Array Networks Blog

03
NOV. 15

How Safe Is OpenSSL? Part II

POSTED IN UNCATEGORIZED BY PAUL ANDERSEN

0 COMMENT

In Part I of this mini-blog series, I briefly explored the frequency and severity of OpenSSL vulnerabilities, and presented an infographic/timeline of a number of vulnerabilities that posed a very high exploitability subscore (i.e. they were deemed much easier for a malefactor to exploit than other vulnerabilities).

As I stated in that post, this series is most emphatically not meant as 'OpenSSL bashing.' The majority of our business is network security, and that's ultimately the focus of this series.

Array does use OpenSSL in certain product functions, such as our XML RPC and SOAP APIs, as well as our Web-based user interface (WebUI). For our core product functionality and production traffic, however, we use our own proprietary SSL stack. As a network/IT manager, why should that matter to you?

- OpenSSL is used by most, if not all, of our esteemed competitors throughout their products –from WebUI to production traffic. In addition, it's used by a wide variety of other products in the networking industry, as well as around a half million Web servers globally.

That very ubiquity carries risk by making OpenSSL a more attractive target for the evil-doers out there. As one commenter wrote on a Schneier on Security blog on the Heartbleed vulnerability:

"Because it is open source and free to use, it is likely to create a mono culture [sic] and thus a single 'disease' can have catastrophic effects."

Our proprietary SSL stack has proven immune to OpenSSL vulnerabilities, such as Heartbleed, Bash and others. We've also 'walled off' production traffic from our product functions that do use OpenSSL –thus limiting exposure – and we use an older, time-tested OpenSSL version that predates the introduction of the Heartbleed code error. In addition:

- OpenSSL has had a reputation for being overly complex, bloated and stuffed with functions that are extraneous to the majority of tasks it is used for.

We chose to develop a proprietary SSL stack for a number of reasons; some of them are lost to the mists of time, but surely must include that the OpenSSL project began at just about the same time as Array was founded. In addition, we chose a different hardware path than others – leveraging Intel architecture rather than ASICs – which has given us advantages in economies of scale and time to market with new features and capabilities.

Above all else, though, was that a proprietary stack allows us to include only those functions that are required for the tasks that our products perform. It has allowed us to keep our code much more agile and flexible, and provides much higher performance through much lower overhead.

It should be noted as well that in the wake of Heartbleed, two separate development efforts forked off of OpenSSL to provide a cleaner, more secure and streamlined SSL/TLS implementation: BoringSSL, run by a team from Google, and LibreSSL, run by a team from OpenBSD , which is no relation to the OpenSSL team. (Note that the LibreSSL team pruned more than 90,000 lines of C code and 150,000 lines of content from its OpenSSL fork in its first few weeks of operation.)

These are great efforts, and we'll be watching them closely as they progress. This leads to one last insight regarding the OpenSSL effort:

- Until recently, the OpenSSL development effort had only one full-time developer and few monetary resources.

As Nicole Perlroth wrote in the New York Times Enterprise Computing blog, "When a crucial and ubiquitous piece of security code like OpenSSL ... can be accessed by all the world's programming muscle, but only has one full-time developer and generates less than $2,000 in donations a year, clearly something is amiss."

Heartbleed appears to have served as a wake-up call for the industry, and the Core Infrastructure Initiative stepped up to provide funding for full-time developers as well as a code audit. This is all great – no, fantastic – news for OpenSSL.

However, a complex code base like OpenSSL can't change overnight, and time will tell if the other initiatives will bring a more secure iteration. Meanwhile, by our choice many years ago to develop and maintain our own SSL stack, and through our careful implementation of safeguards, we've insulated our customers from the vast majority of OpenSSL vulnerabilities.

In Part III of this series, we'll discuss best practices, tips and techniques that can help you gain the important benefits of SSL while mitigating risk.