UNIVERSIDAD DE GUADALAJARA
CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

Análisis de algoritmos

Mtro. Jorge Ernesto Lopez Arce Delgado

Reporte Proyecto Final

Integrantes:
Braulio Hurtado Escoto 220426225
Jorge Daniel Hernández Reyes 220027797
Juan Pablo Solis Regin 220468416

Introducción

A lo largo de estas semanas, hemos estado construyendo nuestro código paso a paso, probando distintos tipos de algoritmos y formas de programar. Esto nos ha servido para ver de cerca cómo trabaja cada uno y cuáles son las mejores características que nos ofrecen. Hemos aprendido mucho sobre lo que hace que un código sea rápido y funcione bien.

Hoy, nuestro código final está listo. Se define usando solo las mejores técnicas de programación que aprendimos durante el semestre, es decir, aquellas que nos permiten ser más rápidos y más efectivos. Hemos elegido e implementado con cuidado las soluciones más eficientes con las que nos sentimos más cómodos para lograr nuestro objetivo: desarrollar un algoritmo de búsqueda de contraseñas por diccionario que sea muy superior.

Objetivo general

Por medio de este proyecto, se busca demostrar la implementación final y optimizada de nuestro algoritmo de búsqueda de contraseñas por diccionario. Esto se logra aplicando las técnicas de programación más eficientes aprendidas durante el semestre, con el fin de verificar la seguridad de un conjunto de contraseñas y, simultáneamente, medir y validar la eficiencia y el rendimiento superior del código desarrollado.

Objetivos particulares

- Implementar un método de búsqueda de contraseñas por diccionario
- Desarrollo de un modelo de fuerza bruta que realiza combinatorias con generación exhaustiva.
- Desarrollar una mejora en el algoritmo base para trabajar con diferentes contraseñas en paralelo utilizando la técnica de divide y vencerás
- Integrar un algoritmo de Huffman para comprimir y descomprimir diccionarios
- Implementar una interfaz gráfica de usuario con Tkinter que permite ingresar contraseñas objetivo

Desarrollo

Las contraseñas son una parte fundamental en la actualidad, pues se encargan de proteger la integridad de la información de las personas en el internet. Hoy en día, se requiere tener cuidado sobre la definición de una contraseña, pues existen muchos ciberatacantes que se encargan de encontrar cuentas con contraseñas débiles con el objetivo de obtener información de las personas. Esto de alguna manera puede poner en riesgo la integridad de la persona, desde su identidad hasta problemas que pueden poner en peligro su vida. Por ello buscamos que por medio de nuestro proyecto, sea posible comprobar que tan seguras son las contraseñas de las personas y que tan fácil es encontrarlas dentro de la web por medio de diccionarios de contraseñas filtradas.

Enfoque

La creación de este proyecto surge de una problemática crítica y actual: la brecha en ciberseguridad que enfrentamos en la vida digital cotidiana. A pesar de que las credenciales de acceso son nuestra primera y más importante línea de defensa, existe una marcada desinformación y desinterés generalizado en la enseñanza y aplicación de buenas prácticas de seguridad digital.

La Necesidad del Proyecto

En la actualidad, muchos usuarios continúan empleando contraseñas débiles o fácilmente predecibles, lo que las hace vulnerables a ataques automatizados, como los de diccionario. Este desconocimiento sobre la facilidad con que pueden ser descifradas en cuestión de segundos representa un riesgo significativo para la privacidad y la integridad de la información personal y profesional.

Contribución y Concientización

La creación de este proyecto tiene un objetivo fundamental que va más allá de la programación: aportar conciencia sobre el riesgo.

El algoritmo desarrollado no solo verificar técnicamente la eficiencia del código, sino que también sirve como una herramienta educativa para el usuario final. Al utilizar el programa, el usuario puede validar de forma práctica la vulnerabilidad de sus propias contraseñas frente a una base de datos preestablecida (el diccionario).

Implementación de las técnicas

El algoritmo fue desarrollado conforme avanzamos durante el semestre haciendo uso de distintas técnicas que fueron aplicadas en las distintas etapas del proyecto, desde el algoritmo base hasta el proyecto final.

Fuerza bruta (algoritmo base)

En este módulo, el algoritmo funciona más como un descifrador de contraseñas, pues realiza combinaciones de letras y números para intentar averiguar la contraseña ingresada por el usuario. Si bien es una etapa temprana del proyecto, como tal el problema de búsqueda de contraseña por diccionario no fue abordado directamente y solo se planteó el funcionamiento para posteriores etapas

Divide y vencerás

En la implementación de divide y vencerás, el algoritmo ahora acepta más de una contraseña y se comenzó a trabajar con hilos de manera que pudiéramos intentar abordar más de una contraseña a la vez haciéndolo paralelo en lugar de hacerlo secuencialmente. El trabajar con hilos nos permitirá posteriormente ahora sí abordar por completo el ataque por diccionario directamente con varias contraseñas a la vez.

Técnica voraz

Utilizamos el algoritmo de Huffman para la compresión de descompresión de diccionarios cargados desde archivos txt, estos diccionarios fueron obtenidos directamente de páginas de ciberseguridad como OWASP. Ya con los diccionarios, el algoritmo ahora utiliza los diccionarios para buscar dentro de ellos las contraseñas ingresadas por el usuario.

El algoritmo busca las contraseñas de manera paralela en distintos hilos, de modo que utilizan un diccionario ya cargado (el usuario puede elegir el diccionario que desea probar) para buscar las contraseñas dentro de él. Es posible utilizar todos los diccionarios a la vez de manera secuencial, sin embargo sería bastante tardado por lo que es preferible utilizar diccionarios uno por uno. Una vez que se deja de utilizar un diccionario, es comprimido de nuevo para ahorrar espacio de almacenamiento.

Roles y responsabilidades

Integrante	***Funciones Principales***	***Actividades específicas***	***Entregables/ resultados***
Braulio Hurtado Escoto	Implementación del Algoritmo de Búsqueda y Optimización	Desarrollo e implementación del algoritmo de búsqueda por diccionario. Aplicación de técnicas de optimización de rendimiento. *(20/Noviembre/25)*	Módulo funcional del algoritmo de búsqueda. Integración de librerías para el manejo de hilos/procesos. *(24/Noviembre/25)*
Jorge Daniel Hernández Reyes	Pruebas, Validación y Documentación	Creación de casos de prueba y conjuntos de datos. Ejecución de pruebas de rendimiento y validación de eficiencia. Redacción final de la documentación técnica. *(25/Noviembre/25)*	Reporte de pruebas de rendimiento. Código depurado y documento final del proyecto y código fuente. *(25/Noviembre/25)*
Juan Pablo Solis Regin	Diseño de Arquitectura y Estructura de Datos	Investigación y selección de la estructura de datos más eficiente para el diccionario. Definición de la arquitectura del código base. *(22/Noviembre/25)*	Documento de diseño de la arquitectura. Implementación de la estructura de datos del diccionario en el código fuente. *(23/Noviembre/25)*

SCRUM del proyecto

Prioridad	***User Story***	***Descripción***
Alta	Como usuario quiero ingresar contraseñas.	Para ingresar mis contraseñas a buscar.
Alta	Como usuario quiero poder ver las contraseñas que he ingresado.	Para visualizar las contraseñas ingresadas a buscar.
Alta	Como usuario quiero cargar diccionarios con contraseñas ya predeterminadas.	Para ampliar la búsqueda de contraseñas.
Media	Como usuario quiero cambiarme de modo a divide y vencerás.	Para complementar la búsqueda más técnica.
Baja	Como usuario quiero modificar el tiempo de delay de los intentos.	Para tener mayor control en el tiempo.
Alta	Como usuario quiero iniciar la búsqueda	Para comenzar la búsqueda.
Alta	Como usuario quiero detener la búsqueda	Para terminar la búsqueda.
Alta	Como usuario quiero ver los resultados de las contraseñas encontradas.	Para saber las contraseñas que ha encontrado.

Complejidad temporal de cada algoritmo

Técnica	**Algoritmo Base**	**Complejidad T emporal (Peor Caso)**	**Tipo de Complejidad**	**Variables Clave**
Fuerza Bruta	Iteración Exhaustiva	$O(\|\Sigma\|L)$	Σ	L
Técnica Voraz	Búsqueda Secuencial en Diccionario	$O(D\ log\ D)$	Lineal	D = Tamaño del diccionario (número de contraseñas a probar).
Divide y Vencerás	Búsqueda Exhaustiva Paralelizada	$(\|\Sigma\|L)$	Σ	L = Trabajo total

Comparación de complejidad temporal entre las distintas tecnicas

Fuerza bruta:

La complejidad es exponencial porque el algoritmo debe probar, en el peor caso, cada posible combinación de caracteres. El número de intentos crece exponencialmente a medida que aumenta la longitud máxima (L).

Técnica voraz:

- **Búsqueda (Ataque)**: El peor caso para un ataque de diccionario es cuando la contraseña es la última en la lista. Si el diccionario tiene D palabras, la complejidad es **lineal** O(D).
- **Preparación (Huffman)**: La construcción de la codificación de Huffman (si se usa para priorizar) es O(D log D), pero el tiempo de búsqueda domina si el diccionario ya está listo.

El ataque de diccionario es significativamente más rápido que la Fuerza Bruta *si y sólo si* la contraseña se encuentra en el diccionario.

Divide y vencerás:

El espacio de búsqueda se divide en particiones que son atacadas simultáneamente por múltiples hilos o procesos. El trabajo *total* que se realiza es el mismo que en la Fuerza Bruta y el tiempo que le toma a la máquina encontrar la contraseña se reduce considerablemente.

El funcionamiento es muy simple para el usuario:

1. El Usuario Ingresa las Contraseñas

El programa le pedirá al usuario que escriba las contraseñas que quiere que revisemos para saber qué tan fáciles son de adivinar.

2. El Programa Se Pone a Buscar

El algoritmo empieza a trabajar rápidamente. No solo busca las contraseñas en los diccionarios que ya tenemos, sino que también crea combinaciones propias con esas palabras para ser más rápido al momento de descifrar. Esto nos ayuda a terminar la búsqueda en el menor tiempo posible.

3. Resultados en Pantalla

Mientras trabaja, el programa muestra en pantalla lo que está haciendo. El usuario verá el proceso de búsqueda y sabrá si fue exitoso o no:

- Si encuentra la contraseña: Muestra inmediatamente la contraseña descifrada y el tiempo exacto que tardó en adivinarla. Esto prueba que la contraseña es débil.
- Si no la encuentra: El programa terminará de ejecutarse y le indicará al usuario que la contraseña resistió todos los intentos de nuestro diccionario, confirmando que es una contraseña segura contra este tipo de ataque.

De esta forma, el usuario puede ver si sus contraseñas son seguras o si necesitan cambiarlas urgentemente.

Conclusión

Por medio de este proyecto que se fue desarrollando a lo largo de estas semanas, se llevaron a cabo los estudios, análisis e implementaciones de los algoritmos vistos en clase, logrando implementarlos de manera eficiente en nuestro código, aprendiendo sobre el uso de estos algoritmos para un problema que es las vulnerabilidades a las que nos exponemos todos en la red.
Nuestro código no pretende dañar a usuarios o usarlo para fines maliciosos, es una demostración y un análisis sobre lo que podemos hacer nosotros para protegernos en la red a partir desde las contraseñas que definimos y utilizamos para guardar toda nuestra información digital delicada que puede exponernos a peligros y amenazas reales si no se sabe proteger de manera correcta nuestras contraseñas.
Nuestro código puede servir también como guía de estudio para ayudar a las demás personas a comprender lo que significa el crear contraseñas seguras y robustas en nuestras distintas cuentas digitales que podamos llegar a tener.

Fuentes de consulta:

- OWASP. (n.d.). *Brute Force Attack Software Attack | OWASP Foundation*. Owasp.org. https://owasp.org/www-community/attacks/Brute_force_attack
- Ranjan, R. (n.d.). *Password Spraying Attack | OWASP Foundation*. Owasp.org. https://owasp.org/www-community/attacks/Password_Spraying_Attack
- OWASP. (2017). *A2:2017-Broken Authentication | OWASP*. Owasp.org. https://owasp.org/www-project-top-ten/2017/A2_2017-Broken_Authentication
- david-palma. (2025). *wordlists/password-dictionaries at main · david-palma/wordlists*. GitHub. https://github.com/david-palma/wordlists/tree/main/password-dictionaries
- *SecLists/Passwords at master · danielmiessler/SecLists*. (n.d.). GitHub. https://github.com/danielmiessler/SecLists/tree/master/Passwords
- PYTHON. (2025). *Python*. Python.org; Python.org. https://www.python.org/