If you’re presently volunteering to wear the hat of “SSL/TLS Vulnerability Manager”, then you’ve come to the realization that patching one vulnerability often raises others. In wake of revelations about mass surveillance, many of us moved quickly to implement Perfect Forward Secrecy, or PFS, since this method of encryption is strongly resistant to man-in-the-middle (MITM) attacks. Implementing PFS is as simple as identifying and prioritizing the ephemeral ciphers supported by whatever terminates TLS in our infrastructures. For those of us managing BIG-IP, that meant prioritizing Diffie-Hellman Ephemeral (DHE) ciphers.

Fast forward to last month, and the publication of the LogJam vulnerability. Suddenly, some PFS ciphers weren’t so resistant to those pesky MITM attacks. Particularly, any DHE cipher using so-called export-grade encryption. DHE key parameters of 512 or 768-bit lengths were proven to be vulnerable by academic researchers. The WeakDH team further estimates “that an academic team can break a 768-bit prime and that a nation-state can break a 1024-bit prime”. SSL Labs then moved quickly to cap the grade of sites negotiating DHE ciphers with key lengths of 1024 bits or less to a “B”. Since ZDnet has just published an article about how Chrome is now flagging some banks for weak crypto due to SHA1-signed certificates, a “B” grade from SSL Labs may not be something we can hang our hats on.

Since TMOS version 11.6, BIG-IP has prioritized ciphers as such: DHE>RSA>ECDHE. The reason for this ordering is that DHE and RSA are accelerated in BIG-IP crypto hardware, while ECDHE (the high performance Elliptic Curve variant of DHE) is handled by general-purpose CPU cycles. For a time, this priority order enabled BIG-IP administrators to achieve “A” grades right out of the box using the DEFAULT cipher string. However, since BIG-IP uses DHE 1024, that same DEFAULT string in TMOS v11.6 now achieves only a “B”. Here is an example of how to re-order and specify an explicit cipher string which omits the weak DHE ciphers and still maintains an A grade:

```
!SSLv2:!EXPORT:ECDHE+AES-GCM:ECDHE+AES:ECDHE+3DES:RSA+AES-GCM:RSA+AES:RSA+3DES:-MD5:-SSLv3:-RC4
```

Many of those same BIG-IP administrators have queried the DevCentral community or opened support cases requesting DHE 2048 or better support on BIG-IP. However, much like the shift from RSA 1024-bit key lengths to 2048-bit key lengths, the additional workload, even with chip-based acceleration is at least 5x greater for DHE 2048. Additionally, there are older clients such as Internet Explorer before IE11 an Java before version 8 that do not support DHE 2048. As such, it is no easy decision to implement support for DHE 2048 on BIG-IP.

Fortunately, all is not lost. Elliptic Curve DHE (ECDHE) performs very well even without hardware acceleration on BIG-IP, because Elliptic Curve Cryptography was designed, in part, to provide greater encryption strength with smaller key sizes. By prioritizing ECDHE in the cipher string, BIG-IP administrators can still easily achieve an “A” grade from SSL Labs.

In the event that we do need to preserve CPU cycles, and leave DHE 1024 as a higher priority, it’s important to understand how BIG-IP handles DH parameters by default. The most alarming weakness revealed by the LogJam research is the re-use of common primes for generating keys. A feature of BIG-IP mitigating use of weak or common DH parameters was recently noted by Ivan Ristic:

> @ziromr I think it makes sense to build DH parameter rotation in software so that users don’t need to worry about. Like F5 did.

— Ivan Ristic (@ivanristic)

June 17, 2015
By default, BIG-IP automatically rotates DH parameters to prevent exploitation related to parameter re-use and weak or common primes.

While we may still need to explain our “B” grade, knowing how and why BIG-IP handles DHE goes a long way toward offsetting concerns about the potential vulnerability to a LogJam-style MITM attack.