Blog

Managed File Transfer and Network Solutions

What Is HMAC And How Does It Secure File Transfers?

Posted by John Carl Villanueva on Wed, Aug 31, 2016 @ 02:10 AM

Overview

Data integrity checks are vital to secure communications. They enable communicating parties to verify the integrity and authenticity of the messages they receive. In secure file transfer protocols like FTPS, SFTP, and HTTPS, data integrity/message authentication is usually achieved through a mechanism known as HMAC. In this post, we explain what HMAC is, its basic inner workings, and how it secures data transfers.

NEED HELP?

Importance of data integrity checks in secure file transfers

Business decisions and processes are highly dependent on accurate and reliable data. If data gets tampered and the alterations go unnoticed, it could affect decisions and processes down the line. So if your data has to be transmitted over a network, especially one as perilous as the Internet, you have to take precautionary measures to preserve its integrity or at least know if it underwent unauthorized alterations.

This is precisely the reason why secure file transfer protocols like FTPS, SFTP, and HTTPS are equipped with mechanisms for countering threats to data integrity. The most commonly used mechanism today is HMAC. Let me explain what it is.

What is HMAC?

HMAC stands for Keyed-Hashing for Message Authentication. It's a message authentication code obtained by running a cryptographic hash function (like MD5, SHA1, and SHA256) over the data (to be authenticated) and a shared secret key. HMAC is specified in RFC 2104.

HMACs are almost similar to digital signatures. They both enforce integrity and authenticity. They both use cryptographic keys. And they both employ hash functions. The main difference is that digital signatures use asymmetric keys, while HMACs use symmetric keys.

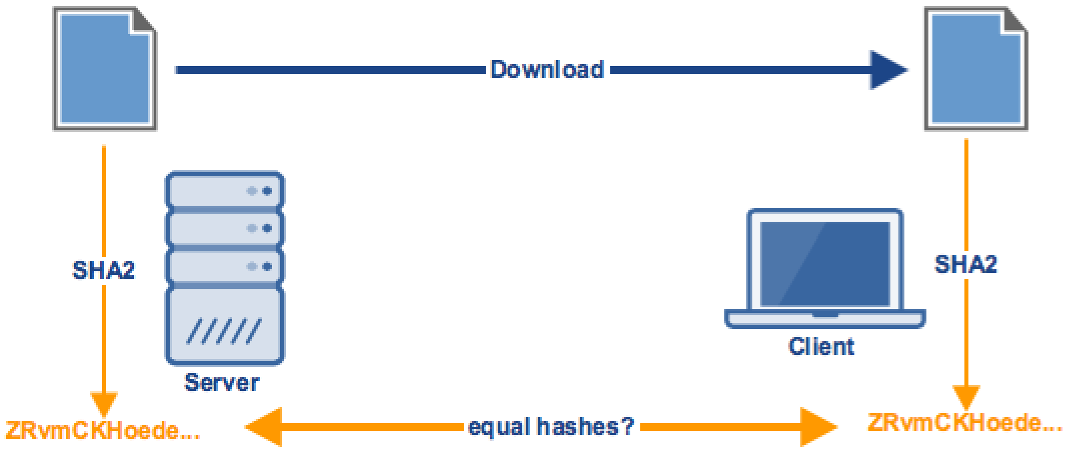
Recommended read: Symmetric vs Asymmetric Encryption

How HMAC works

To understand how HMAC works, let's first examine how a hash function (on its own) could be used for conducting a data integrity check on a file transfer. Let's say a client application downloads a file from a remote server. It's assumed that the client and server have already agreed on a common hash function, say SHA2.

Before the server sends out the file, it first obtains a hash of that file using the SHA2 hash function. It then sends that hash (a.k.a. message digest) along with the file itself. Upon receiving the two items (i.e. the downloaded file and the hash), the client obtains the

SHA2 hash of the downloaded file and then compares it with the downloaded hash. If the two match, then that would mean the file was not tampered along the way.



If an attacker manages to intercept the downloaded file, alter the file's contents, and then forward the tampered file to the recipient, that malicious act won't go unnoticed. That's because, once the client runs the tampered file through the agreed hash algorithm, the resulting hash won't match the downloaded hash. This will let the receiver know the file was tampered along he way.

So a hash function should do the trick then? Not so fast. While a hash function can establish data integrity, i.e. that the file or message wasn't altered along the way, it can't establish authenticity. How would the client know the message it received came from the legitimate source?

That's why secure file transfer protocols like FTPS, SFTP, and HTTPS use HMACs instead of just hash functions.When two parties exchange messages through those secure file transfer protocols, those messages will be accompanied by HMACs instead of plain hashes. An HMAC employs both a hash function and a shared secret key.

A shared secret key provides exchanging parties a way to establish the authenticity of the message. That is, it provides the two parties a way of verifying whether both the message and MAC (more specifically, an HMAC) they receive really came from the party they're supposed to be transacting with.

The secret key enables this capability because it's generated during key exchange, a preliminary process that requires the participation of the two parties. Only those two parties who participated in the key exchange would know what the shared secret key is. In turn, they would be the only ones who would be able to arrive at the same result if they compute the message's corresponding MAC using the shared secret key.

Why is HMAC suitable for file transfers?

Aside from its ability to enable data integrity and message authentication, another reason why HMAC is an excellent file transfer data integrity-checking mechanism is its efficiency. As discussed in the article *Understanding Hashing*, hash functions can take a message of arbitrary length and transform it into a fixed-length digest. That means, even if you have relatively long messages, their corresponding message digests can remain short, thereby allowing you to maximize bandwidth.

Choosing an HMAC

Because an HMAC's properties (especially its cryptographic strength) is highly dependent on its underlying hash function, a particular HMAC is usually identified based on that hash function. So we have HMAC algorithms that go by the names of HMAC-MD5, HMAC-SHA1, or HMAC-SHA256.

You've probably heard about the collision-related vulnerabilities of MD5. It's worth noting that HMAC-MD5, in spite of its underlying MD5 hash function, isn't as affected by those vulnerabilities. Regardless, SHA-1 is still cryptographically stronger than MD5 and SHA-2 (and its different forms, e.g. SHA-224, SHA-256, SHA-512) is likewise cryptographically stronger than SHA1, so you will want to take that into consideration.

So which HMAC should you use? You would normally choose an HMAC based on its underlying hash function. So, for example, you would want to use HMAC-MD5 if performance is more critical to you than security. On the other hand, if security is more critical, then you might want to use HMAC-SHA256 instead.

Need to perform secure file transfers?

Are you looking for a way to secure your file transfers? Try JSCAPE MFT Server, a managed file transfer server that supports FTPS, SFTP, HTTPS, and other secure file transfer protocols. JSCAPE MFT Server comes with a free, fully-functional evaluation edition. Download it now.

Download Now

Related posts

You might also want to read the following articles:

How To Install A SFTP Server on Windows

10 Ways to Make a Server to Server File Transfer Fit Enterprise Use

The SSH / SFTP Key Fingerprint And Its Role In Server Authentication

10 Essential Attributes of a Secure File Transfer

An Overview of How Digital Certificates Work

How To Set Up a Server To Server File Transfer

Topics: Security, Secure File Transfer, SFTP, FTPS
