Actividad no recuperable de evaluación: servicio para computación

Puntuación: 20 puntos sobre 100.

Plazo de entrega: hasta las 14:00h. del miércoles 13 de enero de 2021.

Procedimiento para entrega: proyecto *ComputingService* a entregar se presenta limpio de clases compiladas (directorio *build* eliminado y directorio *dist* solo con el *script* de compilación/embalado que el autor haya preparado.); se entrega comprimido en formato *zip* por medio del sistema de tareas de *MiAulario*; solo se permite una entrega.

Objetivo de la actividad

- Proteger un prototipo muy básico de servicio de cálculo intensivo con los diferentes medios técnicos que proporciona la plataforma *Java SE* para mejorar la seguridad de las aplicaciones:
  - Autenticación de usuarios
  - Control de acceso mediante políticas de permisos
  - Ejecución privilegiada
  - Transmisión protegida mediante protocolo *SSL/TLS*.
- El prototipo adjunto es plenamente operativo.

Planteamiento

- El número de usuarios en algunos sistemas de información es tan elevado que resulta impracticable definir políticas que asignen permisos específicos a cada principal.
- En esa situación se clasifica a los usuarios en categorías con diferentes privilegios y se recurre a técnicas que por diseño prevengan del acceso a funcionalidades no autorizadas en cada categoría.
- Esta actividad sirve como prueba de concepto de la implementación de esa idea en la plataforma *Java SE*.

1. Contexto

- *PEAK Engineering and Knowledge, Inc.* es una empresa que realiza servicios de ingeniería para diversos contratistas.
  - La sede de la empresa se encuentra en la ciudad de Rochester, en el estado norteamericano de New York.
- Esta empresa se ha ganado una valiosa reputación de fiabilidad y confidencialidad en el desarrollo de los proyectos contratados.
- Los campos de actividad con mayor importancia dentro de la empresa son la simulación mediante modelos matemáticos y la minería de datos.
- Para sus cálculos de ingeniería dispone de una plataforma *hardware* de alto rendimiento a la que únicamente pueden acceder los ingenieros de computación autorizados.

- Asimismo dispone de un eficaz sistema de almacenamiento masivo al servicio de los proyectos relacionados con la minería de datos.
- Entre los ingenieros de computación se han distinguido dos categorías dependiendo del campo en el que desarrollen su actividad:
  - Ingenieros de simulación (categoría A)
  - Ingenieros de datos (categoría B)
- La experiencia y conocimiento acumulados en el campo de las simulaciones ha quedado plasmada en una serie de herramientas *software* para construcción de modelos que no requieren el almacenamiento de grandes volúmenes de datos.
- Por esa razón, sólo los ingenieros de datos tienen acceso autorizado al sistema de almacenamiento masivo.

Identificación interna

- La empresa aplica un doble sistema de identificación para acceder a la plataforma de computación de alto rendimiento:
  - Emplea *Kerberos* como infraestructura de autenticación por contraseña.
  - Cada ingeniero de computación dispone además de una *smart card* personal proporcionada por la empresa que contiene un certificado expedido al ingeniero para su identificación[1]. Ese certificado está almacenado en un *keystore* cuyo alias es igual al nombre del principal *Kerberos* del ingeniero.
- Es requisito superar ambos sistemas de autenticación para conseguir una identificación positiva.
- El reino `COMPUTINGSERVICE.PEAK.COM` incluye a todos los ingenieros de computación con acceso al sistema de computación.

Distinción de categoría

- El acceso al servicio de computación emplea además una sencilla base de datos para conocer en qué categoría (simulación o datos) está encuadrado cada ingeniero de computación perteneciente al reino.
- La base de datos consta de una única tabla denominada `ENGINEERS` formada por tres columnas:
  - `idCode`: *BINARY* de longitud 16; actúa como clave primaria; se obtiene como resumen *MD5* del nombre del principal en el reino `COMPUTINGSERVICE.PEAK.COM`.
  - `kerberosPrincipalName`: *VARCHAR* de hasta 60 elementos.
  - `cathegory`: de tipo *CHAR*; indica si el principal es ingeniero de simulación (valor *A*) o es ingeniero de datos (valor *B*).

[1] Si no se dispone del *hardware* para operar con *smart cards*, esa carencia se puede solventar empleando el directorio `./etc/x500` para simular la función de almacenamiento de la *smart card*.

Modo de operación

Parte del cliente

- Tras su autenticación, una sesión de cliente del servicio de computación consta de dos etapas:

  1. Envío al servicio del fichero con el código de la aplicación a ejecutar y del fichero con los argumentos para ejecución; el servicio está preparado para ejecutar aplicaciones *Java* embaladas en ficheros *jar*.
  2. Espera hasta la recepción de los resultados de ejecución.

Parte del servicio

- Solo el administrador de la aplicación está autorizado a ponerla en marcha.
  - Eso significa que la política de permisos de la parte de servicio solo concede los permisos necesarios al principal que actúa como administrador.
- Tras la autenticación, la aplicación entra en un bucle de escucha de peticiones; esas peticiones se espera que lleguen por el puerto 2050.
- Ante cada petición se crea un objeto de servicio que se ejecuta en hebra independiente.

2. Diseño

2.1 Clases del lado del cliente

Clase pública final *client.ClientLogin*

- Incluye el método `main()` a ejecutar del lado del cliente.
  - No lanza excepciones capturables al exterior.
- Su responsabilidad es autenticar al cliente para ejecutar la acción privilegiada que permite ponerse en contacto con el servicio de computación.

    - Crea un contexto de *login* a partir de la configuración etiquetada como `CLIENT` en el fichero *./etc/client.conf*.

- Crea un objeto de clase *ClientComputingTask* para arrancar una tarea de computación en la parte de servicio.

Clase confinada final *client.ClientComputingTask*

- Ofrece el constructor

```
ClientComputingTask (final Subject client,
                     final String jarFileName,
                     final String argsFileName,
                     final String resultsFileName)
```

- Su método `compute()` comienza por enviar al servicio el objeto *client* (se puede emplear un *ObjectOutputStream*), y seguidamente transfiere el contenido del fichero *jar* y el contenido del fichero con los argumentos apara ejecución.

- La ejecución del método termina cuando se reciben los resultados de ejecución por parte del servicio.

2.2 Clases del lado del servicio

Clase pública final *frontend.ComputingServiceLogin*

- Su responsabilidad es autenticar al principal (administrador de la aplicación) que tiene permiso para arrancar el servicio.
    - No lanza excepciones capturables al exterior.

- Crea un contexto de *login* a partir de la configuración etiquetada como `SERVICE` en el fichero *./etc/service.login.conf*.

- Emplea el sujeto autenticado para arrancar y detener la maquinaria del servicio de computación.

Clase pública final *service.ComputingService*

- Ofrece los métodos de arranque y parada del servicio.
- El código operativo de esos métodos se debe ejecutar en modo privilegiado con los permisos asignados por la política al administrador de la aplicación.

Clase confinada final *service.ComputingTask*

- Implementa la interfaz *Runnable*.
- Se encarga de poner en marcha la tarea de ejecución solicitada por un cliente.
- Al código operativo proporcionado se debe añadir lo siguiente:
  - Recepción (por medio de un objeto de clase *ObjectInputStream*) del objeto que representa al cliente autenticado.
  - Extrae el principal *Kerberos* del sujeto recibido.
  - Consulta a la base de datos la categoría asignada a ese sujeto.
  - Instanciación del objeto ejecutor de código *jar* (*jarrunner*) que corresponda a esa categoría.

Clase confinada final *service.CathegoryQuery*

- Se encarga de la consulta de la categoría asignada al ingeniero cuyo principal *Kerberos* se proporciona como argumento al contructor.
- No requiere modificaciones.

Clase pública *jarrunner.JarRunner*

- Se encarga de ejecutar en modo privilegiado el fichero *jar* cuya URL recibe como argumento con los permisos asignados por la política de permisos al administrador de la aplicación.

Clase pública final *jarrunner.JarClassLoader*

- El constructor recibe la ubicación de un fichero *jar* y un *array* de argumentos de invocación.
- Su método `run()` se encarga de invocar el método `main()` de la clase de arranque embalada en el fichero *jar*.
- No requiere modificaciones.

3. Tareas a realizar

3.1. Ejecución privilegiada

- En la parte de servicio, se debe adaptar el código para que las operaciones que requieran permisos se ejecuten con los privilegios asignados al administrador por la política de permisos.
- En la parte cliente, no hace falta realizar adaptación ninguna.
- Los *scripts* de ejecución proporcionados muestran la distribución en paquetes que ha de tener cada parte de la aplicación.
- Para la parte de servicio, esa distribución determina las cláusulas que deben incluirse en la política de permisos a definir.

3.2. Modificaciones y clases a definir

Cambios en clase *jarrunner.JarRunner*

- Convertir la clase pública en confinada al paquete.
- Confinar su constructor.
- Convertir su método `run()` en protegido.

Clase pública final *jarrunner.JarRunnerA*

- Sirve para ejecutar el código de los clientes de categoría *A*.
- Es una extensión de la clase confinada *JarRunner*.
- Su constructor

```
public JarRunnerA (final Subejct subject,
                   final String location,
                   final String[] args) throws MalformedURLException
```

  recibe el sujeto con cuyos permisos se debe ejecutar el método `run()`.
- Exporta un método `run()` que ejecuta el método `run()` de la clase *JarRunner* (`super.run()`) con los permisos asignados por la política de permisos al sujeto recibido por el constructor.

Clase pública final *jarrunner.JarRunnerB*

- Es una adaptación a los clientes de categoría *B* de lo que se ha especificado para la clase *JarRunnerA*.

Embalado por separado

- Cada clase pública del paquete *jarrunner* se ofrece embalada junto con las clase confinadas en un fichero *jar* propio.
- La política de permisos asigna a ese fichero *jar* los permisos adecuados a las operaciones que se autorizan a los usuarios de la correspondiente categoría del servicio de computación.

3.3. Identidades X.500 expedidas por autoridad de certificación local

- La empresa dispone de una autoridad de certificación que expide credenciales X.500 para los empleados y servicios de la empresa.
- Los principales

ligeti@COMPUTINGSERVICE.PEAK.COM
geyer@COMPUTINGSERVICE.PEAK.COM

son dos ingenieros de la empresa que emplean el sistema de computación.

  - *ligeti* pertenece a la unidad organizativa *Simulation Department*.
  - *geyer* pertenece a la unidad organizativa *Data Department*

- A ambos la autoridad de certificación de la empresa les ha proporcionado un certificado de clave pública que emplean para su autenticación en el sistema.
- Ambos poseen un *keystore* personal (*ligeti.keystore* y *geyer.keystore*) que contiene el certificado propio expedido por la autoridad local y el certificado que contiene la clave pública del servicio de computación.
- El servicio de computación guarda en *service.keystore* el certificado que para su identificación ha expedido la autoridad de certificacicación; los certificados de clave pública de los ingenieros autorizados a emplear el servicio se guardan en el depósito *service.truststore*.
- La constraseña de acceso a todos los depósitos de claves debe ser *PS2021*.

3.4. Comunicación protegida

- La comunicación entre clientes y servicio de computación se debe proteger incorporando el protocolo *TLS/SSl*.
- El departamento de seguridad de la empresa ha establecido que ambas partes (cliente y servicio de computación) deben identifcarse una frenta a la otra.

4. Rúbrica de evaluación

- En revisión se van a considerar los siguientes apartados de puntuación
  - Ficheros de configuración de *login*: hasta 0,75 puntos
  - Ficheros con las políticas de permisos: hasta 2 puntos
  - Contenido de *keystores*: hasta 2 puntos.
  - Modificación de código entregado e inserción con criterio de acciones privilegiadas: hasta 3 puntos
  - Definición de nuevas clases ejecutoras: hasta 0,75 puntos.
  - Modificaciones necesarias en código y *scripts* de ejecución para operar con protocolo *TLS/SSL*: hasta 1 punto
  - Construcción y embalado mediante *script* de componentes de la aplicación: hasta 1,5 puntos
  - Operatividad completa de la aplicación al ser ejecutada desde línea de órdenes: hasta 9 puntos.
  - Proyecto con código compilado: -2 puntos
- Se va a poner atención a la calidad y buena terminación de lo entregado.