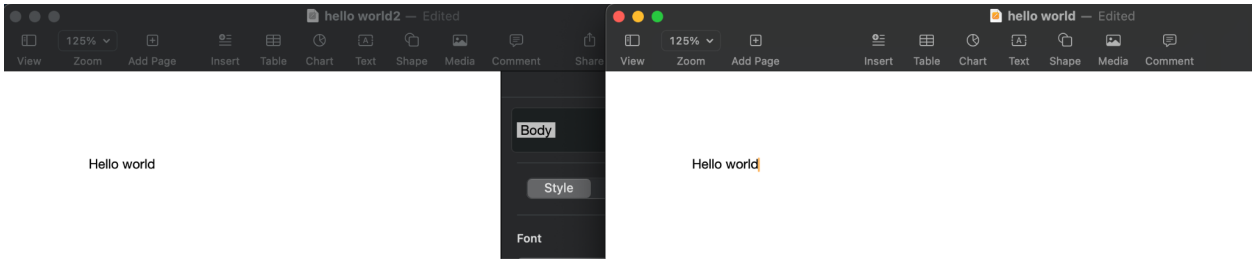
Hashing with Bash

Project description

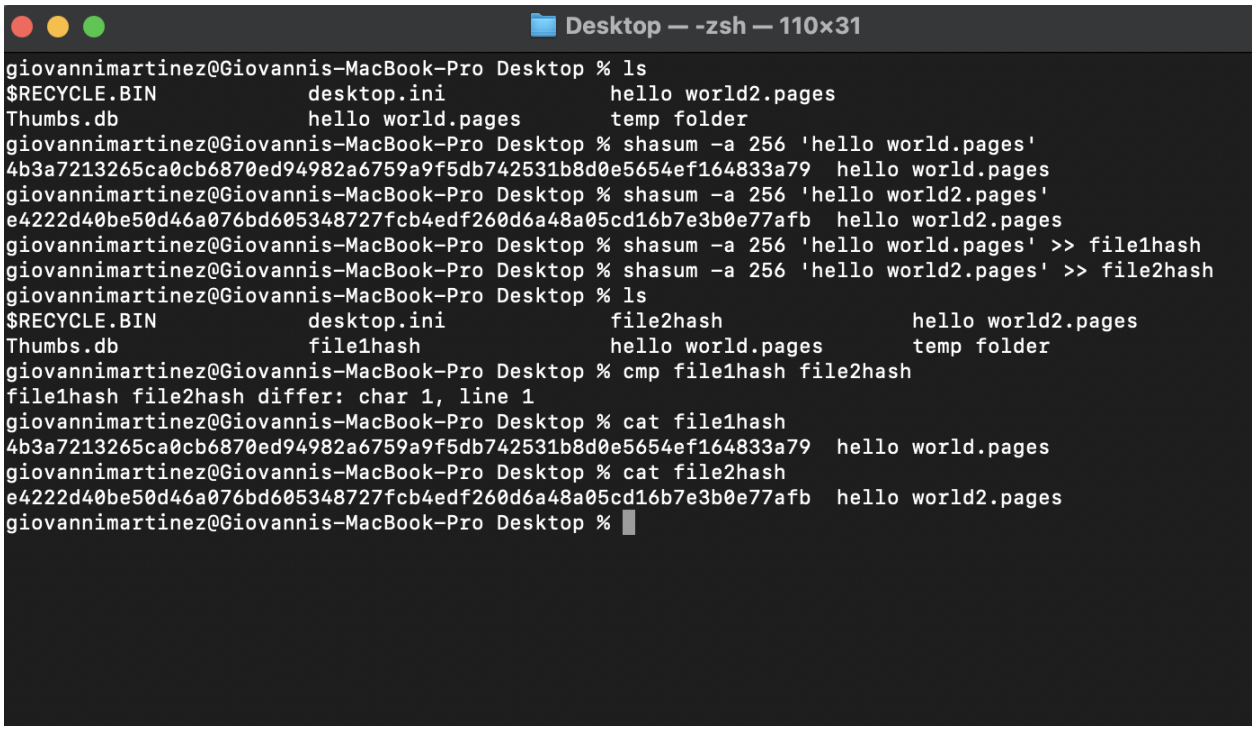
This project required me to use Linux bash commands in a CLI environment in order to determine data integrity through hashing as it relates to the security triad---Confidentiality, Integrity, and Availability. SHA-256 hashes were generated from two files in order to determine if the files had been tampered with.

File Comparison

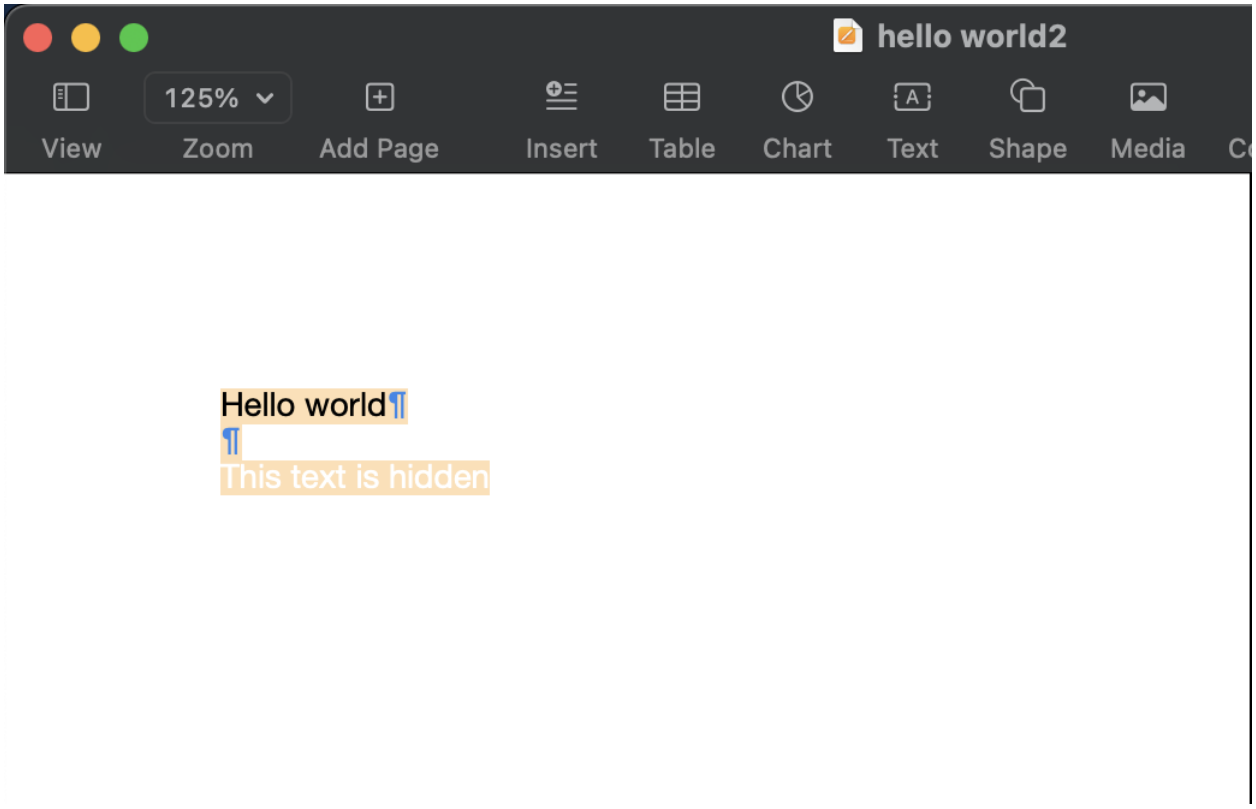


The two files are visually indistinguishable from one another. At first glance, a person could assume no data has been corrupted/deleted or manipulated in any way in the copied file (hello world2). However, let's inspect these two files deeper in the terminal.

Generating Hashes, Data Integrity, and Analysis



Using my knowledge of bash, I was able to inspect the two files on a deeper level than just visually. I used the **shasum** command in order to generate SHA256 hashes for both files and compare the hashes. If the copied file had maintained file integrity, both of the generated hashes should be identical. After generating the hashes, I use the simple **cat** command in order to reveal the file contents of the newly generated files (file1hash & file2hash) that contained the hashes. Although we can visually see they don't match, I nonetheless used the **cmp** command to compare the contents of the two files. From my gatherings, we can see that hello world2 did NOT maintain file integrity.



As we can see here. There did appear to be some corruption in the file.

Summary

As a security analyst, implementing effective security controls against diverse threats is crucial. Hashing plays a pivotal role in this context, utilizing hash functions—algorithms generating codes that can't be reversed—to uniquely identify file contents. These hash values act as distinct identifiers, allowing detection of file alterations. For instance, when a malicious program differs even slightly from an original, it generates a unique hash value, aiding security teams in pinpointing and addressing potential risks. While numerous tools exist to compare hashes in various scenarios, understanding how to manually compare hashes using Linux commands is essential for security analysts. This lab activity involved creating hash values for two files and utilizing Linux commands to manually inspect and discern the differences between them.