Message Integrity Attacks

4-1 Introduction

Data integrity is the maintenance and the assurance of the accuracy and consistency of data over its entire life-cycle, and is a critical aspect to the design, implementation and usage of any system which stores, processes, or retrieves data.

Message Integrity concerns about the validity of a transmitted message. Message integrity means that a message has not been tampered تحرف with or altered . The most common approach is to use a hash function that combines all the bytes in the message with a secret key and produces a message digest that is difficult to reverse.

Data integrity is the opposite of data corruption, which is a form of data loss. **data integrity aims to prevent unintentional changes to information**. Data integrity is not to be confused with data security, the discipline of protecting data from unauthorized parties.

Any unintended changes to data as the result of a storage, retrieval or processing operation, including malicious intent, unexpected hardware failure, and human error, **are examples for failure of data integrity**. **If the changes are the result of unauthorized access, it may also be a failure of data security**. Depending on the data involved this could manifest itself as benign as a single pixel in an image appearing a different color than was originally recorded, to the loss of vacation pictures or a business-critical

database, to even catastrophic كارثي loss of human life in a life-critical system.

4-2 Message Authentication

Message authentication is a mechanism or service used to verify the integrity of a message. Message authentication assures that data received are exactly as sent (i.e., contain no modification, insertion, deletion, or replay). In many cases, there is a requirement that the authentication mechanism assures that purported المزعوم identity of the sender is valid. When a hash function is used to provide message authentication, **the hash function value is often referred to as a message digest.** The essence جوهر of the use of a hash function for message authentication is as follows:

آلية العمل

The sender computes a hash value as a function of the bits in the message and transmits both the hash value and the message. The receiver performs the same hash calculation on the message bits and compares this value with the incoming hash value. If there is a mismatch, the receiver knows that the message (or possibly the hash value) has been altered.

The hash function must be transmitted in a secure fashion, Why?. That is, the hash function must be protected so that if an adversary alters or replaces the message, it is not feasible قابلة for adversary to also alter the hash value to fool the receiver.

4-3 What Are Hash Functions

A hash function is simply a function that takes in input value, and from that input creates an output value deterministic of the input

value. For any x input value, you will always receive the same y output value whenever the hash function is run. In this way, every input has a determined output.

Q What is the difference between Message Integrity and message authenticity

Q what is the role of hash function in the message integrity?

A function is basically something that takes an input and from that input derives an output.

$$f(x) = y$$

A hash function is therefore something that takes an input (which can be any data - numbers, files, etc) and outputs a hash. A hash is usually displayed as a hexadecimal number.

$$md5(\text{"}hello\ world\text{"}) = 5eb63bbbe01eeed093cb22bb8f5acdc3$$

This is the hash function md5, which from any input data creates a 32 character hexadecimal output**. Hash functions are generally irreversible (one-way)**, which means you can't figure out the input if you only know the output – unless you try every possible input (which is called a brute-force attack).

Hash functions are often used for proving that something is the same as something else, without revealing the information beforehand. Here's an example.

مثال توضيحي

Let's say Alice is bragging to Bob that she knows the answer to the challenge question in their Math class. Bob wants her to prove that she knows the answer, without her telling him what it is. So, Alice hashes her answer (let's say the answer was 42) to produce this hash:

$$md5(42) = a1d0c6e83f027327d8461063f4ac58a6$$

Alice gives this hash to Bob. Bob can not find out what the answer is from this hash – but when he finds the answer himself, he can hash his answer and if he gets the same result, then he knows that Alice did indeed have the answer. Hashes are often used in this context of verifying information without revealing it to the party that is verifying.

Ref:

https://learncryptography.com/hash-functions/what-are-hash-functions

4-4 Why hash function is Irreversible?

A common question many people have upon learning about one-way cryptographic hash functions is how the "one-way" part works. A certain amount of data goes through a process and a result is given – shouldn't it be possible to do that same process in reverse, since we know what the process is in the first place?. It applies mostly the same to all hash functions. Hash functions essentially discard information in a very deterministic way - using the modulo operator.

الفقرة التالية للاطلاع فقط

For a quick review, modulus is essentially the same as saying “the remainder of” (applying to division). A quick example would be:

$$16 \ mod \ 5 = 1$$

What’s happening here is that we’re dividing 16 by 5, and the result of the operation is whatever is left over – or the remainder.

$$25 \ mod \ 10 = 5$$

Obviously if the number was evenly divisible my the modulo then it would result in zero, and if it was less than the number it would remain as the number itself. Now – back to the important part. Why is this so important in one-way hash functions? Because the modulo operation is not reversible. If the result of the modulo operation is 4 – that’s great, you know the result, but there are infinite possible number combinations that you could use to get that 4.

Another thing to consider is that **a lot of data is discarded during the hash process**. **While the input string may be as big as it wants, the output must always be of a set size determined by the function**. Because of this, a lot of data is thrown out. It would be impossible to figure out the original data of the function with just the resulting hash – as not much of that data is left – the only workable method is to brute force every possible combination. If we could reverse a hash, we would be able to compress data of any size into a mere few bytes of data!

Ref:

https://learncryptography.com/hash-functions/why-are-hashes-irreversible