Ben Lister

Professor Otten

CS 110

21 November 2017

Connecting to the internet to send and receive information means connecting one's own device to a server, or chain of servers. Without requisite authorizations, such as on a public network, the packets of information sent to the server are susceptible to being seen by someone listening for WiFi signals - causing search history, emails, and even login credentials visible to those with malign intent. To foil such information theft, various security protocols were established for a more secure connection, and the packets themselves may be encrypted to anyone but the intended server recipient (Geier). OpenSSL is a library of tools for maintaining secure client-server connections for Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols (OpenSSL). Throughout its use as a cryptographic toolkit, OpenSSL has been found to have over a dozen security vulnerabilities, including the critical-level exploit CVE-2016-6304 that put a large portion of the internet at risk and may still be exploitable today (Khandelwal).

OpenSSL v0.9.8h was released May 28, 2008, and among many changes from the previous version, added the capability for a TLS certificate negotiation. Essentially, when a client attempts to connect to a server via HyperText Transfer Protocol (HTTP), the server returns a certificate using Online Certificate Status Protocol (OCSP), indicating the validity of the client's SSL certificate. The client could then use this information to determine why the server considered them unauthorized; an invaluable tool when there are multiple reasons why a connection attempt can fail. Unfortunately, this feature was implemented without a limit on the number of negotiation attempts, meaning that a remote attacker could endlessly fire off connection status requests, even with invalid SSL credentials, to overload the server's memory with processing OCSP certificates. This memory exhaustion would eventually cause valid clients to be denied service from the server, as each status request could use up to sixty-four thousand bytes of the server's memory. Servers using OCSP and supported with these versions of OpenSSL could potentially be shutdown (Khandelwal). The security breach could even be exploited on non-OCSP servers with default configuration via a TLS extension; the only way to prevent the denial-of-service attack in the versions of OpenSSL where the exploit is possible is to run a custom server build using the "no-ocsp" option (Lei).

The TLS certificate negotiation in v0.9.8h, implemented by core member and founder Steve Henson, was exploitable until v0.9.8l, where Ben Laurie disabled renegotiation entirely due to a severe problem entirely distinct from DoS attacks. In the next patch, Eric Rescorla, along with Henson and Laurie, re-enabled renegotiation, with the added requirement of the client's extension, and Henson added a timeout function to OCSP utilities. The exploit was possible again until v1.0.1.u, when Rob Stradling added timestamps to OCSP certificates, yet reenabled in v1.0.2. Ironically, David Ramos from Stanford reported the potential for failed renegotiation on the client's side to cause a NULL pointer to dereference on the server's side, resulting in a DoS attack. This exploit, CVE-2009-3555, was patched in v1.0.2 by Henson and Matt Caswell, though despite needing to debug what is likely the same section of code causing the recurring exploit, it was missed and continued to live until v1.0.2i when Henson changed renegotiation yet again. The exploit was re-enabled for a final time in v1.1.0, when Henson introduced an experimental change that involved the server asking for a certificate upon failed renegotiation attempt, before being formally discovered by security researcher Shi Lei and disabled permanently in v1.1.0a, presumably by adding a maximum client rate to the number of renegotiation attempts, and/or by reverting the experimental v1.1.0 change (OpenSSL).

Fortunately, there is reason to believe this exploit, CVE-2016-6304, was never actually used in a DoS attack. Attempting an attack as easy as repeatedly asking for renegotiations from a remote setting, especially with the majority of servers vulnerable, would certainly have been done if the weakness known (Lei). Furthermore, the exploit was possible for nearly eight years, excepting three patches during that time, before it was finally discovered and addressed in September 22, 2016. Were such an attack successful, the server's crash logs would have showed countless requests to renegotiate from the attacking client and OCSP status reports in the server's memory. The hypothetical crash logs would have been reported, investigated, and addressed. Thus, the actual damage inflicted by this particular exploit is low. While there is still a lingering possibility that a server running an outdated version of OpenSSL could fall victim to a DoS attack, the more pressing concern is how such a critical weakness with the potential to have caused so much damage could exist for almost eight years. All throughout this time, multiple experts updated and even fixed similar exploits in the relevant code section, before killing the exploit by adding a simple maximum to the client renegotiation rate. This weakness was never discovered by OpenSSL employees; rather, it took an outside party to see the vulnerability. Additionally, CVE-2016-6304 is only one of dozens of reported security vulnerabilities throughout OpenSSL's history, and isn't even the most severe – the infamous 'Heartbleed' exploit was actually used in attacks, and resulted in widespread data theft, not just server crashes (Heartbleed).

OpenSSL has violated several of the eight principles in the Software Engineering Code of Ethics and Professional Practice, the most egregious of which are failures of product and management. As the most popular SSL and TSL cryptographic library, they have a professional responsibility to ensure their software lacks bugs and security weaknesses. That so many are affected by mistakes in the code, and could suffer severe damage in data or privacy loss, should drive OpenSSL's standard even higher than most developers. They should also have some culpability to mistakes, as the success of countless other businesses rely on secure internet connections. As it stands, dozens upon dozens of failures have been discovered in the OpenSSL library, failures resulting from security vulnerabilities or even faulty logic and poor program design. This is not in accordance with principle 1.03 of the Software Engineering Code of Ethics, which claims that software engineers "approve software only if they have a well-founded belief that it is safe, meets specifications, passes appropriate tests, and does not diminish quality of life, diminish privacy or harm the environment" (Gotterbarn). Given their enormous publicly-available database of previous security vulnerabilities, it is difficult to say the developers could have a well-founded belief that any of their code was correct. Yet the developers themselves were all undoubtedly talented, hard-working, and committed to the success of their product. A lack of funding, perhaps?

Alongside a violation of the principle of the highest-standard product, OpenSSL has violated the principle of management and maintenance in software development. It is impossible for a small team of developers to meet the demands of an open source TSL SSL library. Throughout OpenSSL v0.9.8h to v1.1.0, only four unique people ever worked on the renegotiation section of the code, leading to CVE-2016-630, with Steve Henson responsible for the lion's share (OpenSSL). From a management perspective, it is gravely irresponsible and unfair to place one person, who may not have express ethical training and education, in a position to negatively impact so many people if a coding mistake is made. This is in violation of principle 5.01, which states software management must "ensure good management for any project on which they work, including effective procedures for promotion of quality and reduction of risk", and principle 5.05, which states software management must "ensure realistic quantitative estimates of cost, scheduling, personnel, quality and outcomes on any project on which they work or propose to work, and provide an uncertainty assessment of these estimates"(Gotterbarn). The team at OpenSSL should have realized they needed more developers to professionally continue – which they likely did at some point, but could not afford to.

As an open source, free-to-use software, funding was a major issue until Heartbleed-spurned major tech companies pledged to support open source cryptographic libraries (with OpenSSL being the main recipient) in 2014. Linux Foundation Director Jim Zemmling even said "we should have done this three years ago" and that “the folks at OpenSSL are guys who have dedicated most of their adult careers to super hard software development that is, I would argue, in some ways thankless work” (Brodkin). Nevertheless, it took another eighteen months to fix the maximum OCSP renegotiation rate of CVE-2016-6304, even with all that money. As such, it is possible, though perhaps unlikely, that a simple lack of funds is insufficient to explain OpenSSL's violation of the software engineering principles of product and management.

Works Cited

Brodkin, Jon. "Tech Giants, Chastened by Heartbleed, Finally Agree to Fund OpenSSL." *Ars Technica*, 24 Apr. 2014.

Geier, Eric. "Here's What an Eavesdropper Sees When You Use an Unsecured Wi-Fi Hotspot." *PCWorld*, 28 June 2013.

Gotterbarn, Donald. "Software Engineering Code of Ethics and Professional Practice." *ACM Ethics,* WordPress. Accessed 22 Nov. 2017.

Khandelwal, Swati. "Critical DoS Flaw Found in OpenSSL." *The Hacker News*, 23 Sept. 2016. *The Hacker News*. Accessed 21 Nov. 2017.

Lei, Shi. Weblog post. *360 Security*, Qihoo, 22 Sept. 2016. Accessed 22 Nov. 2017.

OpenSSL. *OpenSSL*, OpenSSL Software Foundation. Accessed 22 Nov. 2017.

"The Heartbleed Bug." *Heartbleed*, Synopsys. Accessed 22 Nov. 2017.