Análisis y diseño de algoritmos avanzados (Gpo 602)

AI1 Actividad Integradora 1

Mauricio Garcia Villanueva - A01704098
Azul Rosales - A01706348
Jose Emiliano Riosmena Castañon - A01704245

Propuesta de Solución

1 de octubre de 2023

Reporte de Proyecto

Introducción

Hoy en día la transmisión de datos e información de un dispositivo a otro es de lo más normal. Debido a esto existen personas mal intencionadas que pueden interceptar estas transmisiones con el objetivo de modificarlas y obtener cierto control del dispositivo que recibe la información enviada. Debido a esto es esencial contar con herramientas y técnicas que nos permitan identificar la presencia de códigos maliciosos dentro de las transmisiones de datos.

El objetivo principal de esta situación problema es desarrollar un programa que pueda analizar archivos de transmisión en busca de código malicioso, representado como secuencias de caracteres y determinar si dichas secuencias están presentes en los archivos de transmisión.

Problemática

La situación está dividida en tres partes. En la primera parte se realiza la búsqueda de código malicioso en los archivos de transmisión. Se debe informar si hay presencia de código malicioso y en el caso de encontrarse también se debe informar la posición en la que comienza el código en el archivo de transmisión.

En la segunda parte, nos encargamos de la detección de código “espejado” (palíndromos) dentro de los archivos de transmisión. Para esta etapa se asume que el código malicioso tiene la característica de ser un palindroma a nivel de caracteres, lo que significa que puede leerse igual en ambas direcciones. En esta parte se muestra la posición inicial y final en la que se encuentre el código espejada más largo en cada archivo de transmisión.

Por último la tercera parte, esta se centra en analizar la similitud entre los archivos de transmisión. Para este caso el programa devuelve la posición inicial y final del primer substring común más largo entre los dos archivos de transmisión.

Propuestas de Solución

KMP (Knuth-Morris-Pratt) - Parte 1

Para abordar la Parte 1 del proyecto, la cual como ya se ha mencionado está enfocada en la detección de código malicioso en los archivos de transmisión y la posición de estos, se hizo uso del algoritmo de KMP.

El algoritmo de Knuth-Morris-Pratt (KMP) es una técnica de búsqueda de patrones en cadenas de texto, el cual nos permite encontrar las ocurrencias de una subcadena dentro de la cadena principal de una manera eficiente. Esto se puede lograr por medio de una tabla de preprocesamiento la cual nos ayuda a evitar las repeticiones innecesarias en la búsqueda (las cuales suceden cuando se usa fuerza bruta).

A medida que se examina el texto buscando una coincidencia con el patron, si encuntrauna discrepancia entre los caracteres el algoritmo no vuelve a verificar los caracteres que ya ha evaluado, en lugar de esto, hace uso de la información que ha sido previamente calculada par determinar dónde continuar con la búsqueda(Para esto se hace uso de la tabla de preprocesamiento (LPS)). Esto hace que la búsqueda sea mucho más eficiente.

Implementación:

En el archivo "BusquedaKMP.h", se creó una función llamada construirLPS, esta función se encarga de la construcción de la tabla LPS (Longest Prefix Suffix). Después tenemos la función algotirmoKMP**,** la cual implementa el algoritmo KMP para buscar el código malicioso dentro de los archivos de transmisión. Por último tenemos la función lectorArchivos, desde esta función se mandan a llamar a la función algoritmoKMP, la función simplemente se encarga de leer los archivos, asignas los datos a variables y coordinar la búsqueda de los códigos maliciosos dentro de los archivos de transmisión.

Criterios de la selección del algoritmo:

Se eligió el algoritmo KMP, ya que es el que mejor se adapta a lo pedido en esta parte, pues este algoritmo está diseñado para encontrar patrones dentro de textos. De igual forma existen otros posibles algoritmos pero se tomó en cuenta como un factor muy importante el tiempo de complejidad. Por ejemplo si hubiéramos usado fuerza bruta el tiempo de complejidad sería de O(m*n), en donde m es la longitud del código malicioso y n es la longitud del texto de transmisión. Sin embargo al hacer uso del algoritmo KMP, el tiempo de complejidad se reduce a O(m+n), pues estamos evitando

tener que regresar al inicio en la búsqueda de patrones. Como ya se explicó previamente. De igual forma un criterio fue la precisión**,** este algoritmo nos garantiza la detección precisa de las secuencias de caracteres de la manera más eficiente.

Palíndromos - Parte 2

Como se mencionó anteriormente el código malicioso puede tener la característica de ser palíndromo. Lo cual significa que es espejo, es decir, se lee igual que como lo hacemos al revés. Para ello hemos generado un algoritmo que compara desde el inicio y el final de la cadena, y va analizando caracter por caracter para determinar si es un palíndromo, y si en el proceso encuentra un carácter que no coincide ya sea con el del inicio o el final, entonces determina que no es un palíndromo.

Implementación

En el archivo de “Palindromo.h” implementamos la función esPalindromo, la cual es una función booleana, en donde recibe un string. Después definimos el inicio en 0, y el fin con la longitud de la cadena menos uno. Y hacemos un bucle hasta que el inicio deje de ser menor que el fin. Entonces en ese ciclo, comparamos si el valor actual del inicio y fin son iguales, si no lo son entonces regresa falso, indicando que el string no es palíndromo. Si el bucle termina, entonces regresa cierto, indicando que el string sí es palíndromo.

Criterios de selección del algoritmo

Mayormente, elegimos este algoritmo ya que es muy sencillo y como solamente nos basamos de un bucle while, entonces tenemos que la complejidad del algoritmo es de O(n). Este algoritmo es bastante preciso y muy sencillo, porque simplemente va comparando caracter por caracter de inicio a fin y de fin a inicio, y compara si los caracteres actuales son iguales.

Substring común más largo - Parte 3

En el contexto de la detección de actividad maliciosa o compromiso de datos, el substring común suele ser la parte de los datos que se mantiene sin cambios o que es idéntica en ambos conjuntos de datos originales y recibidos. La parte maliciosa o comprometida sería cualquier diferencia o modificación que haya ocurrido en los datos y que no esté presente en el substring común. Por lo tanto, el substring común puede servir como referencia para identificar discrepancias o alteraciones en los datos durante la transmisión.

Implementación

En el archivo “StringComunMasLargo.h” implementamos el algoritmo de *Longest Common Substring* (LCS). El propósito de este algoritmo es encontrar el substring más largo que tienen en común dos strings, así como identificar sus posiciones iniciales y finales en ambos strings. El algoritmo busca el substring más largo que es idéntico en ambas cadenas de texto y utiliza una matriz para rastrear las longitudes de los substrings comunes a medida que compara los caracteres entre los archivos. Utilizamos la programación dinámica para optimizar la búsqueda del substring común más largo al dividir el problema en subproblemas más pequeños, evitando el cálculo repetitivo..

Criterios de selección del algoritmo

Es un algoritmo ampliamente utilizado y es fácil de implementar y entender, por su simplicidad conceptual y versatilidad. A pesar de tener una complejidad cuadrática en el peor de los casos (O(m * n)), el LCS es eficiente en la mayoría de las situaciones prácticas.

Conclusiones

En el desarrollo de este proyecto se implementaron tres soluciones distintas para abordar cada uno de los diferentes puntos. Cada parte del proyecto tuvo un enfoque diferente y fue de mucha ayuda para la detección y comprensión de amenazas en la transmisión de datos.

La capacidad de identificar transmisiones de datos comprometidas y potencialmente maliciosas es esencial para garantizar la seguridad y la confianza en el mundo digital. Los algoritmos desempeñan un papel esencial en el desarrollo de mecanismos eficaces para esta tarea, permitiendo tener una detección temprana y una respuesta rápida. Pero esta respuesta rápida depende de un diseño eficiente de los algoritmos, por ello, un buen diseño de algoritmos es esencial para analizar grandes volúmenes de datos en tiempo real y tomar decisiones informadas de manera ágil.

En la primera parte, se empleó el algoritmo Knuth-Morris-Pratt (KMP) para detectar patrones de código malicioso en archivos de transmisión, optimizando la búsqueda y mejorando la precisión en la identificación de secuencias maliciosas. En la segunda parte, se desarrolló un algoritmo simple pero eficaz para identificar palíndromos en el código malicioso. Esta capacidad es fundamental, ya que algunos códigos maliciosos utilizan palíndromos para ocultar su verdadera intención. La implementación directa y lineal proporcionó una herramienta efectiva para identificar estos patrones especiales. Por último, en la tercera parte, se implementó el algoritmo Longest Common Substring (LCS) para

encontrar las subcadenas comunes más largas entre dos archivos de transmisión. Esto fue de ayuda para comparar versiones de archivos y detectar incluso alteraciones mínimas en los datos transmitidos. La precisión y capacidad del LCS para identificar discrepancias nos dieron una base sólida para detectar cambios en la transmisión de datos.

En conclusión, este proyecto ha subrayado la importancia de la combinación adecuada de algoritmos, diseño eficiente y comprensión de las distintas amenazas para garantizar la integridad y seguridad de los datos en un mundo digital. Estas soluciones proporcionan una sólida base para la detección y prevención de amenazas o códigos maliciosos.

Referencias

1. Back To Back SWE. (2019, January 5). *Knuth–Morris–Pratt (KMP) Pattern Matching Substring Search - First Occurrence Of Substring* [Video]. YouTube. https://www.youtube.com/watch?v=BXCEFAzhxGY
2. Mathur, T. (2021, November 15). *KMP Algorithm | Knuth Morris Pratt Algorithm* - Scaler Topics. *Scaler Topics*. https://www.scaler.com/topics/data-structures/kmp-algorithm/
3. HKUST. (2016, October 28). *Longest Common Subsequences and Substrings*. Department of Computer Science. https://cse.hkust.edu.hk/mjg_lib/Classes/COMP3711H_Fall16/lectures/LCS_Handout.pdf