Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
Laboratorio de Redes de Computadores (CI-4835)

2do Proyecto: Programación con RPC

Descripción general

PDVSA desea mejorar el sistema de distribución de gasolina que usted desarrolló como primer proyecto del curso y desea, que esta nueva implementación del prototipo, sea desarrollada en lenguaje C y empleando el paradigma de programación RPC.

Primera parte:

Para lograr esto se espera que, en una primera etapa, desarrolle el mismo sistema anterior pero usando RPC. Es decir, el software para las bombas de gasolina (clientes) y los centros de distribución de combustible (servidores). Se mantendrán así las órdenes "*bomba*" y "*centro-n*" con las especificaciones previamente indicadas, a excepción de la opción "p" que indica el puerto de socket.

Segunda parte:

En una segunda fase, se desea que al establecer la conexión entre cualquier bomba y los centros de suministro, instrumente un sistema de autenticación simple que permita determinar con quien se interactúa. Dicha autenticación se hará solamente para determinar la identidad de la estación de gasolina que desea ser atendida. Si dicha identificación resulta correcta, el centro otorgará un ticket de atención que da derecho al servicio de suministro de gasolina. Para ello, seguidamente se le provee de la información sobre un esquema básico de autenticación denominado "*desafío-respuesta*".

Esquema de Autenticación Desafío – Respuesta: este mecanismo distribuido consiste en una comunicación de datos preliminar que sucede entre dos partes. El ente que va a verificar la identidad de la otra parte inicialmente le envía un desafío o reto (*challenge*), del cuál ya conoce la respuesta (*response*). Luego, espera que su contra parte le responda la misma respuesta que el ya conoce. Como ambas entidades, previamente, acordaron la forma de producir la respuesta correcta para cualquier desafío que se suministre, entonces el resultado les permite determinar si están interactuando con su par correcto. Es decir, si ambas respuestas coinciden dan por válida las identidad de quien desea ser atendido.

Es común además, que la comunicación entre las partes se haga a través de un canal privado, reduciendo así la posibilidad de ataques de hombre en el medio, pero para simplificar la programación de este proyecto, esta importante característica de protección se ignorará.

Por otra parte, el esquema desafío - respuesta es algo más confiable que la clásica autenticación basada en contraseña, debido a que se espera que el desafío varíe en el tiempo, pero ello no requiere que se intercambien las respuesta, ni que se memorice algún secreto, dado que siempre existe un algoritmo para calcular la respuesta correcta. En otras palabras, la seguridad se soporta sobre el algoritmo compartido, el valor de turno a proveerle como entrada y el canal confiable de comunicación. En este proyecto se empleará la función picadillo (*hash*) MD5 (*resumen del mensaje número 5*) del profesor Ron Rivest, como el algoritmo que producirá la respuesta apropiada. Para ello se le suministrará un archivo compacto con el código en lenguaje C ya programado (*md5-c-100.tar.gz*)

MD5: Es un algoritmo que se comporta como una función biyectiva, y para cualquier posible entrada de datos de tamaño variable, produce 128 bits de salida únicos. En el código que se le suministra, la orden puede ser usada de varios modos. Por ejemplo, con una entrada a partir de una cadena de caracteres o de un archivo. A continuación se muestra algunas de las sintaxis:

$ **./md5 -s"Hola ¿como está usted?"**

MD5 ("Hola ¿como está usted?") = a708822fff65098430846314c8f92d4b

$ **./md5 ingenieria_seguridad.pdf**

MD5 (ingenieria_seguridad.pdf) = 00931fd940fa7abc73b8583ea278cc05

$

Para completar el desafío-respuesta únicamente falta especificar como se debe producir el reto, y ello se efectuará a través de un número pseudoaleatorio.

Generación del desafío: para la segunda etapa, cuando el centro de suministro reciba la petición de atención de un servicio, deberá responder suministrando un valor pseudoaleatorio a la bomba que llamó, para que esta pueda producir la respuesta adecuada y al recibirla, comparará con su propio resultado. En caso de que los resultados concuerden, atenderá la petición, de lo contrario rechazará la conexión.

Recuerde además, que para que algunos algoritmos establezcan una serie de valores pseudo aleatorios, requieren inicialmente operar con una semilla de datos. Las funciones para implementar los valores pseudoaleatorios quedan sujetas a su voluntad.

Por otra parte, toda autenticación realizada tiene una lapso de vigencia y en este proyecto se espera que ese control sea manejado a través de un ticket de atención que tendrá tiempo limitado.

Ticket de servicio: La autenticación que usted va a implementar, será persistente por cada suministro de gasolina. Escrito de otro modo, si una bomba requiere suministro de gasolina, deberá identificarse solamente una vez en forma correcta para poder recibirlo. En caso de que la conexión falle, o de que la última autenticación haya sido efectuada, al menos cinco (5) minutos antes de la hora actual -tiempo de las ocho (8) horas de la simulación-, entonces deberá repetirse el proceso. Es por ello que este requerimiento demanda que la solución que usted instrumente, opere de forma parecida a un "servidor con estado". Es decir, que el centro de servicio recuerde algunas operaciones ejecutadas cuando se sucedan diferentes conexiones.

Una forma de instrumentar este requerimiento es a través de un "ticket de servicio", el cuál estará vigente por el tiempo válido que el centro le asigne. Si se solicita la ejecución de cualquier programa remoto que conforma el servicio de suministro de gasolina, el servidor debe solicitar previamente que le entreguen un ticket válido, de lo contrario debe imponer que se ejecute una nueva autenticación que provea el ticket que demanda.

En la vida real, todo ticket de servicio debe incorporar algún mecanismo para establecer correctamente si es auténtico o no lo es. La dirección IP de cualquier máquina es un pésimo mecanismo para ello, sin embargo, por razones de simplicidad en este trabajo, se usará.

Consideraciones Adicionales

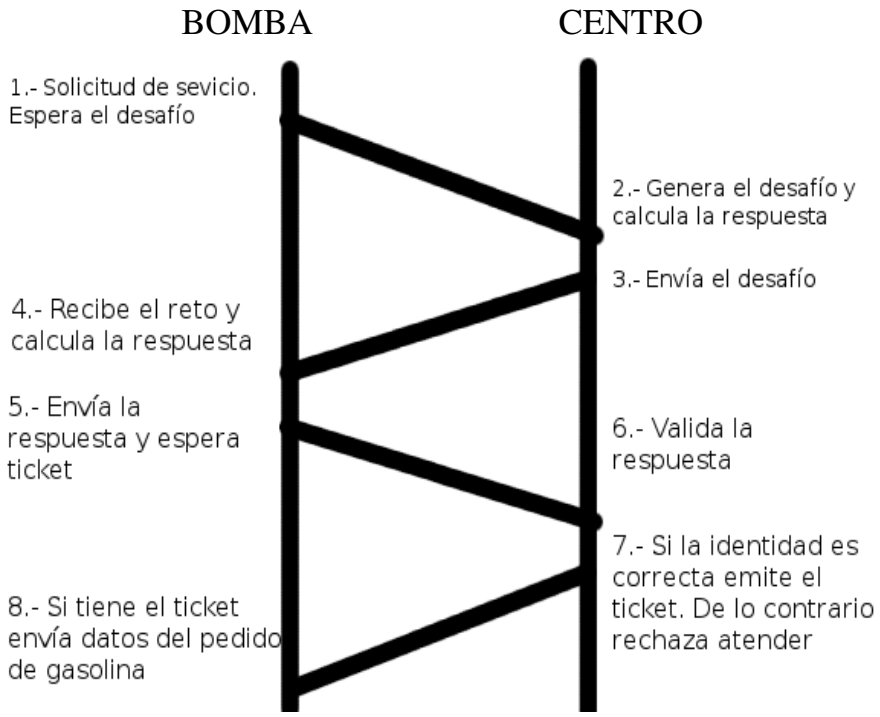
– Cualquier autenticación efectuada, correcta o fallida deberá ser añadida como un evento más a registrar en los archivos bitácoras.

– Los tickets de servicio pueden ser tan simples como usted lo decida. Un posible formato de tupla que los defina es:

(nro-del ticket, ip-servidor, fecha y hora en que expira la vigencia (cronómetro del servidor)

– Si desea sustituir el código que se le suministra o programar el suyo propio, no hay impedimento alguno, siempre que cumpla con el estándar del algoritmo MD5 (RFC 1321).

Un diagrama simple de flujo de acciones en el tiempo que ilustra la secuencia de operaciones de la autenticación, al inicio de la segunda etapa, es:



- Para aquellas funciones remotas que no formen parte de la etapa de autenticación, el ticket de servicio deberá ser suministrado como otro dato más. Si este resulta válido, se proveerá el servicio de la llamada. En caso contrario, no se procesará la misma.

Evaluación

- Este proyecto corresponde a 12% de la nota del curso. El proyecto que se entregue debe funcionar correctamente. Si no corre no será evaluado.
- Las especificaciones de la entrega serán publicadas oportunamente.
- Para su evaluación el proyecto será probado con un conjunto de datos que serán establecidos en el momento de la corrección, no necesariamente los que fueron suministrados como ejemplo.

- Otros detalles de la entrega, así como archivos de prueba, serán publicados oportunamente.
- El programa debe correr en las máquinas del LDC. Específicamente en la sala 221 (Debian 6 ver 2.6.32). Todos los proyectos deben funcionar en las computadoras de esa sala. Disponen de más de 20 equipos, suficientes para que todos los estudiantes donde pueden acceder de manera presencial o por "ssh".

Observaciones

- La implementación se hará con RPC y el algoritmo MD5.
- Los formatos, nombres de los comandos, etc., deberán ser estrictamente los especificados aquí.
- Por simplicidad de los proyectos se pide que no se implemente ningún tipo de interfaz gráfica, que se emplee la línea de comandos para invocar a los programas y la salida estándar para cualquier información que se desee reportar.
- El no seguir las instrucciones puede acarrear penalizaciones.
- Para efectos de desarrollo puede utilizarse una máquina para correr todos los componentes (cliente y varios servidores). Una vez funcione en una máquina, debe probarlo en al menos dos máquina (una con un cliente y un proveedor, y un proveedor en otra máquina) para verificar que funcione bien. **La corrección se hará usando varias máquinas.**

GDRC
Febrero 2013