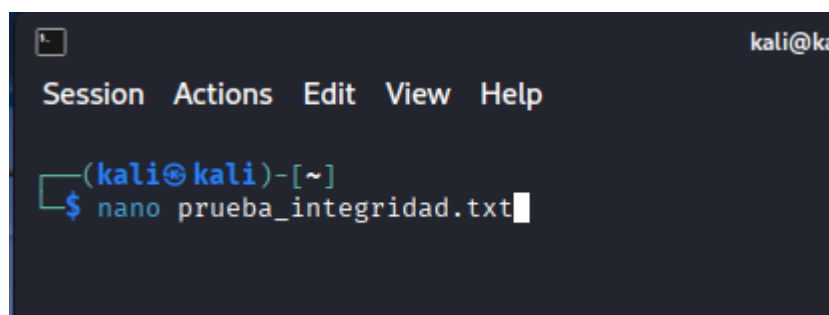


Práctica de Integridad de Archivos: Empleo de Checksums

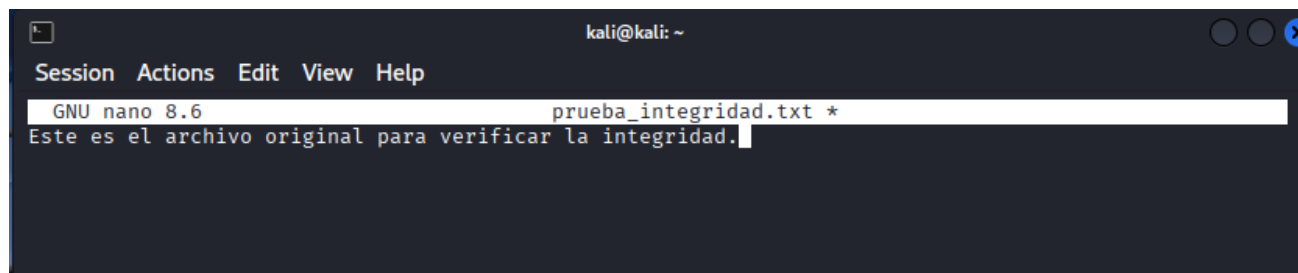
Paso 1: Creación de un archivo de texto

Crea un archivo de texto llamado **prueba_integridad.txt** que contenga una breve frase. Este archivo se utilizará para generar un checksum inicial.

```
nano prueba_integridad.txt
```



```
kali@kali: ~  
Session Actions Edit View Help  
(kali@kali)-[~]  
$ nano prueba_integridad.txt
```



```
kali@kali: ~  
Session Actions Edit View Help  
GNU nano 8.6 prueba_integridad.txt *  
Este es el archivo original para verificar la integridad.
```

Paso 2: Generación del Checksum

En Linux, utiliza el comando **sha256sum** o **md5sum** para generar el checksum. Anota el resultado del checksum en un documento.

```
sha256sum prueba_integridad.txt
```

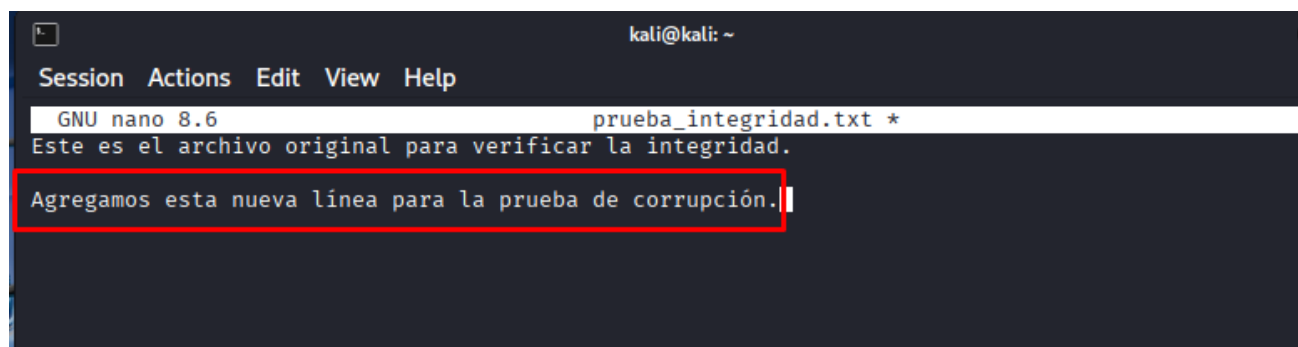


```
kali@kali: ~  
Session Actions Edit View Help  
(kali@kali)-[~]  
$ nano prueba_integridad.txt  
(kali@kali)-[~]  
$ sha256sum prueba_integridad.txt  
7bfbb85dca7d120cet2b77833f0ffb67cbdf17f49071842a03962b3876be9428 prueba_integridad.txt  
(kali@kali)-[~]  
$
```

Paso 3: Modificación del archivo

Realiza una pequeña modificación en el archivo de texto, como agregar una nueva línea o modificar una palabra.

nano prueba_integridad.txt

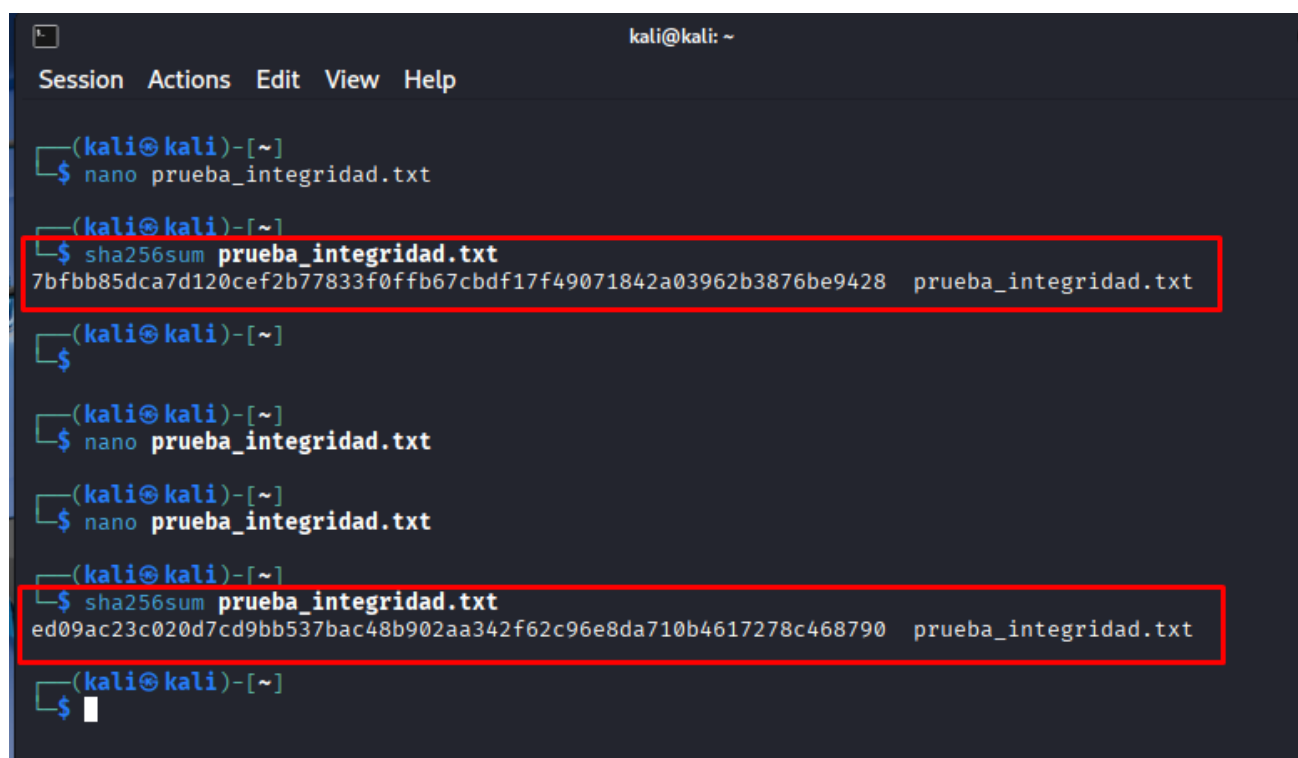


```
kali@kali: ~
Session Actions Edit View Help
GNU nano 8.6 prueba_integridad.txt *
Este es el archivo original para verificar la integridad.
Agregamos esta nueva línea para la prueba de corrupción.

```

Paso 4: Generación del nuevo checksum

Después de modificar el archivo, genera nuevamente el checksum. Compara este nuevo valor con el checksum original y observa las diferencias.



```
kali@kali: ~
Session Actions Edit View Help

(kali@kali)-[~]
$ nano prueba_integridad.txt

(kali@kali)-[~]
$ sha256sum prueba_integridad.txt
7bfbb85dca7d120cef2b77833f0ffb67cbdf17f49071842a03962b3876be9428 prueba_integridad.txt

(kali@kali)-[~]
$

(kali@kali)-[~]
$ nano prueba_integridad.txt

(kali@kali)-[~]
$ nano prueba_integridad.txt

(kali@kali)-[~]
$ sha256sum prueba_integridad.txt
ed09ac23c020d7cd9bb537bac48b902aa342f62c96e8da710b4617278c468790 prueba_integridad.txt

(kali@kali)-[~]
$

```

Análisis

1. ¿Por qué crees que el checksum cambia cuando modificas el archivo?

El checksum cambia porque el algoritmo hash (SHA-256) está diseñado para ser extremadamente sensible a cualquier alteración. Incluso un cambio de un solo carácter (un bit) en el archivo de texto hace que la fórmula matemática del hash genere un resultado completamente diferente. Este principio se conoce como el "efecto avalancha".

2. ¿Qué pasaría si solo cambias una letra del archivo? ¿El cambio sería igualmente notable en el checksum?

Sí, el cambio sería igualmente notable. Si solo se modifica una letra o un signo de puntuación, el checksum resultante sería tan distinto del checksum original como si se hubiera modificado todo el archivo. La longitud del checksum (la cadena de números y letras) seguiría siendo la misma, pero el valor de la cadena sería totalmente diferente.

3. ¿Cómo podrías utilizar los checksums para garantizar que un archivo no ha sido alterado después de su transferencia en una red?

La forma de utilizar los checksums es mediante la comparación de valores antes y después de la transferencia:

- a. Generación del Valor Inicial (Emisor):** El emisor del archivo genera el checksum del archivo original y lo publica o lo adjunta de manera segura antes de subirlo a la red.
- b. Generación del Valor Final (Receptor):** Una vez que el archivo es descargado de la red, el receptor genera su propio checksum localmente (como se realizó en el Paso 4 de la práctica).
- c. Verificación de la Integridad:** Si el checksum generado por el receptor coincide exactamente con el valor publicado por el emisor, se confirma que el archivo es idéntico al original. Si los valores no coinciden, es una prueba de que el archivo fue alterado, corrompido o manipulado durante la transferencia a través de la red.