What's the difference between FTPS, SFTP or FTP over SSH

Devolutions

WHEN STARTING A CONVERSATION ABOUT FTPS, SFTP OR FTP OVER SSH, IT MIGHT QUICKLY GET CONFUSING

We all know how sometimes between geeks we can start a discussion and quickly realize that those outside of our little group seems somewhat confused about our conversation. When starting a conversation about FTPS, SFTP or FTP over SSH, it might quickly get confusing, so I thought I would clear that up and give a little crash course about it!

FTP, or File Transfer Protocol, is a rather standard way to transfer files over a network, and even over the internet. It is an age old protocol that has been designed in a timewhere the only network users were computer nerds (like me!) and whose only malice was to create more software. Since then, things have changed and security has become a serious concern. FTP accounts need passwords for access, but those passwords are transferred in the clear and it would be easy for an attacker to get them by watching the network traffic.

This is why it was necessary to improve on FTP and add security to encrypt the network traffic as well as authenticate both the client and the server. This is where several flavors of FTP appeared: FTPS, SFTP, FTP over SSH. These terms can be quite confusing for a new user, and even amongst aficionados.

FTPS (IMPLICIT VS EXPLICIT)

FTPS stands for FTP over SSL. It is the same protocol as FTP, but adds a security layer through the use of SSL (Secure Sockets Layer). This usage of SSL can be done in two ways, it can be either implicit, or explicit.

Implicit FTPS starts by a security negotiation and then uses the FTP protocol normally over the encrypted connection. This provides the advantage that the FTP protocol can be used after the connection is established: it will be implicitly encrypted by the SSL connection. But it has the disadvantage of requiring that the client is aware of SSL and thus breaks compatibility with old clients.

This is where explicit FTPS comes in. The connection starts normally over an insecure connection and then the client can try to upgrade the connection to an encrypted one using FTP extended commands. This allows old clients to access a server in the old insecure way, although the server administrator could forbid it, then allows new clients to negotiate a secure connection.

FTP OVER SSH

FTP over SSH is quite different from SFTP. It is standard FTP tunneled through an SSH connection. Those of you who knows SSH forwarding would ask: "Then, I just need to open a tunnel for the FTP port and have FTP over SSH?" Well... not really.

This is because FTP uses more than one connection to work. If you open an SSH tunnel for the FTP port, you successfully secure the FTP "control" connection. However, data is transferred over another port which is usually at the discretion of either the server or the client. This makes it difficult to open a tunnel for the port required for the data connection. In order for FTP over SSH to be completely secured, the FTP client needs to be tightly integrated with the SSH client.

SFTP

SFTP stands for SSH File Transfer Protocol. SSH is an encrypted and secure communication protocol, and it provides an extension to transfer files. In fact, SFTP is completely different from FTP. It still does essentially the same job, but securely, and with better compatibility and formality than FTP.

Especially regarding directory listings, which is quite a challenge with FTP because there is no normalized way for an FTP server to respond to a client requesting a list. I will admit, this is mostly a programmer's concern, meaning that connecting to a SFTP server will yield correct operations assuredly, while there could be some obscure FTP server that could respond in way that baffles any FTP client GUI.

ACTIVE VS PASSIVE

This brings me the opportunity to clarify another aspect of FTP: active vs passive. This specifies how the data connection is performed. In active mode, the server will actively connect to a client data port. It's up to the FTP client to inform the server about its data port number.

Whereas, in passive mode, the server will wait for the client to connect to its data port. It's up to the FTP client to query the server for this port number. The choice between active and passive is also the responsibility of the client. Usually, passive mode is the preferred way. It makes it easier to pass through many types of proxy, some of which would allow only connection from the client to the server. I, for myself, is unaware of situations where active mode might be useful, but FTP has the flexibility to accommodate such a case. SFTP does not have this problem as it uses the same connection for both control and data transfer.

BINARY VS ASCII

There is one last point I would like to bring: binary transfer vs ASCII transfer. FTP clients have this option, but what does it really do? Remember I stated earlier that FTP is an age old protocol? Well, it was designed in a time when computers did not have all the same internal data representation. It was especially true for text.

This is where the ASCII transfer option was necessary. In order for a text file to be readable across different systems, a standard has been designed and it was up to the server to convert between its internal text representation and the network ASCII representation. Nowadays, all commonly used computers use the ASCII representation for their simple text files. With only one notable difference remaining: end of lines in Windows systems. The FTP standard does not specify formally how to treat these end of lines, so many servers will transmit and receive text files the same in both ASCI and binary transfer mode.

I hope this was helpful, now you can transfer files like never before, knowing a little more what happens on the wire, or the radio waves, because we now also have Wi-Fi.

As always, please let us know your thoughts by using the comment feature of the blog. You can also visit our forums to get help and submit feature requests, you can find them here.