A group of hackers known as THC (The Hacker’s Choice) last week [released an interesting DoS tool](http://www.thc.org/thc-ssl-dos/) that works at the SSL/TLS layer. The tool is exploiting the fact that, when a new SSL connection is being negotiated, the server will typically spend significantly more CPU resources than the client. Thus, if you are requesting many new SSL connections per second, you may end up using all of the server’s CPU.

The issue abused by the tool is not new, but what is new is that we now have a well publicised working exploit, and that always makes you pay attention. In addition, the tool uses the renegotiation feature, which means that it can force a server to perform many expensive cryptographic operations over a single TCP connection. It’s not clear if the relying on renegotiation helps with the DoS attack (there’s a [very good analysis of the trade-offs on Eric Rescorla’s blog](http://www.educatedguesswork.org/2011/10/ssltls_and_computational_dos.html)), however the fact that external DoS mitigation tools (e.g., rate limiting setups) are only seeing one TCP connection certainly helps with avoiding detection.

But that’s only if your server supports client-initiated renegotiation. If it does not, anyone wishing to perform a DoS attack against the SSL layer will have to fall back to using one TCP connection for one SSL connection. IIS, for example, does not support client-initiated renegotiation. Apache used to, but changed its behaviour when implementing RFC 5746 (which fixed the [TLS Authentication Gap problem](http://blog.ivanristic.com/2009/11/ssl-and-tls-authentication-gap-vulnerability-discovered.html)). Even if you depend on a product that does support client-initiated renegotiation, chances are you can easily disable that feature. And, when you do, you are not going to miss it (unlike server-initiated renegotiation, which some sites that require client certificates might need).

To help you with assessing your systems for this weakness, we have updated the [SSL Labs assessment tool](https://www.ssllabs.com/ssldb/) to test not only if secure renegotiation is supported (which we’ve been testing for some time now), but also to check if secure client-initiated renegotiation is enabled. Previously we only tested for insecure client-initiated renegotiation.

The sensible thing to do is to check for client-initiated renegotiation support in your servers, and disable it where possible. Although that won’t substantially help you overall (defending against DoS attacks is notoriously difficult and expensive), it will harden your defences against this particular technique.