Explaining perfect forward secrecy

December 2 2013, by Richard Mortier

How do you keep your private info under lock and key? Credit: IntelFreePress

Twitter has announced it is introducing perfect forward secrecy to help users protect their information from spies and cyber-criminals.

Even if we don't realise it, we all rely on cryptography when we use the web. It is at the heart of social networks, retail sites and any other sites that provide web addresses beginning with HTTPS, the secure HTTP protocol.

When you use HTTPS instead of HTTP, it invokes a set of protocols that encrypt communications between your browser and the server it's talking to so no eavesdropper can listen in. But a malicious attacker will still do their best to get at what you're saying. It is one class of attack like this that perfect forward secrecy attempts to block.

Chatting without PFS

Encryption schemes all rely on some secret information held by one or both parties to the communication. The basic operation of the HTTPS protocol is for the browser and server to exchange information so both can agree on a secret session key. This key is used to encrypt the rest of the communication session. The clever bit is that while all the information in the exchange is public, even if an attacker observes the entire exchange, they still cannot capture the secret your browser and the server agree on.

To agree on this session key browser and server use public key cryptography where a secret key used to encrypt communications is split into two parts, one public, the other private. Then, if one user, let's call him Bob, encrypts his data with another, Alice's, public key, only Alice's private key – and thus, if she's careful, only Alice – can decrypt it. Assuming Alice is running the web server and Bob is running a browser connecting to Alice's server, in traditional HTTPS Bob's browser would generate a random session key, encrypt it with Alice's public key and send it to Alice. Alice can then use her private key to decrypt this session key, and the session key becomes a shared secret that can be used to encrypt the rest of the session's communication between Alice and Bob.

If an attacker were able to capture Alice's private key – whether due to Alice's carelessness, legal demands requiring Alice to surrender her keys, or through more nefarious means – and they're also able to capture all communications with Alice's server, then the attacker would be able to

decrypt the key exchange part of these sessions. They could then extract the no-longer-secret session keys and read all of these communications between clients and Alice's server.

What makes PFS different

By applying PFS, a different set of cryptographic protocols replace the session key exchange process with one that never sends the secret session key across the network, even in an encrypted form. As a result, even if the attacker manages to get Alice's private key, they will not be able to recover the still-secret-session keys, and so they will not be able to decrypt any of the communications with Alice's server.

There is a cost to doing this: the cryptography used in PFS is slightly more complex than the traditional techniques so it does take more processing power. But it is not an insurmountable burden. In practice it will usually be negligible compared to all the other things that the server and browser will be doing at the same time.

The other problem arises if you run a farm of servers rather than a single server, as all modern large-scale web services must. Much as with human conversation, sessions between browser and server will often go idle but then start up again. To manage the load in their web-farm, a service provider will often wish to resume a session on a different server from that it originated on. To do this with PFS means sharing the session keys among all the servers in the web-farm. And of course, this has to be done without recording the secret session keys anywhere. Otherwise all that's been achieved is to change the file that the attacker needs to steal from the one containing the private key to the one containing the session keys.

Ultimately then, PFS should be a good thing for everyone, keeping your communications secure against another class of attack. Everyone, that is, who can make use of it – not all browsers, particularly older browsers,

support it. But that's just another good reason to upgrade.

This story is published courtesy of The Conversation *(under Creative Commons-Attribution/No derivatives).*

Source: The Conversation