The background is a vibrant, abstract digital composition. It features a gradient of colors ranging from deep blue and purple on the left to bright yellow and green on the right. Overlaid on this are numerous thin, wavy lines that create a sense of motion and depth. Scattered throughout the image are various binary digits (0s and 1s) in different sizes and colors, some appearing to float or fall like rain. The overall effect is a high-tech, futuristic aesthetic.

Verificación de la integridad de archivos mediante funciones hash

Sumario

Práctica en Windows.....	2
Comando CertUtil.....	2
SHA256.....	2
MD5.....	3
Opcional QuickHash GUI.....	4
Práctica en Linux.....	8
Calculo de hash MD5 con Md5sum.....	8
Calculo de hash SHA-256 con Sha256sum.....	9
Explica cómo podrías utilizar estos comandos para verificar la integridad de un archivo descargado de internet.....	9
Extra comparativa con script.....	9
Resultado del Script.....	11
Comparación de Algoritmos (análisis).....	11
¿Por qué los algoritmos MD5 y SHA-1 ya no son recomendados para aplicaciones críticas?.....	11
Indica en qué situaciones podría ser aceptable utilizar MD5 en lugar de algoritmos más seguros como SHA-256 o SHA-512.....	12

Práctica en Windows

Utilizando la herramienta CertUtil en Windows, calcula el hash de un archivo que tengas disponible en tu ordenador (puede ser un archivo de texto pequeño o cualquier archivo que elijas). Usa al menos dos algoritmos diferentes (ej. MD5 y SHA256).

Copia y pega en el informe los comandos utilizados y los resultados obtenidos.

Opcional: Instala HashTab o QuickHash GUI y genera el hash de un archivo usando uno de estos programas. Explica brevemente tu experiencia con estas herramientas (¿te parecieron fáciles de usar?, ¿cuál es la ventaja de tener una interfaz gráfica?).

Haz captura del uso de la herramientas

Comando CertUtil

SHA256

CertUtil -hashfile [FILENAME] SHA256

```

Directorio de C:\proyecto\ASIR 2\SAD\UT2

07/11/2024  17:24    <DIR>          .
07/11/2024  17:24    <DIR>          ..
07/11/2024  17:07             1.380.554 hacking google shodan.pdf
                1 archivos          1.380.554 bytes
                2 dirs      3.652.657.152 bytes libres

C:\proyecto\ASIR 2\SAD\UT2>CertUtil -hashfile "hacking google shodan.pdf" SHA256
SHA256 hash de hacking google shodan.pdf:
18dba5d98b5d94cf63510b9c828ef016d5f829b9888618238eda989906ede55a
CertUtil: -hashfile comando completado correctamente.

C:\proyecto\ASIR 2\SAD\UT2>

```

MD5

CertUtil -hashfile [FILENAME] MD

```

C:\proyecto\ASIR 2\SAD\UT2>CertUtil -hashfile "hacking google shodan.pdf" MD5
MD5 hash de hacking google shodan.pdf:
dd8d3ddf236f50d3f19c2c5a0910a81a
CertUtil: -hashfile comando completado correctamente.

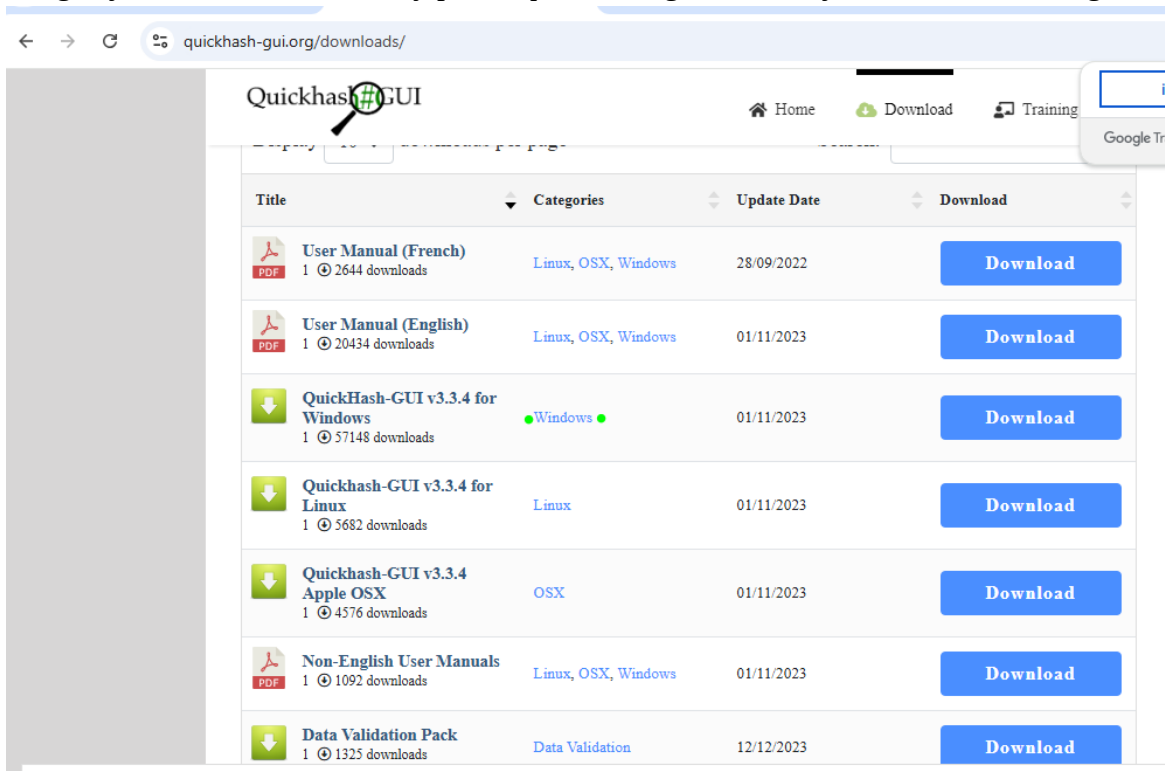
C:\proyecto\ASIR 2\SAD\UT2>

```

Opcional QuickHash GUI

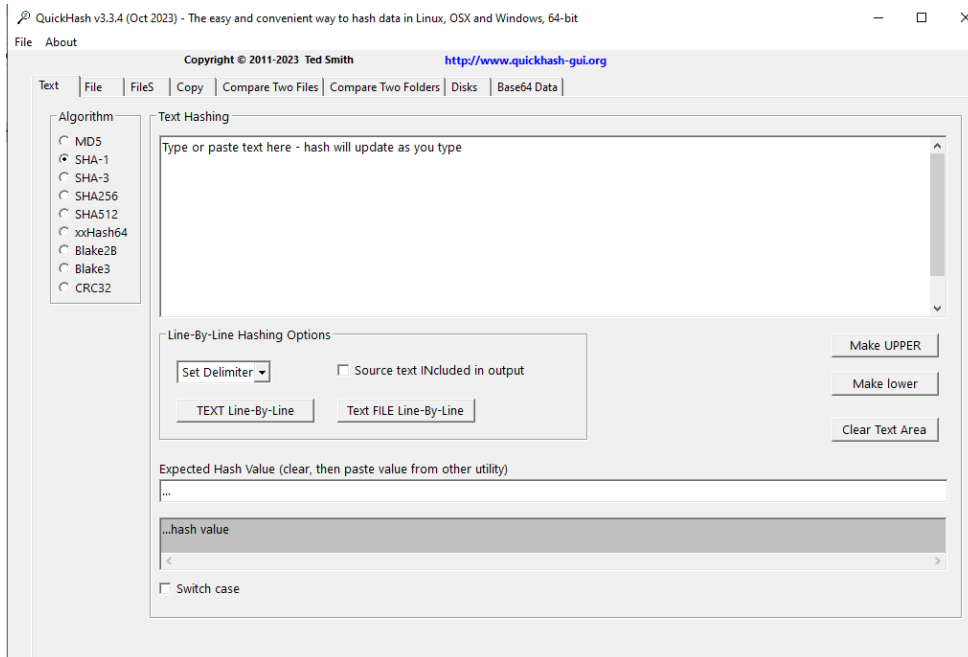
Descargamos el fichero

Luego ejecutamos el instalador y puede que nos salga un mensaje de Windows de seguridad.

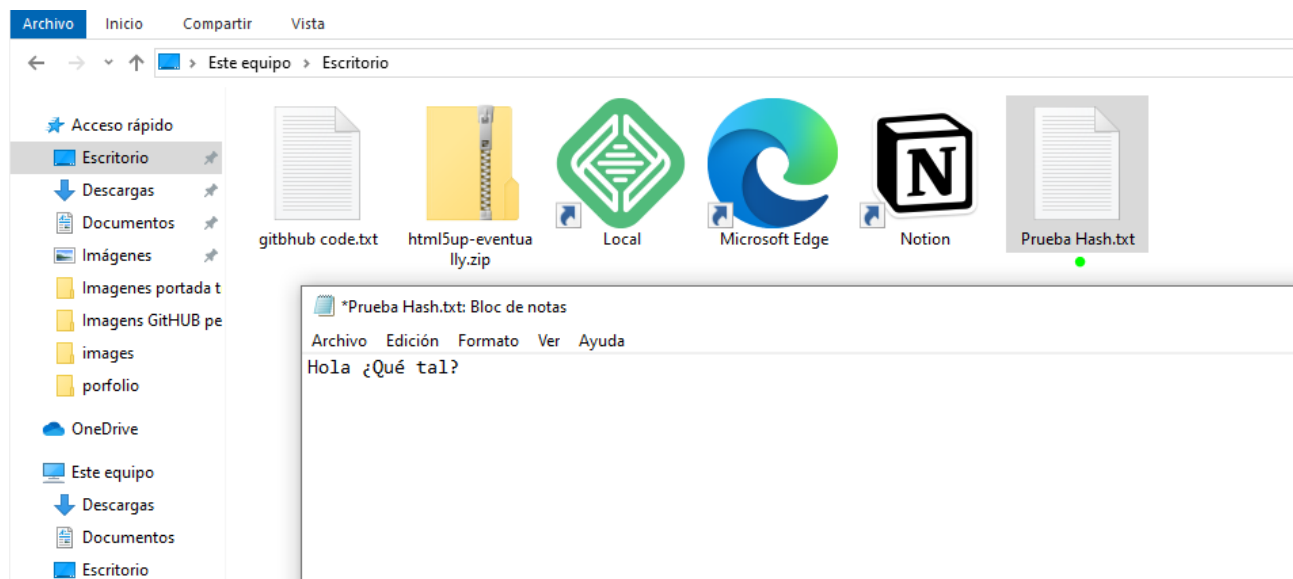


The screenshot shows the QuickHash-GUI website interface. The browser address bar displays "quickhash-gui.org/downloads/". The website has a navigation bar with "Home", "Download", and "Training" links. A search bar is visible on the right. The main content area features a table with columns: Title, Categories, Update Date, and Download. The table lists several items for download, including user manuals in French and English, and the QuickHash-GUI v3.3.4 for Windows, Linux, and Apple OSX. Each item has a download button and a download count.

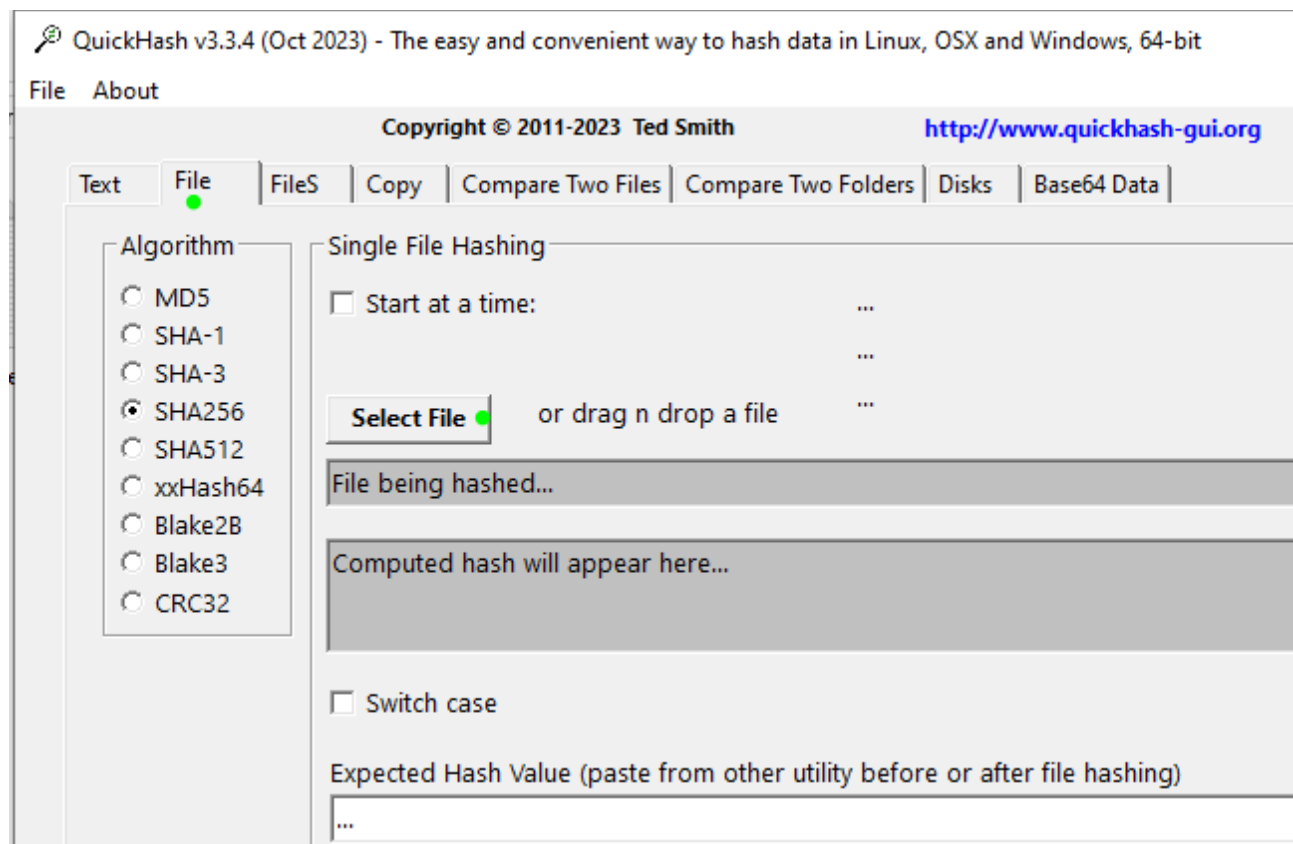
Title	Categories	Update Date	Download
User Manual (French) 1 2644 downloads	Linux, OSX, Windows	28/09/2022	Download
User Manual (English) 1 20434 downloads	Linux, OSX, Windows	01/11/2023	Download
QuickHash-GUI v3.3.4 for Windows 1 57148 downloads	Windows	01/11/2023	Download
Quickhash-GUI v3.3.4 for Linux 1 5682 downloads	Linux	01/11/2023	Download
Quickhash-GUI v3.3.4 Apple OSX 1 4576 downloads	OSX	01/11/2023	Download
Non-English User Manuals 1 1092 downloads	Linux, OSX, Windows	01/11/2023	Download
Data Validation Pack 1 1325 downloads	Data Validation	12/12/2023	Download



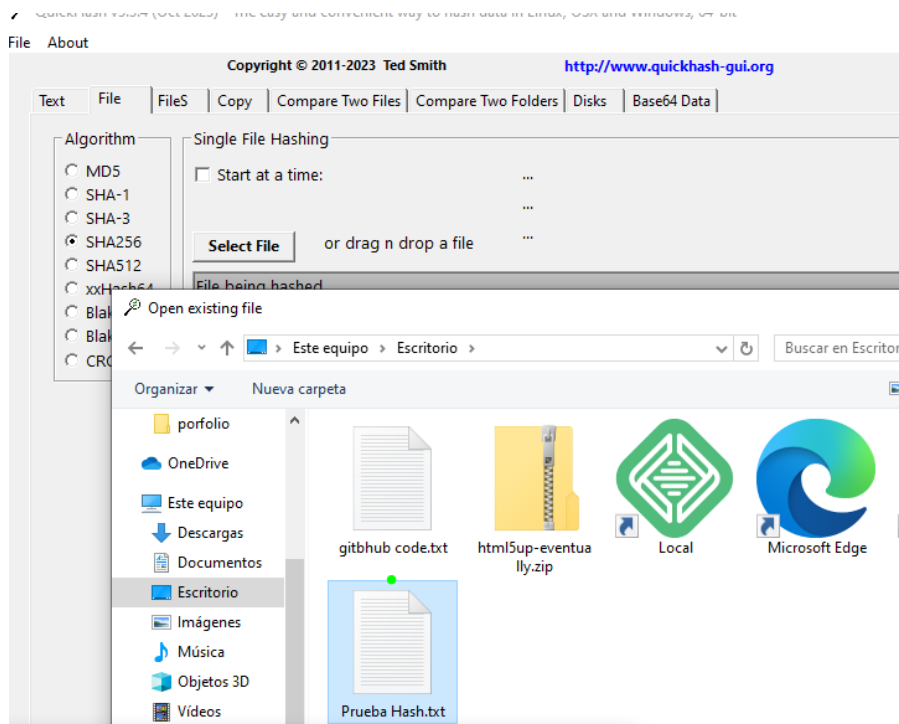
Ahora crearemos un fichero



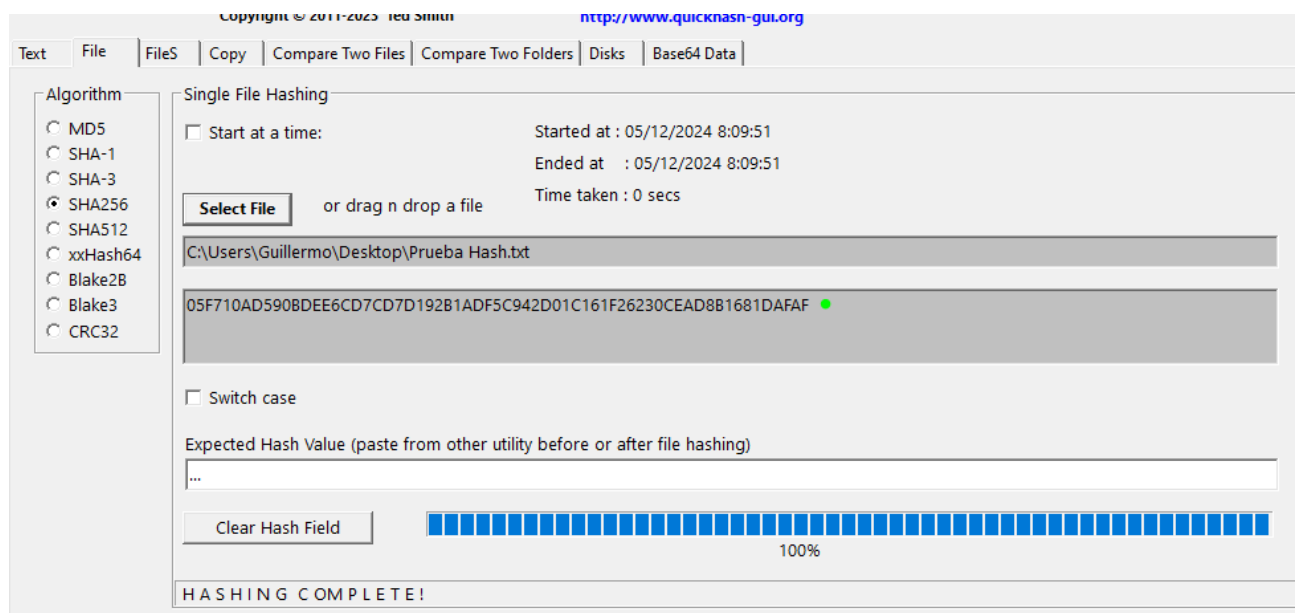
Vamos a la pestaña File y luego le damos a **Select File**



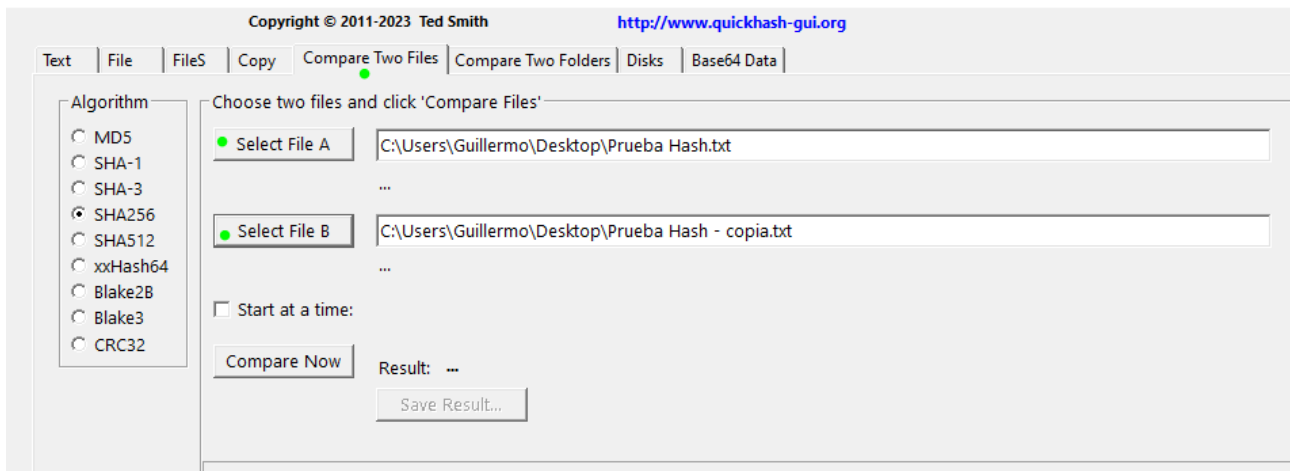
Seleccionamos el fichero.



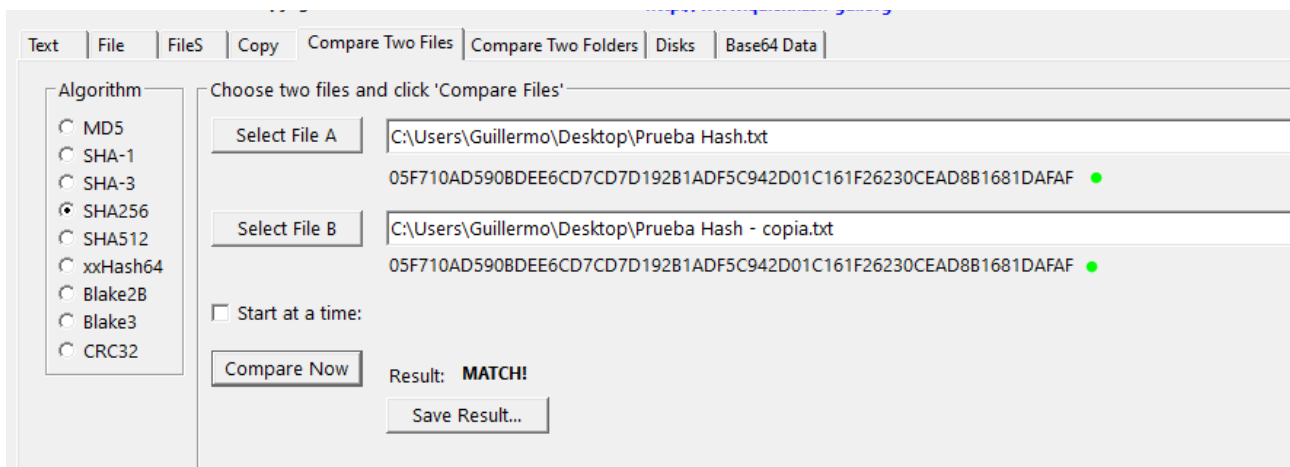
Aquí vemos que deja nuestro hash



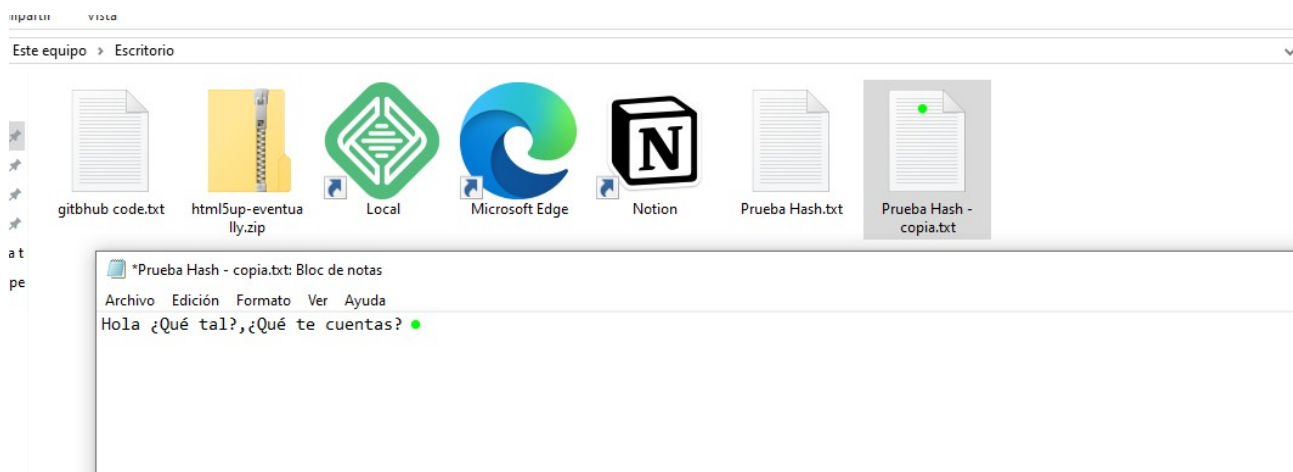
Ahora haremos una copia del fichero y lo compararemos. Para ello nos iremos a la pestaña **Compare Two Files** y en **Select File A** y **Select File B** cargamos los dos ficheros. Luego le damos en **Compare Now**



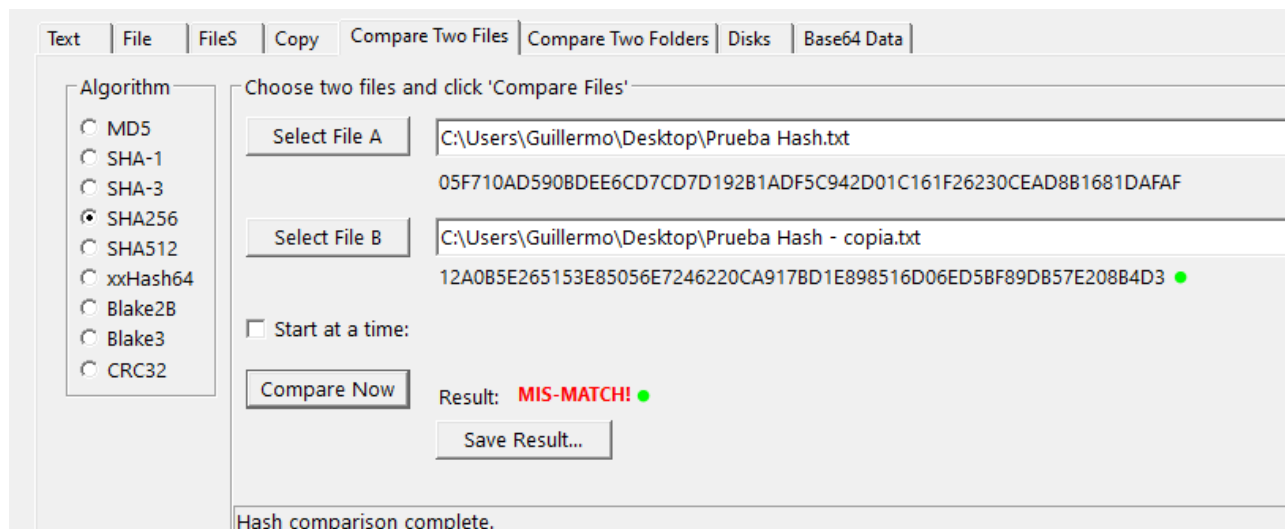
Podemos observar que tienen el mismo hash



Ahora modificaremos uno de los ficheros.



Si lo volvemos a comparar vemos que al modificar el fichero no sale el mismo hash.



¿Te parecieron fáciles de usar?

Si, es una herramienta bastante intuitiva.

¿Cuál es la ventaja de tener una interfaz gráfica?

En mi opinión te facilita mucho su uso, ya que no necesitas recordar comandos. La interfaz gráfica es bastante clara.

Práctica en Linux

Calculo de hash MD5 con Md5sum

Creamos un fichero para hacer la prueba.

```
guillermo@guillermo-VirtualBox: ~  
guillermo@guillermo-VirtualBox:~$ cat pruebahas.txt  
Hola ¿Qué tal?  
guillermo@guillermo-VirtualBox:~$ md5sum pruebahas.txt  
6d4a0a41687a8e52561e2588fe0d754c pruebahas.txt  
guillermo@guillermo-VirtualBox:~$ md5sum pruebahas.txt >> resultadosMD5.txt  
guillermo@guillermo-VirtualBox:~$ cat resultadosMD5.txt  
6d4a0a41687a8e52561e2588fe0d754c pruebahas.txt  
guillermo@guillermo-VirtualBox:~$
```


Calculo de hash SHA-256 con Sha256sum

Usaremos el mismo fichero del apartado anterior.

```
guillermo@guillermo-VirtualBox: ~  
guillermo@guillermo-VirtualBox:~$ cat pruebahas.txt  
Hola ¿Qué tal?  
guillermo@guillermo-VirtualBox:~$ sha256sum pruebahas.txt  
6138e2abf032d61fc63af162b2f5039c680650d371cc45fcc14146ea1bb6e3d2 pruebahas.txt  
guillermo@guillermo-VirtualBox:~$ sha256sum pruebahas.txt >> resultado.txt  
guillermo@guillermo-VirtualBox:~$ cat resultado.txt  
6138e2abf032d61fc63af162b2f5039c680650d371cc45fcc14146ea1bb6e3d2 pruebahas.txt  
guillermo@guillermo-VirtualBox:~$
```

Explica cómo podrías utilizar estos comandos para verificar la integridad de un archivo descargado de internet.

La mejor forma es ver si la pagina donde nos hemos descargado el fichero nos ofrece su hash, luego podríamos usar uno de esos comandos.

Extra comparativa con script

Con ayuda de mi primo le dije que si me podía hacer un script con el siguiente prompt:

“he visto que hay un programa que se llama quihash y te compara ficheros, he pensado que podríamos hacer un script donde te pregunte:
que tipo de hash quieres comprobar (m)Md5 y (s)SHA-256:
luego diga. diga la ruta absoluta del archivo A:
luego diga, diga la ruta absoluta del archivo B:
si los resultados no son el mismo hash diga. Los archivos no coinciden y muestre los hashes
y si coinciden que digan los archivos coinciden y muestre los hash
Seria para hacerlo en Ubuntu. no se si es posible aplica las mejoras que consideres”

Me arrojé el siguiente script:

```
#!/bin/bash  
  
# Solicitar el tipo de hash  
echo "¿Qué tipo de hash deseas comprobar?"  
echo "(m) MD5"  
echo "(s) SHA-256"  
read -p "Selecciona una opción (m/s): " tipo_hash  
  
# Validar la entrada del tipo de hash  
if [[ "$tipo_hash" == "m" || "$tipo_hash" == "M" ]]; then
```

```

comando_hash="md5sum"
nombre_hash="MD5"
elif [[ "$tipo_hash" == "s" || "$tipo_hash" == "S" ]]; then
    comando_hash="sha256sum"
    nombre_hash="SHA-256"
else
    echo "Opción no válida. Por favor, selecciona 'm' o 's'."
    exit 1
fi

# Solicitar las rutas de los archivos
read -p "Ingresa la ruta absoluta del archivo A: " archivo_a
read -p "Ingresa la ruta absoluta del archivo B: " archivo_b

# Verificar que los archivos existan
if [[ ! -f "$archivo_a" ]]; then
    echo "El archivo A no existe: $archivo_a"
    exit 1
fi

if [[ ! -f "$archivo_b" ]]; then
    echo "El archivo B no existe: $archivo_b"
    exit 1
fi

# Calcular los hashes
hash_a=$(comando_hash "$archivo_a" | awk '{print $1}')
hash_b=$(comando_hash "$archivo_b" | awk '{print $1}')

# Mostrar los resultados y verificar si coinciden
echo "-----"

```

```

echo "Tipo de Hash: $nombre_hash"

echo "Hash del archivo A ($archivo_a): $hash_a"

echo "Hash del archivo B ($archivo_b): $hash_b"

echo "-----"

if [[ "$hash_a" == "$hash_b" ]]; then

    echo "✅ Los archivos coinciden."

else

    echo "❌ Los archivos NO coinciden."

fi

```

Resultado del Script

```

guillermo@guillermo-VirtualBox:~/Documentos$ ls
archivo_1_copia.txt  archivo_1.txt  Auditoria.txt  script_hash.sh
guillermo@guillermo-VirtualBox:~/Documentos$ ./script_hash.sh
¿Qué tipo de hash deseas comprobar?
(m) MD5
(s) SHA-256
Selecciona una opción (m/s): s
Ingresa la ruta absoluta del archivo A: /home/guillermo/Documentos/archivo_1.txt
Ingresa la ruta absoluta del archivo B: /home/guillermo/Documentos/archivo_1_copia.txt
-----
Tipo de Hash: SHA-256
Hash del archivo A (/home/guillermo/Documentos/archivo_1.txt): fd2a3feda5b4b53a99be0ae0e5f239536c23d23ecbd285165c603310c749e5b9
Hash del archivo B (/home/guillermo/Documentos/archivo_1_copia.txt): fd2a3feda5b4b53a99be0ae0e5f239536c23d23ecbd285165c603310c749e5b9
-----
✅ Los archivos coinciden.
guillermo@guillermo-VirtualBox:~/Documentos$ nano archivo_1.txt
guillermo@guillermo-VirtualBox:~/Documentos$ cat archivo_1.txt
¿hola que tal?Modificado

guillermo@guillermo-VirtualBox:~/Documentos$ ./script_hash.sh
¿Qué tipo de hash deseas comprobar?
(m) MD5
(s) SHA-256
Selecciona una opción (m/s): s
Ingresa la ruta absoluta del archivo A: /home/guillermo/Documentos/archivo_1.txt
Ingresa la ruta absoluta del archivo B: /home/guillermo/Documentos/archivo_1_copia.txt
-----
Tipo de Hash: SHA-256
Hash del archivo A (/home/guillermo/Documentos/archivo_1.txt): 0b539e3415fac55ca1ab921dd87113a7883bdefc6feeb52a12d0f7f2a06612ff
Hash del archivo B (/home/guillermo/Documentos/archivo_1_copia.txt): fd2a3feda5b4b53a99be0ae0e5f239536c23d23ecbd285165c603310c749e5b9
-----
❌ Los archivos NO coinciden.
guillermo@guillermo-VirtualBox:~/Documentos$

```

Comparación de Algoritmos (análisis)

¿Por qué los algoritmos MD5 y SHA-1 ya no son recomendados para aplicaciones críticas?

Mientras que MD5 produce un hash de 128-bit, SHA1 genera un hash de 160-bit (20 bytes). En formato hexadecimal, es un entero de 40 dígitos de largo. Como MD5, fue diseñado para aplicaciones de criptología, pero pronto se le encontró vulnerabilidades también. Hoy en día, ya no es considerado ser menos resistente de atacar que MD5.

Actualmente existe una versión mejorada con SHA-2 que genera un hash de 256 bits y 512 bits, pero en 2015 salió SHA-3 que genera un hash de 224, 256, 384, 512 bits, es más rápido para el hardware.

Ejemplo.

Imagina una plataforma de almacenamiento en la nube que utiliza SHA-1 para proteger las contraseñas de los usuarios. Un atacante podría crear dos archivos diferentes que produzcan el mismo hash SHA-1. Si uno de esos archivos es una contraseña legítima y el otro es una contraseña falsa, el atacante podría acceder al sistema utilizando la contraseña falsa, comprometiendo la seguridad de los datos de los usuarios.

Indica en qué situaciones podría ser aceptable utilizar MD5 en lugar de algoritmos más seguros como SHA-256 o SHA-512.

Verificación de integridad de archivos

MD5 se puede usar para verificar que un archivo no ha sido alterado durante la transferencia. Por ejemplo, al descargar un archivo desde una fuente confiable, puedes comparar el hash MD5 del archivo descargado con el hash proporcionado por el remitente.

Sistemas integrados y entornos de bajos recursos

En sistemas embebidos o dispositivos con recursos limitados, MD5 puede ser una opción viable debido a su menor consumo de recursos en comparación con algoritmos más complejos como SHA-256 o SHA-512.

Deduplicación de datos

MD5 se puede utilizar para identificar y eliminar duplicados en sistemas de almacenamiento, aunque no sea la opción más segura.

Sistemas de control de versiones

Algunos sistemas de control de versiones pueden seguir utilizando MD5 para la verificación de archivos, aunque se recomienda migrar a algoritmos más seguros cuando sea posible.