Conceptos Básicos y Algoritmos Fundamentales

**Sistema de ordenamiento y búsqueda de registros de login fallidos**

Juan Carlos Garfias Tovar

A01652138

David Alonso Cantú Delgado

11 de septiembre del 2020

**Introducción**

En la actualidad debido a la cantidad de servicios e información se han creado distintos métodos de ataque con el fin de obtener la información que cada vez es mas valiosa. Los botnets usan ataques cibernéticos como keyloggers, troyanos, entre otros. Los Botnet son capaces de ejecutarse de manera autónoma y automática permitiendo asi tomar control de los ordenadores/servidores de forma remota. Este tipo de sistemas comenzaron a popularizarse a partir del ano 2000 cuando un adolescente en Canadá llamado Mafiaboy ataco al sobrecargar paginas como Yahoo!, Etrade, EBay, Amazon, entre otros. Un botnet puede ser propiamente definido como una red de ordenadores infectados con variedad de malware (Kaspersky, 2020). Este tipo de sistemas tienden a realizar ataques DDoS causando inestabilidad y violando la integridad de los sistemas.

Los crackers usan distintos métodos de ataque tal como el drive by downloads y ataques por email. Explotando vulnerabilidades que permitan cargar código malicioso en la pagina permitiendo redirigir al usuario a un sitio clonado donde el delincuente puede descargar e instalar archivos al equipo. En el segundo caso se utilizan ataques de archivos con contenido malicioso donde se aloja el código. Tras infectar el equipo el atacante puede manejar comandos, cargar archivos, cediéndole asi control al equipo.

Uno de los ataques mas populares de los crackers es el ataque de diccionario. Este tipo de ataques intentan averiguar contraseñas probando todas las palabras de un diccionario. Los diccionarios son conjuntos de frases, datos o palabras que son relevantes para la víctima, el cracker busca todas las combinaciones posibles con el fin de encontrar la posible contraseña. Con este tipo de ataques se ha logrado comprobar que un gran numero de usuarios eligen las mismas contraseñas, las cuales se vuelven fáciles de encontrar para los delincuentes (BBVA, 2020). Este método tiene un alto grado de efectividad si el sistema atacado no cuenta con medidas como captcha u otros sistemas para controlar el alto volumen de solicitudes de login de una misma cuenta.

El sistema propuesto busca añadir una capa de seguridad al poder realizar búsquedas para logins fallidos en un sistema general. Permitiendo asi identificar casos atípicos, los cuales pudieran representar un intento de delincuentes. Por ello, el sistema es capaz de tomar un registro y realizar búsquedas por fecha, permitiendo asi observar la frecuencia de los intentos de login y a su vez observar las IPs de los atacantes. Para asi poder bloquear la cuenta e informar al usuario que su cuenta se ha visto comprometida.

**Importancia de la eficiencia algorítmica**

Con el fin resolver un problema a través de una computadora se pueden implementar diferentes algoritmos. Sin embargo, las distintas implementaciones pueden tener diferencias en cuanto a tiempos de ejecución y a memoria por lo que contar con un sistema que permita delimitar que algoritmo es mas eficiente es fundamental. A partir de esta situación se crea el sistema Big O notation, el cual permite describir el rendimiento o complejidad de un algoritmo. Esta notación se encarga de describir el peor escenario y a su vez permite describir el tiempo de ejecución requerido o de espacio en memoria. Existen diversos tipos de eficiencia en algoritmo, de los cuales los más comunes son O 1), O(n), O(n^2), O(2^n) y O(log n). A continuación, se muestra una imagen que describe de manera grafica el comportamiento de cada uno de los principales casos.

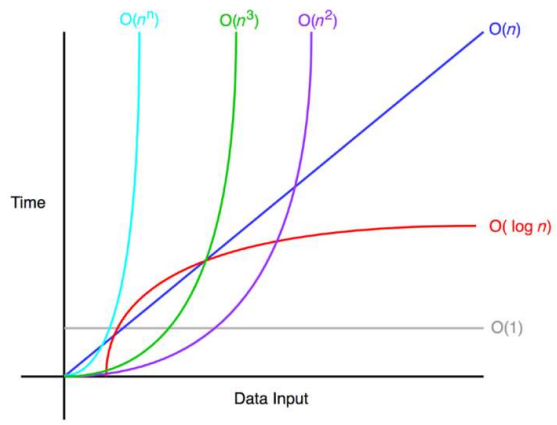


**Imagen 1.0:** Grafico de complejidad

El problema de búsqueda de intentos de login en una base de datos no ordenada implica ciertas problemáticas, las cuales deben de ser solucionadas de la manera mas efectiva posible. Existen diversos algoritmos que resultan útiles para esta situación, de los cuales destacan los de ordenamiento y búsqueda. Debido a que una base de datos tiende a tener cientos de miles de entradas es necesario considerar únicamente a aquellos algoritmos que tengan un grado de eficiencia alto. De ellos destacan los algoritmos de merge sort (O (n*Log n)) y quick sort (O (n*Log n)) para los casos de ordenamiento. Mientras que para la búsqueda destaca el algoritmo binary search (O ($\log_2 n$)).

Estos algoritmos son fundamentales para la resolución del problema debido a que es factible que los ataques sean de diferentes IPs pero tiendan a estar en horas similares por lo que ordenarlos es vital para encontrar patrones de ataque. A su vez, al ser un registro tan amplio se requiere de un sistema de búsqueda eficiente y sumamente efectivo.

**Registros de logins fallidos**

El sistema de log de registros de login fallidos esta dado como un txt, el cual cuenta con mas de dieciséis mil registros desordenados. Con el fin de realizar una búsqueda eficiente fue necesario realizar un algoritmo de ordenamiento. Se planteo la utilizar merge sort y quick sort debido a su eficiencia algorítmica. Sin embargo, con el fin de seleccionar únicamente uno se realizó una comparación entre ellos. Ambos algoritmos son de tipo “Divide and conquer” sin embargo tienen grandes diferencias a partir del tamaño del conjunto de datos donde se realizará la búsqueda.

Quick sort divide al vector en ratios, teniendo en su peor caso un O(n^2), por lo que únicamente funciona rápidamente en vectores pequeños con su algoritmo de ordenamiento interno. Por otro lado, el algoritmo merge sort divide al vector en ratios, pero debido a la naturaleza del mismo es capaz de ser consistente en cuanto a velocidad en cualquier tamaño de vector, sin embargo, utiliza más memoria su ordenamiento externo. Sin embargo, debido a que la intención del sistema es realizar búsquedas rápidas se decidió utilizar merge sort para los registros. En cuanto a la búsqueda se realizo un algoritmo de binary search modificado pasando a ser recursivo. Esto debido a que si la query ingresada contiene como limite de fechas un valor que no se encuentra en el registro, el sistema es capaz de poder buscar el elemento siguiente o el anterior. Es decir que, si un usuario ingresa la fecha de febrero 3 y como fin febrero 20, pero los datos van de febrero 4 a febrero 19, el sistema es capaz de resolverlo. Esto al no encontrar los datos en el vector, resta o suma un día, mes u hora. Para poder realizar las búsquedas y ordenamiento se implemento un struct con los datos de un registro y sus métodos para evaluar en las funciones de búsqueda y ordenamiento.

```
struct RegistryEntry{
    string month;
    int day;
    int hour;
    int minute;
    int second;
    string ip;
    string error;

    //methods
    bool dateIsEqual(RegistryEntry const & entry) const{
    }
    bool dateIsMinor(RegistryEntry const & entry) const{
    }
    bool hourIsMinor(RegistryEntry const & entry) const{
    }
    bool hourIsMajor(RegistryEntry const & entry) const{
    }
    bool hourIsEqual(RegistryEntry const & entry) const{
    }
}
```

**Imagen 2.0:** Estructura de datos para el registro

Este tipo de problemas en situaciones reales son solucionados con métodos ya implementados en sistemas como MySQL, entre otros tipos de sistemas de bases de datos. Principalmente se utiliza en elementos como indexado de entradas, donde la base de datos recibe una entrada y la coloca de manera ordenada. De igual manera en sistemas como phpMyAdmin para poder realizar búsquedas, esto haciéndolo a partir de los keys dependiendo del tipo de query. Por otro lado, distintos lenguajes de programación utilizan implementaciones nativas de este algoritmo para hacer búsquedas tal como data.table en R o dataframes en Python. Las cuales son sumamente eficientes debido a que lo hacen en C. Otro tipo de implementación es para hacer debugging en Git con git busect. Finalmente, otro tipo de implementación es en juegos 3D los cuales utilizan el espacio dividido en una estructura de árboles y la búsqueda binaria es usada para encontrar subdivisiones al display a partir de la posición 3D y la cámara. Binary search es un algoritmo que se utiliza para acceder a datos de manera rápida cuando la memoria es poca, permitiendo asi implementaciones de todo tipo para encontrar soluciones, elementos, entre otros.

**Conclusión**

Tras realizar esta implementación de ordenamiento y búsqueda me pude dar cuenta de la importancia de conocer a mayor profundidad los algoritmos fundamentales. Esto para a partir de ellos poder realizar implementaciones eficientes y útiles. En este caso las adaptaciones de ambos algoritmos que fueron creados para ints pasados a una adaptación para struct demostraron la importancia de conocer el como funcionan cada uno de ellos. Gracias a esto pude realizar métodos internos que evaluaran aquellos elementos necesarios para ser implementados en las búsquedas y ordenamientos. Considero que a pesar de que existan otro tiempo de implementaciones mas eficientes como arboles para este tipo de problemas, realizar ejercicios, los cuales puedan presentar escenarios reales es importante para desarrollar inteligencia algorítmica, creando asi un habito de buscar la solución mas eficiente y no aquella que sea la más rápida o simple de hacer. El trabajar con gran cantidad de datos me permitió ver la importancia de escalar los problemas, pensado en los peores escenarios, excepciones y buscando hacer del algoritmo mas efectivos posibles. Al igual que sea una implementación que a futuro se pueda modificar para añadir funcionalidades tales como detección de accesos maliciosos, IPs malignas, entre otras.

Finalmente considero que un sistema de este tipo tendría que pensarse más allá de las fechas ya que en casos de años serian millones de entradas por lo que la implementación del código realizado únicamente requeriría de una pequeña modificación gracias a la implementación generalizad. Considero que esta implementación es eficiente y permite que los administradores puedan realizar las queries con éxito. Los métodos pueden acoplarse a otro tipo de implementaciones sin necesidad de hacer refactoring debido a su amplia escalabilidad. Abriendo asi posibilidades para obtener más información sobre los ataques, detección de login, aumentar los niveles de seguridad y encontrar vulnerabilidades en contraseñas.

**Referencias**


¿Qué es un ataque de diccionario (si no se refiere a arrojar un libro a alguien)?. (2020). Retrieved 8 September 2020, from https://www.bbva.com/es/que-es-un-ataque-de-diccionario-si-no-se-refiere-a-arrojar-un-libro-a-alguien/

¿Qué es un botnet? – Kaspersky Daily. (2020). Retrieved 8 September 2020, from https://www.kaspersky.es/blog/que-es-un-botnet/755/

A beginner's guide to Big O notation - Rob Bell. (2020). Retrieved 8 September 2020, from https://rob-bell.net/2009/06/a-beginners-guide-to-big-o-notation/#:~:text=Big%20O%20notation%20is%20used,on%20disk)%20by%20an%20algorithm.

Glosario. (2020). Retrieved 8 September 2020, from https://www.ccn-cert.cni.es/publico/seriesCCN-STIC/series/400-Guias_Generales/401-glosario_abreviaturas/index.html?n=90.html

practice? W., Panther, G., Huima, A., & Hawes, A. (2020). Where is binary search used in practice?. Retrieved 8 September 2020, from https://stackoverflow.com/questions/540165/where-is-binary-search-used-in-practice

Quick Sort vs Merge Sort - GeeksforGeeks. (2020). Retrieved 8 September 2020, from https://www.geeksforgeeks.org/quick-sort-vs-merge-sort/

Tiempo de ejecución y eficiencia de algoritmos — Laboratorio de Matemáticas 2010/2011. (2020). Retrieved 8 September 2020, from http://matematicas.uam.es/~pablo.angulo/doc/laboratorio/b2s2.html