Universidad de los Andes
Ingeniería de Sistemas y Computación
ISIS 2203 Infraestructura Computacional

Caso de estudio # 3
Seguridad

Objetivos

- Comprender el alcance y limitaciones en la aplicación de códigos criptográficos de hash para el manejo de confidencialidad e integridad de datos.
- Construir un prototipo a escala de una herramienta que evalúa el esquema para manejo de confidencialidad de las contraseñas en Sistemas Operativos Linux y Windows.

Problemática:

Linux y Windows (y otros sistemas) implementan varios mecanismos de protección para garantizar la confidencialidad de las contraseñas de usuarios, uno de estos mecanismos consiste en almacenar el código criptográfico de hash de las contraseñas en vez del texto plano. Esto permite garantizar que ni siquiera el administrador del sistema tendrá acceso directo a la información de contraseñas de usuarios.

Por otro lado, los programas conocidos como Password Crackers intentan encontrar la contraseña de un usuario a partir del código criptográfico de hash de dicha contraseña. Por las propiedades de los algoritmos de generación de códigos criptográficos de hash, el proceso de búsqueda debe usar una aproximación de ensayo y error, intentando diferentes valores hasta encontrar uno que genera la contraseña buscada.

Su tarea consiste en evaluar el nivel de protección ofrecido por este mecanismo de protección de contraseñas. Para ello, adelante las actividades que se describen a continuación.

A. [85%] Implementación del Prototipo.

Escriba un programa en Java que:

- Recibe como datos de entrada: (i) un algoritmo (H) de generación de código criptográfico de hash, (ii) una cadena (C) que representa el código criptográfico de hash de una contraseña, (iii) una secuencia de 2 caracteres (S) que representa la sal. Adicionalmente, el programa recibe el número de threads que implementará: los valores que usaremos son 1 y 2.
- Explora un espacio de búsqueda, buscando un valor V tal que el código criptográfico de hash de V concatenado con S: H(V concat S), corresponda al código C. Si el número de threads es 2, entonces el programa debe distribuir el espacio de búsqueda entre los dos threads.

Su programa debe cumplir con las siguientes condiciones:

- No puede usar las instrucciones break, ni continue. Vamos a contribuir al debate break, continue, goto vs. programación estructurada.
- No use librerías especiales, solo las librerías estándar de java (java.security). Si tiene alguna duda sobre una librería específica consulte con los profesores.
- El programa debe reportar la cadena V y la sal S que permiten generar el código C y el tiempo que tomó la búsqueda.

Para simplificar un poco el problema:

- Los algoritmos para generación de código criptográfico de hash que usaremos son SHA256 y SHA512.
- El espacio de búsqueda para el valor V debe considerar entradas con una longitud entre 1 y 7 caracteres (longitud máxima de 7 caracteres) y solamente consideraremos letras minúsculas (a-z). Tenga en cuenta que, en el caso de contraseñas reales se recomienda una longitud mayor y el uso de mayúsculas, números y caracteres especiales.

Adicionalmente, resuelva las siguientes tareas:

1. Corra su programa para dos códigos (uno de los cuales debe corresponder al peor caso en su espacio de búsqueda) en diferentes escenarios construidos a partir de las variaciones en los datos de entrada: para cada algoritmo, sobre cadenas con longitud 1, 2, 3, 4, 5, 6 y 7, con 1 y 2 threads. En cada caso, use dos sales diferentes (Recuerde que la sal se recibe como parámetro de entrada). Construya una tabla con los tiempos recopilados.
2. Construya dos gráficas:
    - Use los datos del punto anterior para construir una gráfica que muestre los tiempos registrados para 1 thread: para los dos algoritmos, sobre cada longitud y las dos sales.
    - Use los datos del punto anterior para construir una gráfica que muestre los tiempos registrados para 2 threads: para los dos algoritmos, sobre cada longitud y las dos sales.
3. Identifique la velocidad de su procesador, y estime cuántos ciclos de procesador tomaría, en promedio, generar y evaluar un valor para determinar si genera o no genera el código buscado. Escriba todos sus cálculos.
4. Con base en los cálculos del punto anterior, calcule cuánto tiempo tomaría un programa monothread, en promedio, para encontrar una contraseña en los siguientes casos:
    - contraseñas de 8 caracteres, cada carácter puede ser mayúscula, minúscula, número o uno de los siguientes carácteres especiales:.:,;!?(%)\+-/*{}, la sal es una secuencia de 16 bits
    - contraseñas de 10 caracteres, cada carácter puede ser mayúscula, minúscula, número o uno de los siguientes carácteres especiales:.:,;!?(%)\+-/*{}, la sal es una secuencia de 16 bits
    - contraseñas de 12 caracteres, cada carácter puede ser mayúscula, minúscula, número o uno de los siguientes carácteres especiales:.:,;!?(%)\+-/*{}, la sal es una secuencia de 16 bits

B. [15%] Análisis y Entendimiento del Problema.

Responda las siguientes preguntas.

Nota: No copie contenido de otros autores, construya sus propias respuestas con base en las referencias consultadas (e incluya dichas referencias en el informe).

1. Busque información adicional sobre los algoritmos de generación de códigos criptográficos de hash: (i) cuáles se usan hoy día y en qué contexto y (ii) por qué dejamos de usar aquellos que se consideran obsoletos.
2. La tecnología bitcoin usa un procedimiento conocido como "mining" que también depende de la existencia de algoritmos criptográficos de hash que no tienen funciones inversas. Describa en qué consiste el proceso de mining.
3. La adición de la sal se relaciona con el problema asociado al uso de Rainbow Tables. (i) Describa cuál es el problema de seguridad asociado, (ii) Describa cómo ayuda la sal a resolver o mitigar ese problema.

Entrega:

- Cada grupo debe entregar un zip de un proyecto Java con: La implementación del prototipo. En el subdirectorio docs debe haber un archivo que incluya el informe con las respuestas a los puntos A.1 a A.4 y B.1 y B.2. **Al comienzo del informe, deben estar los nombres y carnés de los integrantes del grupo**. Si un integrante no aparece en el documento entregado, el grupo podrá informarlo posteriormente. Sin embargo, habrá una penalización: la calificación asignada será distribuida (dividida de forma equitativa) entre los integrantes del grupo.
- Recuerde incluir en su informe **todas las referencias** que use para resolver este proyecto.
- El trabajo se realiza en los grupos asignados. No debe haber consultas entre grupos.
- El grupo responde solidariamente por el contenido de todo el trabajo, y lo elabora conjuntamente (no es trabajo en grupo repartirse puntos o trabajos diferentes). Se puede solicitar una sustentación a cualquier miembro del grupo sobre cualquier parte del trabajo. Dicha sustentación será parte de la calificación de todos los miembros.
- El proyecto debe ser entregado por Bloqueneon por uno solo de los integrantes del grupo.
- **La fecha límite de entrega es mayo 1 de 2023 a las 23:50 p.m.**

Referencias:

- *Cryptography and network security*, W. Stallings, Ed. Prentice Hall, 2003
- *Computer Networks*. Andrew S. Tanenbaum. Cuarta edición. Prentice Hall 2003, Caps 7, 8.
- *To use or not to use the goto statement*: Programming styles viewed from Hoare Logic. Hidetaka Kondoh, Kokichi Futatsugi. Science of Computer Programming. Volume 60, issue 1. 2006
- *Bitcoin: A Peer-to-Peer Electronic Cash System*. Satoshi Nakamoto. 2009

Cronograma Propuesto:

17 de abril: Publicación del enunciado
17 a 18 de abril: Leer el enunciado completo y comunicarse con los compañeros de grupo
19 a 21 de abril: Implementar y probar el prototipo (con longitudes cortas para validar su funcionamiento)
22 a 23 de abril: Correr el prototipo para todos los escenarios requeridos con longitudes 1 a 6 y recopilar los datos
24 a 26 de abril: Correr el prototipo para todos los escenarios requeridos con longitud 7 y recopilar los datos
27 de abril a 1 de mayo: Informe y entrega