Hash Functions, Data Integrity and Digital Signatures

Reading

1h

Status | Incomplete

Introduction

This reading covers Hash Functions, Data Integrity, and Digital Signatures, which is a crucial topic for new cybersecurity professionals looking to build a strong foundation in encryption. Hash functions are an essential tool for ensuring data integrity, allowing organizations to verify the authenticity of data and detect any unauthorized changes or tampering. Digital signatures provide a means of authenticating the origin of data, ensuring that it comes from a trusted source.

In this reading, we will explore the basics of hash functions, data integrity, and digital signatures, their practical applications, and their importance for modern cyber security practices. By understanding the importance of these cryptographic techniques, you will be better equipped to design, implement and manage secure cryptographic systems and workflows, ensuring that data remains protected from unauthorized access and tampering.

Hash Function Overview

A **hash function** is an algorithm that maps variable-length input data to fixed-length data. The values returned by a hash function are hash values, also known as hashes. The hash function is used to resolve the integrity principle of cryptography, ensuring that the message has not been altered or tampered with during transmission.

Communication problems can cause the message not to reach the person in its correct form, which is then considered to be corrupted. How do you ensure that your messages are received in the correct order?

HMACs sign packets to ensure the information is the same as sent. HMACs ensure integrity through a keyed hash, the result of a mathematical calculation in a message with a shared secret key.

A **hash** is often described as a signature on the packet. It uses a shared secret key, while a digital signature uses the computer's public and private keys to send the data. A digital signature provides non-repudiation, which is not guaranteed by hash. Non-repudiation ensures that the communication may originate from a person whose identity can be verified.

Hash functions are also called hand functions, which is easy to determine the hash from the message. In bidirectional functions, the original message can be determined from its converted form. Encryption schemes and decryption are examples of bidirectional functions.

The hash is a cryptographic checksum or a code of message integrity (MIC) that each party must compute to verify the message. For example, the computer sending the data uses the hash function and a shared key to calculate the checksum of the message by including it with the package. The computer receiving the data must perform the same hash function on the incoming message as

on the shared key, and compare it with the original one (included in the sender). If the message is altered while in transit, the hash values will differ, and the packet will be rejected.

Main Features of a Hash Function

- Fixed-size output (hash): hash functions take a message (input) of any size and always produce the same size output. They are usually smaller than the input data, representing the original data (also known as digest).
- The efficiency of operation: hash functions must be efficient in their computation, as they are typically used for data verification.
- Deterministic: a message will always generate the same hash value.

Hash Function Properties

To be used in the field of cryptography, a hash function must have specific properties:

- **Preimage hardiness:** this means that it is computationally infeasible to reverse a hash. If a hash function produces a z value, it must be arduous to find an input x that contains the hash value z. This property protects against someone who has only the hash value and is trying to find the input value.
- **Second preimage resistance:** given an input x and a hash z, finding a second input that yields the same z hash should be challenging and also known as poor collision resistance. This property protects against attack by someone who has the input value and its hash and wants to substitute the input as a valid message.
- **Collision resistance:** it should be challenging to find two different inputs of any size that result in the same hash, also known as strong collision resistance. Since a hash function has a fixed output value, it is impossible not to have a collision. This property only guarantees that it is complicated to find. If a function has this property, it automatically has resistance to the second preimage.

Popular Functions

Below are some commonly used hash functions:

- Message Digest (MD): it was one of the most used functions for several years (and still is in some cases). The MD family has MD2, MD4, MD5, and MD6 functions and has been adopted as an Internet standard for a long time. It has a 128-bit hash. The MD5 function was used extensively in the software world to ensure file integrity. For example, a server can provide an MD5 hash value of a pre-computed file for the user to compare when they have downloaded it. In 2004, vulnerabilities were found in this algorithm, and it is no longer recommended.

- Secure Hash Function (SHA): the SHA family consists of four algorithms: SHA-0, SHA-1, SHA-2, and SHA-3. Although they are from the same family, they are structurally different. The original 160-bit version of SHA-0 was published in 1993. Due to some flaws, it did not become popular.

  The SHA-1 version was widely used on the Internet in applications and protocols like SSL for a long time (in 2016, SHA-2 became the SSL standard). The SHA-2 family contains four variations: SHA-224, SHA-256, SHA-384, and SHA-512, depending on the final number of bits in the hash. In 2012, the Keccak algorithm was chosen as the new SHA-3 standard.

- RIPEMD: the RIPEMD algorithm is an acronym for “RACE Integrity Primitives Evaluation Message Digest”. This family of functions was designed by a community of researchers and includes hashes of sizes 128, 160, 256, and 320 bits, with the RIPEMD-160 version being the most common. The original RIPEMD version and RIPEMD-128 are considered unsafe due to size and design.

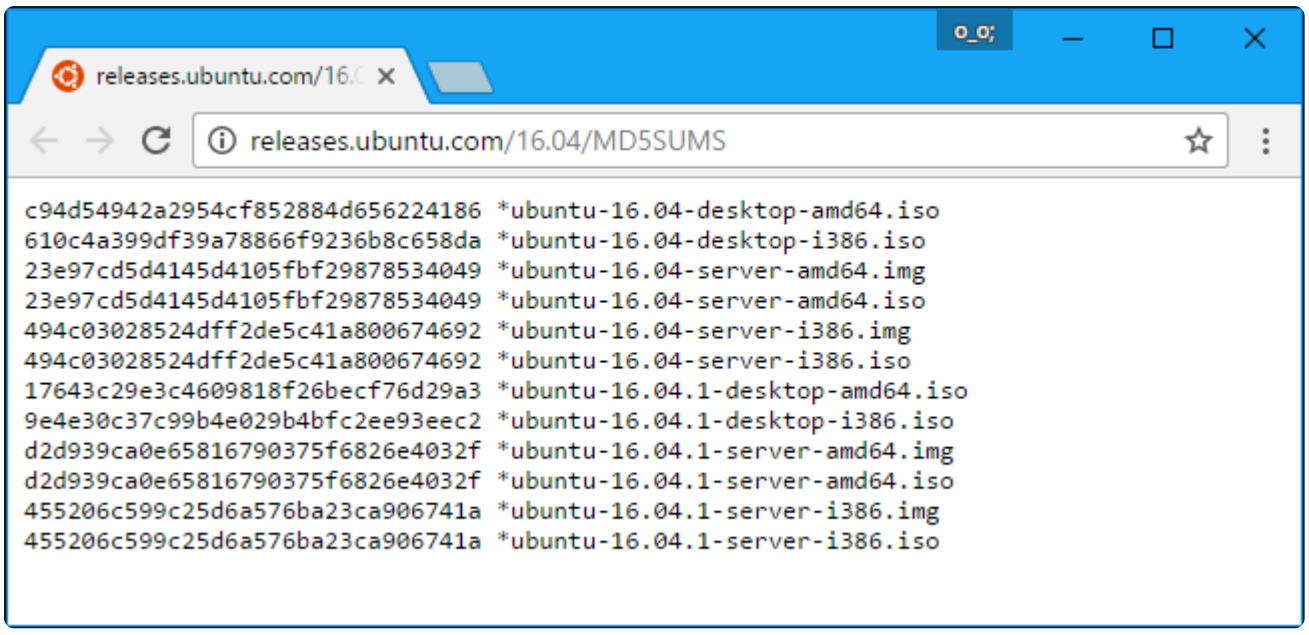
- Whirlpool: designed by a co-creator of AES, it is a derivative version of AES (encryption algorithm) itself. It has three versions released, known as WHIRLPOOL-0, WHIRLPOOL-T, and WHIRLPOOL, all containing a 512-bit hash.

- Checksum algorithms, such as CRC32, do not have these properties and therefore are not considered cryptographic hash functions.

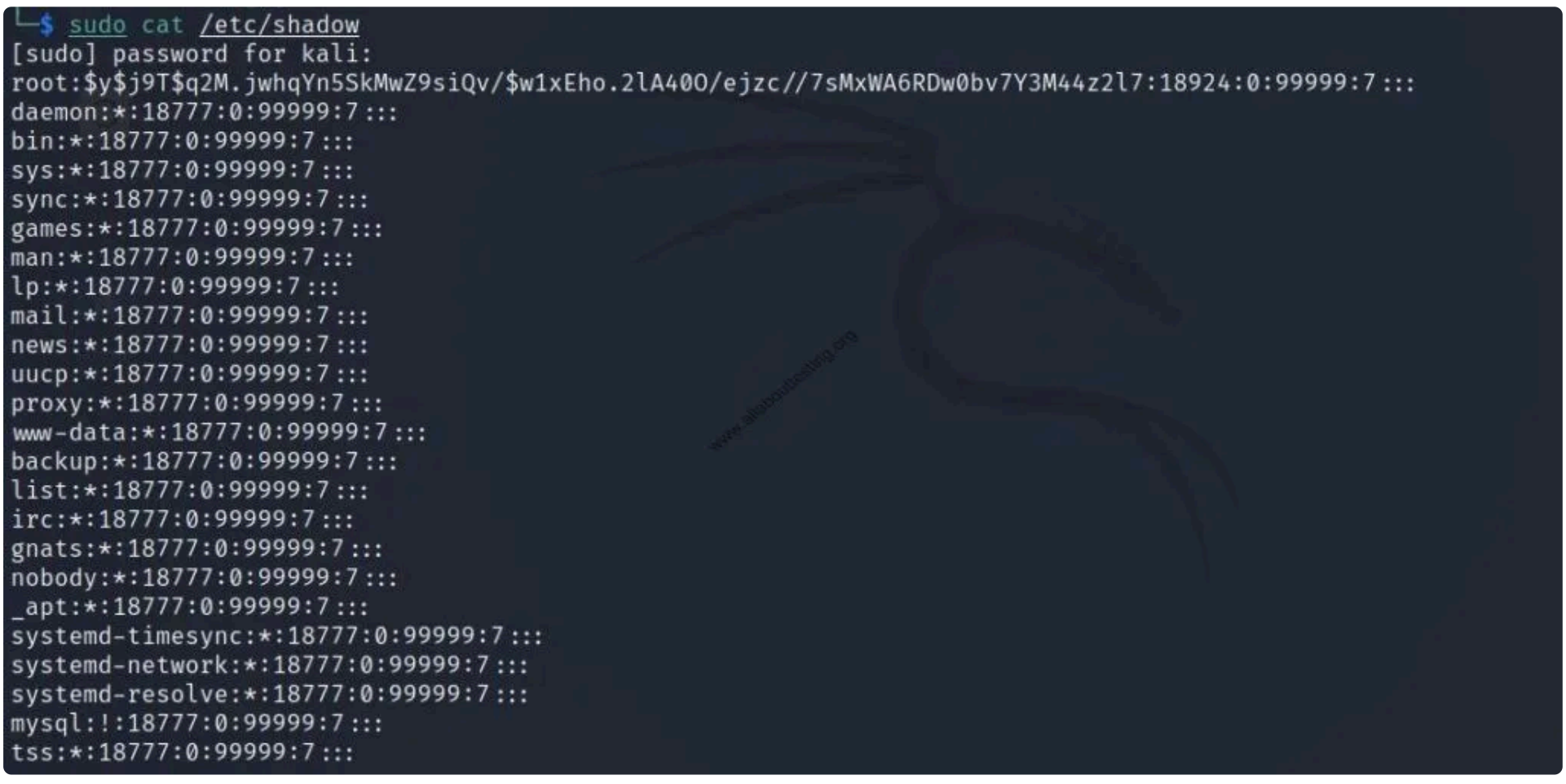
HASH Applications

Hash functions are used in many different scenarios. Here you will learn about some of them.

- Document integrity: check the integrity of documents/files/messages. You may have already seen on some download sites that it contains a hash value next to the file (MD5 or SHA-2, for example). In the image below, you can see an example of the hashes of files which you can download, and once you download the file, you can compare its original hash with the hash of the file you downloaded.



- Save passwords: saving and verifying passwords is another application. Rather than storing the password in plain view in a database, developers often save password hashes or a more complex value derived from the password (using salt or other steps). For example, the image below is the /etc/shadow file on Linux (Ubuntu). The password is saved as multiple rounds of salted SHA-512.



- Generate a unique ID: generate an (almost) unique ID of a specific document or message, as a cryptographic hash function can generate a single value from the contents of a document. In theory, collisions are possible in any hash function, but it's improbable, so most systems assume that their hash is collision-free.

  For example, Git uses a hash (SHA-1) to identify each commit made to a repository in GitHub. Later, if you want to go to a specific commit or reference it somehow, you can use this value to identify it in the system.

- Pseudo-random number generator: hashes can be used as pseudo-random number generators or key derivation. A simple way to generate a random sequence of numbers.

- Proof-of-work algorithms: protocol used to prevent cyberattacks such as DDoS and Spam. For a user to act, they must prove that they performed a task. This proof guarantees that the user spent time generating a response that satisfies an evaluator's condition.

Another well-known use of this system is in blockchains. Most algorithms calculate a hash value greater than a certain defined value (known as the mining difficulty). To find this hash value, miners must compute billions of different hashes until they find the one that satisfies the condition.

Exercise - Using Checksum

A checksum is a name given to the procedure for verifying the authenticity and integrity of a given file, which can be a photo, disk image, audio, video, text document, or virtually any other digital file format you may have on your computer.

Several different encryption formats (known as hash functions) can be used to perform checksums. However, today you will focus on two of the most common ones; they are MD5 and SHA-256. Although MD5 (which works in 128 bits) is still widely used, it is slowly being replaced by SHA-256 with its 256-bits hash value.

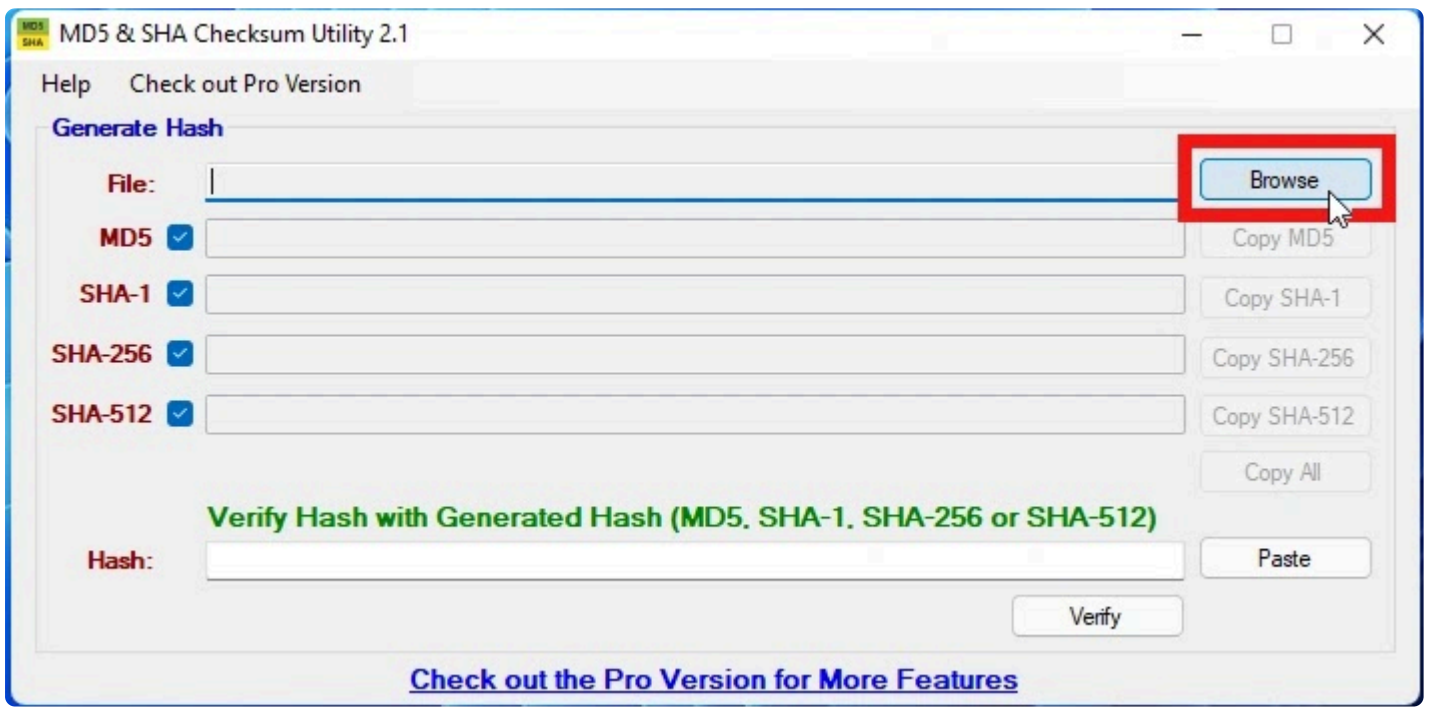
As explained before, one of the most common examples of checksums is the download of files, in this example case, the ".iso" installation images that can be found all over the Internet. In the vast majority of cases, on the download pages of the central Linux-based OSs, you will also find a security code to verify the download, which is the MD5 or SHA-256 sum of the ".iso" file in question.

After downloading the target file, you can verify the authenticity and integrity of that image. For that, open a Linux on the EVE environment, then open the terminal inside the directory where the file is located, and execute the following command:

```
$ sha256sum file_name.iso
```

And after the result, you can see the output of the command SHA-256 hash referring to the file that you just downloaded, and now you can compare the hash with that available on the website. And if it's the same hash number or checksum, the file was downloaded 100% correctly, and if both are identical, you will have proof that the downloaded file has not been interrupted or altered in any way.

You can do almost the same process using Microsoft Windows. Still, you have to use a specific tool, such as the "MD5 & SHA Checksum Utility", a free utility capable of generating MD5, SHA-1, SHA-256, and SHA-512 hashes of files in general and also verifying their integrity, all with a simple and easy-to-use interface, as you can see in the image below:



Or you can use a Powershell command on your Windows. For example, you can check the MD5 checksum using the command below:

```
C:\Get-FileHash <Name_of_file> -Algorithm MD5
```

You have to choose the algorithm from the list:

PowerShell Core (version 6 and 7): MD5, SHA1, SHA256, SHA384, and SHA512

Or, as an optional step, you can write your code and import the Python library hashlib:

```
import hashlib

inputFile = raw_input("Enter the name of the file:")

openedFile = open(inputFile)

readFile = openedFile.read()

md5Hash = hashlib.md5(readFile)

md5Hashed = md5Hash.hexdigest()

sha1Hash = hashlib.sha1(readFile)

sha1Hashed = sha1Hash.hexdigest()

print "File Name: %s" % inputFile

print "MD5: %r" % md5Hashed

print "SHA1: %r" % sha1Hashed
```

Conclusion

Now that you have got undersanding of hash functions & their properties, and how they can be used, let's learn how to manage cryptographic keys in the next reading.

✓ Mark Completed

Previous
Report to IT Manager

Next
Key Establishment Protocols and Management Techniques

How well did this activity help you to understand the content?

Let us know how we're doing

☆ ☆ ☆ ☆ ☆

W07D3

Wed Aug 7

- Lectures (1)
- Work (7)

  6 hrs

  - Configure Apache Web Server
  - Capture an HTTP and an HTTPS Wireshark PCAP

- Report to IT Manager
- Hash Functions, Data Integrity and Digital Signatures
- Key Establishment Protocols and Management Techniques
- Data Integrity Email
- Project Reading

W07D3 Schedule »