Perfect Forward Secrecy in SSH

Finnbarr P. Murphy

(fpm@fpmurphy.com)

Perfect Forward Secrecy (PFS) is a property of public-key encryption systems which generate random public keys per session for the purposes of key agreement which are not based on any sort of deterministic algorithm. A compromise of one message cannot lead to the compromise of another message or multiple messages. Twitter, Apache *mod_ssh*, SSL, TLS, and IPSec all support forward secrecy.

According to the referenced Wikipedia article:

> Forward secrecy is designed to prevent the compromise of a long-term secret key from affecting the confidentiality of past conversations. However, forward secrecy (including perfect forward secrecy) cannot defend against a successful cryptanalysis of the underlying ciphers being used, since a cryptanalysis consists of finding a way to decrypt an encrypted message without the key, and that forward secrecy only protect keys, not ciphers themselves.

Traditionally secure messaging systems such as secure email rely on protocols like GnuPG (the GNU implementation of PGP) for security. If you wish to receive encrypted mail, you generate a pubic-private key pair, advertise or otherwise disseminate the public key, and those wishing to send encrypted email to you encrypt their outgoing message with that public key. The problem with this approach is that every time anyone sends a message to you, their message is encrypted with your one static (immutable) public key.

If an attacker can record your encrypted traffic in the communications stream over some extended period of time, and can compromise your private key at any point in the future by whatever means, all of the previously recorded encrypted traffic can be decrypted. This is often called “wiretap then crack.”

One solution to this potential attack vector is to use what is called *ephemeral key exchange*. Instead of always encrypting messages using the same static public key, peers in a message exchange negotiate secrets through an ephemeral key exchange. An number of key exchange and agreement mechanisms support ephemeral key exchange. Diffie-Hellman (DH) is probably the most common and well known. It is used to provide perfect forward secrecy in SSH and Transport Layer Security (TLS) ephemeral modes that are referred to, depending on the particular cypher suite, as either *EDH* or *DHE*.

SSH uses SSL. To get PFS in SSL, you need to be able to generate a ephemeral key pair for the actual encryption. For authentication, you need another non-transient key pair at least on one side. This is what happens in SSL in a DHE cipher suite: the client and server use newly generated DH keys for the key exchange, but the server uses a permanent key pair such as RSA, DSA, ECDSA, etc. for signatures.

A cipher suite is a set of ciphers used in the privacy, authentication, and integrity of data passed between a server and client. There are lots of SSL-related cypher suites using by SSH, HTTPS and more. There are a number of websites that will display information on the SSL cipher suites your browser supports for securing HTTPS connections.

Here is the output for one such website when I used a Firefox v25.0 browser:

Two versions of SSH currently exist. SSH v1 makes use of several patented encryption algorithms (some of which have now expired) and is vulnerable to a security exploit that allows an attacker to insert data into the communications stream. In SSH v1, PFS is achieved by using a second ephemeral set of host keys. The session key is actually encrypted with these ephemeral keys, not directly with the permanent host key, and then the ephemeral keys are discarded periodically. The default is one hour. They could be discarded after every connection, but generating new keys is expensive. For more information, see the SSH v1 *KeyRegenerationInterval* parameter.

In SSH v2, PFS is achieved by using DH to set up the session keys. DH inherently provides PFS without needing a second set of host keys as was required in SSH v1. In fact it does it better than SSH v1, because SSH v2 destroys the information that would compromise the session key immediately after generating it, instead of some time later.

From RFC 4251:

> 9.3.7. Forward Secrecy
>
> It should be noted that the Diffie-Hellman key exchanges may provide perfect forward secrecy (PFS). PFS is essentially defined as the cryptographic property of a key-establishment protocol in which the compromise of a session key or long-term private key after a given session does not cause the compromise of any earlier session [ANSI-T1.523-2001]. SSH sessions resulting from a key exchange using the diffie-hellman methods described in the section Diffie-Hellman Key Exchange of [SSH-TRANS] (including "diffie-hellman-group1-sha1" and "diffie-hellman-group14-sha1") are secure even if private keying/authentication material is later revealed, but not if the session keys are revealed. So, given this definition of PFS, SSH does have PFS. However, this property is not commuted to any of the applications or protocols using SSH as a transport. The transport layer of SSH provides confidentiality for password authentication and other methods that rely on secret data.

With SSH v2, if you want PFS, you need to use ephemeral DH (AKA *DHE* or *EDH*), because that is what is defined and supported by existing implementations. Why not use ephemeral RSA in SSH v2? After all, it was supported in SSH v1? The reason is that producing a new DH key pair is extremely fast compared to RSA provided that some DH parameters are reused.

