Team Members:

Richard Joy

Proposed Project Title:

A comparison of popular open source libraries implementing the SSL/TLS protocols with particular focus on compatibility, security, and performance.

Introduction and Motivation

Secure Sockets Layer (SSL) and, more recently, its successor Transport Layer Security (TLS) are foundational technologies of secure communications on the Internet. Secure web browsing, Virtual Private Networks (VPN), and other secure protocols (such as SFTP and SSH) leverage TLS to ensure the privacy of their communications. To meet the needs of software developers, many libraries have been created to perform standard TLS functions. Some of these libraries use proprietary code that is difficult to examine to verify security and functionality. However, some of the most widely used SSL/TLS libraries are open source, and are therefore easier to examine and verify.

When creating an application that uses TLS, or in some cases simply configuring one, a developer must make choices about which library to leverage for TLS functionality. Of course, it would be possible to create a custom library, but this approach is impractical because it takes significant time and produces a product that has not been verified. Instead, many developers choose to use one of the open source libraries. With this in mind, it would be valuable to perform a comparison of popular open source libraries that would be usable by a developer attempting to choose between them.

Focus of Evaluation

OpenSSL, NSS, and GnuTLS are very popular SSL/TLS library choices for developers of secure web technologies. Because of this, they will be examined mainly in comparison to each other. As mobile applications increase in popularity, libraries that can run on mobile platforms are a necessity. JSSE is a java implementation of SSL/TLS that can be used by applications developed for Android, and will therefore be included in this examination. Finally, MatrixSSL is an implementation that is designed to run on platforms with extremely limited resources. As embedded applications increase in popularity, such an approach should have significant value. Therefore, MatrixSSL will also be examined.

Evaluation Criteria, Questions, and Hypotheses

The various implementations will be evaluated with a focus on compatibility, security, and performance. Specifically:

Compatibility

- On what operating systems and hardware is the library supported?
- What, if any, major applications use the library?

Security

- How well does the library conform to published TLS standards?
- Has the library been formally accredited (e.g. by NIST or another accrediting agency)?
- What security vulnerabilities have been found to affect the library?

- How quickly do the library's developers respond to security vulnerability discoveries?

Performance

- What resources are required to run the library (storage on disc, CPU specifications, memory usage, etc.)?
- How quickly does the library perform standard encryption and decryption tasks that occur in TLS communications?

Because it is the most popular open source SSL/TLS library, OpenSSL should be expected to excel against these metrics. However, based on its design philosophy, MatrixSSL may prove to be the better choice from a performance perspective.

Evaluation Methods

Compatibility an Security will be evaluated mainly using publicly available documentation for each of the libraries. The RFC's for TLS will be considered the authoritative sources for evaluating if a particular library conforms to the TLS standard or not. Performance will be evaluated using a combination of the library developer's published minimum system requirements (if available) and a set of performance tests performed against a set of test data on a reference system.

Reference Systems

Two reference systems will be used for performance testing. These two machines are intended to mimic development systems that an open-source developer might use to test applications.

System 1

Linux Virtual Machine

Hardware:

2 vCPUs
2GB RAM
20GB Hard Disk

OS:

Ubuntu 14.04.03 32-bit server

System 2

Linux Virtual Machine

Hardware:

2 vCPUs
2GB RAM
20GB Hard Disk

OS:

Ubuntu 14.04.03 64-bit server

Evaluation metrics

Asymmetric encryption and decryption – Measure the average time needed to encrypt and decrypt data using asymmetric keys

Variables to manipulate:

Key size

Symmetric encryption and decryption – Measure the average time needed to encrypt and decrypt data using symmetric keys

Variables to manipulate:

Key size
Algorithm

Existing Research

Some research has been done to compare SSL libraries. Timo Bingmann produced a comparison of several popular SSL libraries[10] and even Wikipedia contains information on the subject[1]. However, Bingmann's comparison only looked at symmetric encryption performance, and made no attempt to quantify the degree to which each library conformed to the TLS standards. Also, while he performed tests on OpenSSL as part of his efforts, the other libraries in this project were not evaluated. The Wikipedia article on the subject does not attempt to provide performance comparisons, and it is also worthwhile to remember that Wikipedia is a source of dubious reliability.

Timeline

Oct. 26-27 – Identification of major research sources for compatibility and security criteria. Initial setup of performance benchmarking reference systems. Initial compatibility comparison research completed.
Nov. 9-10 – performance benchmarking tools identified/created. Initial security benchmarks collected. Licensing comparison completed
Nov. 23-24 – Initial performance benchmarks collected
Dec. 7-8 – Final performance and security benchmarks collected and analysed.

Potential Specification Changes

If time permits, it may be possible to add one or more additional libraries to the comparison set.
If time permits, it may be possible to evaluate performance of the libraries on additional reference systems.

Tentative Table of Contents

Literature

[1] “Comparison of TLS implementations,” *Wikipedia, the free encyclopedia*. 11-Oct-2015.

[2] R. Barnes, M. Thomson, A. Pironti, and A. Langley, “Deprecating Secure Sockets Layer Version 3.0.” [Online]. Available: https://tools.ietf.org/html/rfc7568. [Accessed: 11-Oct-2015].

[3] “gnutls.org.” [Online]. Available: http://gnutls.org/. [Accessed: 03-Oct-2015].

[4] “JSSE Reference Guide.” [Online]. Available: https://docs.oracle.com/javase/8/docs/technotes/guides/security/jsse/JSSERefGuide.html. [Accessed: 03-Oct-2015].

[5] “MatrixSSL - Open Source Embedded SSL and TLS.” [Online]. Available: http://www.matrixssl.org/. [Accessed: 03-Oct-2015].

[6] “OpenSSL.” [Online]. Available: http://www.openssl.org/. [Accessed: 03-Oct-2015].

[7] “Overview of NSS,” *Mozilla Developer Network*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Mozilla/Projects/NSS/Overview. [Accessed: 03-Oct-2015].

[8] S. Turner and T. Polk, “Prohibiting Secure Sockets Layer (SSL) Version 2.0.” [Online]. Available: https://tools.ietf.org/html/rfc6176. [Accessed: 03-Oct-2015].

[9] “Security/Server Side TLS - MozillaWiki.” [Online]. Available: https://wiki.mozilla.org/Security/Server_Side_TLS. [Accessed: 13-Oct-2015].

[10] T. Bingmann, “Speedtest and Comparsion of Open-Source Cryptography Libraries and Compiler Flags.” [Online]. Available: https://panthema.net/2008/0714-cryptography-speedtest-comparison/. [Accessed: 14-Oct-2015].

[11] A. Freier, P. Karlton, and P. Kocher, “The Secure Sockets Layer (SSL) Protocol Version 3.0.” [Online]. Available: https://tools.ietf.org/html/rfc6101. [Accessed: 03-Oct-2015].

[12] T. Dierks and C. Allan, “The TLS Protocol Version 1.0.” [Online]. Available: https://tools.ietf.org/html/rfc2246. [Accessed: 03-Oct-2015].

[13] T. Dierks and E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.1.” [Online]. Available: https://tools.ietf.org/html/rfc4346. [Accessed: 03-Oct-2015].

[14] T. Dierks and E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.2.” [Online]. Available: https://tools.ietf.org/html/rfc5246. [Accessed: 03-Oct-2015].

[15] “Validated 140-1 and 140-2 Cryptographic Modules.” [Online]. Available: http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140val-all.htm. [Accessed: 03-Oct-2015].