A High-Level SSL API for Java Applications

Summary

This project was inspired by a recent academic paper titled, “The Most Dangerous Code in the World: Validating SSL Certificates in Non-Browser Software” [1]. The paper presents an in-depth study of SSL implementations in non-browser software and asserts that most non-browser software using SSL is doing so insecurely due to not understanding how to properly configure the SSL libraries. The goal of my project is to design an SSL API in Java to wrap the existing Java SSL APIs and incorporate the recommendations presented in [1]. The result should make it very easy for an application developer to accurately configure the underlying SSL library in order to secure the application’s network traffic, preventing misuse arising out of complicated or inconsistent interfaces for low-level options and error handling.

Description of the Problem

This project was inspired by a recent academic paper titled, “The Most Dangerous Code in the World: Validating SSL Certificates in Non-Browser Software” [1]. The paper presents an in-depth study of SSL implementations in non-browser software and asserts the following conclusions: many SSL libraries are unsafe by default, requiring higher-level software to correctly set their options, provide hostname verification options, and interpret return values; even safe-by-default libraries are misused by application developers who misinterpret their various options; SSL bugs are often hidden in middleware, above the actual SSL implementation but below the application layer, making the problem hard to locate and repair; some developers deliberately disable certificate validation, probably due to the hassle of setting up a working environment and unit tests, assuring their customers that the application supports SSL but not informing them that protection against active attacks has been turned off; and finally, that developers do not perform adversarial testing on their software, even for critical applications such as banking applications and e-commerce SDKs

The authors of [1] recommend for SSL library developers to make the libraries more explicit about their semantics, in order to make it easier for application developers to choose settings correctly. The authors also recommend for SSL library developers to rephrase questions to users and application developers to be more relevant - for example, instead of “verify hostname?” the SSL libraries should ask “Anyone can impersonate the server. Ok or not?”

Solution

The goal of my project was to design an SSL API in Java to wrap the existing Java SSL APIs and incorporate the recommendations presented in [1]. The result makes it very easy for an application developer to correctly configure the underlying SSL library in order to secure the application's network traffic, preventing misuse arising out of complicated or inconsistent interfaces for low-level options and error handling.

The solution was written in Java and still needs to be ported to other languages.

My poster will include important points from [1] as well as the design of my API and example code.

Results/Impact/Relevance

The high-level SSL API is being used in Mt Wilson to SSL connections between the clients and the server and between the server and monitored hosts. It has been open sourced as part of the Open Attestation project. If you are writing a Java application and using SSL, you need to look at your SSL code and if your boilerplate is disabling SSL security features you need to replace it with this API.

References

[1] M. Georgiev et al. "The Most Dangerous Code in the World: Validating SSL Certificates in Non-Browser Software". ACM Conference on Computer and Communications Security, 2012.