realizará el día lunes de la semana 8 antes de las 12:00 am, a través de Aula Virtual.

El programa debe contener un *makefile* que permita la correcta compilación de todos sus componentes simplemente al invocar el comando *make*. Para ejecutar los programas se debe seguir de forma estricta la sintaxis especificada en el enunciado, de no ser así se considerará que el proyecto no funciona y no será corregido.

El proyecto será corregido en los equipos del LDC, específicamente en las maquinas de la sala Ernesto Leal. Si su proyecto no puede compilarse con su *makefile* o ejecutarse según sus *script*s en estos equipos, entonces no será corregido.

Si usted es un integrante del grupo X (según la numeración de aula virtual) debe generar un archivo de nombre proy2grupoX.tar.gz, generado con el comando (tar cvfz) que no contenga ningún directorio interno y que contenga todos los archivos que hagan falta para hacer la corrección, incluyendo el *makefile*. Ese debe ser el único archivo que usted coloque en aula virtual.

Para el manejo de los archivos y formatos XML solo podrá emplear las librería NanoXML Lite, que fueron colocadas en

/net/raquella/ldc/redes/nanoxml/gcj/nanoxml-lite-2.2.3.o

/net/raquella/ldc/redes/nanoxml/java/nanoxml-lite-2.2.3.jar

No incluya copias de estas librerías en la entrega de su proyecto.

El código que entregue vía aula virtual debe estar debidamente documentado, siguiendo el estándar de documentación `javadoc` y todas las llamadas al sistema deben ser correctamente manejas aplicando los conceptos y reglas de estilo de programación vistos en el curso de sistemas operativos. Para este proyecto no se está pidiendo que imprima todo el código fuente de su proyecto.

Documentación Impresa:

Se debe entregar un informe impreso, con una portada que identifique a los integrantes del equipo y con los siguientes contenidos:

- Cuál es el estado de su proyecto. (que funciona y que no funciona)
- Se debe describir los protocolos diseñados para la comunicación entre el `clicert – buscert` y para la comunicación `buscert – sercert` para cada una usar al menos un diagrama de secuencia (UML) y al menos un diagrama de estados (UML).
- Indicar claramente la sintaxis XML con la semántica asociada a cada una de las etiquetas. Para ello debe indicar la sintaxis de consulta de certificados, respuesta con los certificados y mecanismo de inscripción de los repositorios ante el buscador.

Notas Adicionales:

Si no se siguen las normas especificadas en este enunciado, su proyecto no será coregido.
Al ser realizado el proyecto en equipo, cada uno de los miembros debe conocer plenamente todos y cada uno de los detalles de implementación del proyecto y podrá ser interrogado al respecto durante la corrección del proyecto que con alta probabilidad será presencial.
Debido al incremento de casos en los que se han conseguido proyectos “similares” se realizará una comparación de todos los proyectos empleando herramientas tanto automáticas como manual especificas para este tipo de casos, por lo que recomendamos evitar cualquier tipo de práctica que conduzca a proyectos “similares” ya que esto será severamente penalizado, dejando de corregir ambos proyectos y aplicando las normas y reglamentos establecidos para este tipo de casos en la universidad.

GDRC