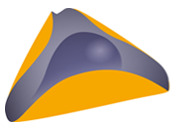
Tomcat Native 2

Final Report

Jocelyn Thode and Simon Brulhart
jocelyn.thode@unifr.ch
simon.brulhart@unifr.ch

21st June 2016

Tomcat Native 2 is a project mandated by Red Hat. It aims at providing a JNI wrapper for OpenSSL for use in both Undertow and Tomcat. This project is the practical part of the R&D Workshop course given at the University of Neuchâtel as part of the Joint Master in Computer Science.



MASTER IN
COMPUTER
SCIENCE

redhat.

Contents

List of Figures

1 Project description

1.1 Project context

For this project we are working with Red Hat which is a multinational software company that provides open-source software and technical support to enterprises[**redhat**]. At the moment, Tomcat[**tomcat**] and Undertow[**undertow**] are among the most used Java web servers. Both rely on TLS/SSL to encrypt and secure communications between a client and a server. To achieve good performance Tomcat Native[**tomcat-native**] provides access to OpenSSL in Tomcat. It uses Apache Portable Runtime (APR) to do so. Unfortunately, Undertow isn't compatible with Tomcat Native at the moment. Until last year, the APR would manage the encrypted sockets by itself. However this implied a lot of C code to maintain. To ease maintenance, some work has been done to use OpenSSL APIs directly instead. This works well enough despite some acceptable performance penalty. However the APR still provides the bindings to these APIs. We now want to strip the current solution of the APR and make the result compatible with Undertow.

Figure 4 shows an overview of the different ways to use TLS/SSL in Tomcat. This figure describes the path taken by a piece of data, going from Tomcat/Undertow to the OS network layer. There is a different color for each way to use TLS/SSL. We describe the architecture in more detail in Section 4.

1.2 Goals and objectives of the project

Our project aims to refactor Tomcat Native to drop APR and only rely on our own JNI[1] wrapper to interface with OpenSSL. This should make Tomcat Native much more maintainable while keeping great performance when using TLS/SSL. The project also aims to make the resulting Tomcat Native 2 usable in Undertow while keeping the compatibility with Tomcat. This means we need to :

1. Study the source code of Tomcat Native, Undertow and some experiments already done to use OpenSSL in Undertow.
2. Remove all APR code present in Tomcat Native
3. Abstract Tomcat Native to be used in Tomcat and Undertow
4. Provide good documentation so that the open-source community can use the project
5. Run benchmarks to test the implementation on multiple platforms
6. Merge OpenSSL 1.1 in Tomcat Native 2
7. Extend the OpenSSL support by adding JNIs for more features

Figure 5 shows what is expected at the end of the project.

[1] Java Native Interface

2 Project organization

This project requires us to read some documentation and source code. We also need to write reports and presentations. To keep the code base clean, we decided to adopt the same strategy as Numa de Montmollin last year and split our work in two different repositories on Github.

tomcat-native2-doc is the repository documenting the progress of our project. We put every report and presentation in the repo. This repo also contains a wiki which hosts the logbook, work plan and our different articles. The articles are useful to share our findings and more importantly to not forget about them.

tomcat-native2 is the repository dedicated to the code. This repo includes our implementation, and the documentation directly concerning it. For example, it should contain a ReadMe describing the building process and the main purpose of the project.

3 Realization

The project was realized as follows: Tomcat Native 2 is built as a common code base between Undertow and Tomcat. It should be optionally compiled on the side and loaded at run-time if the user needs to use the OpenSSL engine.

For this purpose, we needed to move some Tomcat code related to Tomcat-Native into our repo. Unfortunately the entire Java code could not be the same in Undertow and Tomcat. The reasons are explained in the section 6.

The C code used in Tomcat Native 2 does not rely on APR anymore. It only works under POSIX compatible systems for now. Tomcat Native 2 is working with OpenSSL up to 1.1. To use Tomcat Native 2 one needs to build the native and java parts and then include the generated jar as a dependency on both Tomcat and Undertow.

In Undertow, a sample example has been provided to showcase how to use Tomcat Native 2 together with the web server.

In Tomcat, simply using the NIO.2 or NIO connector should work when explicitly using Tomcat Native 2.

3.1 Build

3.1.1 Tomcat Native 2

1. `ant jar`
2. `cd native`
3. `./buildconf && ./configure && make`

   (`make` might return an error because it tries to install the libraries after building them, but this is not important)

3.1.2 Tomcat

1. `TCN2=path/to/tomcat-native-repository`
2. `git clone https://github.com/jocelynthode/tomcat.git && cd tomcat`
3. `git checkout undercat_trunk`
4. `echo "tcn2.jar=$TCN2/dist/tomcat-native-1.2.8.jar" >> build.properties`
5. `ant`

3.1.3 Undertow

1. `TCN2=path/to/tomcat-native-repository`
2. `git clone https://github.com/jocelynthode/undertow.git && cd undertow`
3. `git checkout undercat_trunk`
4. `./deps.sh "$TCN2"`
5. `mvn package -DskipTests -Dcheckstyle.skip`

The build process is also available in more details on the ReadMe of the Tomcat Native 2 repository, as well as specific instructions to run the obtained builds.

4 Architecture

4.1 State Of The Art

Currently there are 3 ways to use SSL/TLS in Tomcat, as shown on Figure 4. The first one uses the OpenSSL through the APR connector (blue path). Here the APR handles encryption and networking. It provides its own socket API and transparently encrypts and decrypts data flowing through it. Since the APR is programmed in C, Tomcat-Native offers JNIs to be used in Tomcat.

The second way (brown and black path) uses an NIO/NIO.2 connector with the JSSE engine, included in Java standard library. This has the benefit of using no external dependency. However the JSSE engine is not native code, which makes it the slowest option for SSL/TLS. Note that JSSE also provides a networking API, but Tomcat uses NIO/NIO.2 instead.

The third way (red and black path) also uses an NIO/NIO.2 connector, but it uses OpenSSL instead of JSSE for encryption. For this OpenSSL engine, Tomcat provides Java classes implementing the JSSE API. Then these classes call OpenSSL functions through the JNIs offered in Tomcat Native. However, the JNIs still use some features of the APR behind the scene, such as portable memory management. As explained in previous sections, this way was implemented last year with the goal of replacing the APR connector.

Concerning encryption, Undertow follows a similar architecture as the second way (NIO/NIO2 with JSSE engine).

4.2 Tomcat Native 2

Figure 5 describes the architecture of Tomcat Native 2 after our changes. We got rid of the APR connector and don't use the APR in JNIs anymore. Most of the Java code concerning the OpenSSL engine was moved from Tomcat to Tomcat Native 2 so that Undertow can use it. However Tomcat and Undertow use a different system to provide and instantiate the OpenSSL engine. These differences are explained in section 6. To fix this issue, we kept slightly different versions of some classes in both projects.

5 Benchmarks

This section describes the benchmarks that were run to ensure Tomcat Native 2 meets our performance expectations.

5.1 Tomcat

Jean-Frederic Clere ran a benchmark that consists in requesting a file of fixed size, 100,000 times and from 160 concurrent clients. He repeated this test with files of different sizes, and measured the average throughput and CPU usage of Tomcat for each test.

The cluster running this benchmark includes 4 machines generating the load (HTTP requests) and one test machine running Tomcat.

Here are the specifications for the test machine:

Model:	IBM System x3650
CPU:	Intel Xeon 2.50 GHz
CPU Cores:	2x4 (8 total)
CPU Cache:	6144 KB
Memory (RAM):	20 GiB

The machines are connected together via a 10 Gbit/s switch (*Cisco Nexus 5000*).

5.2 Undertow

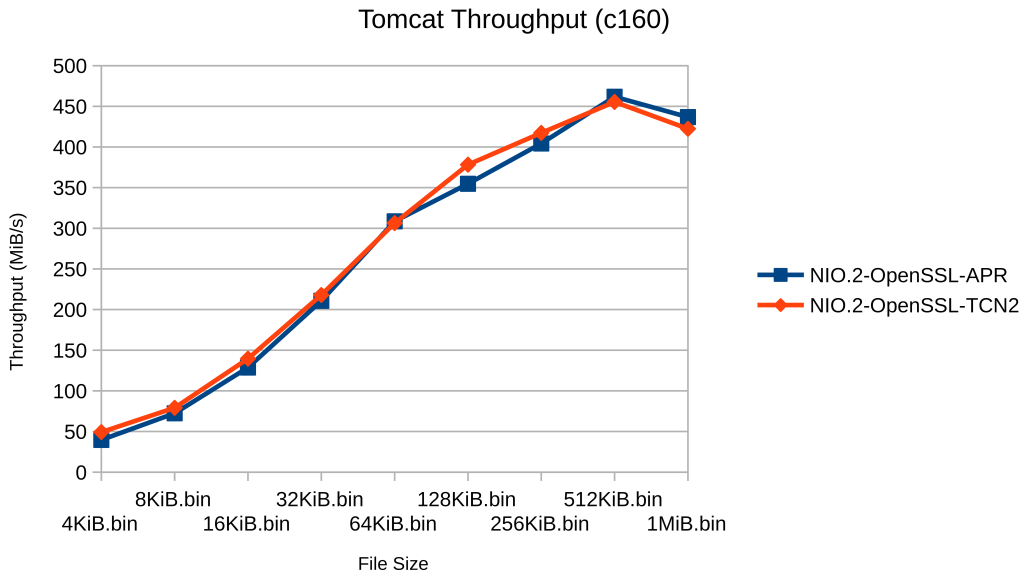


Figure 1: **Diagram showing the difference in throughput between Tomcat with Tomcat Native 2 and Tomcat with Tomcat Native.**

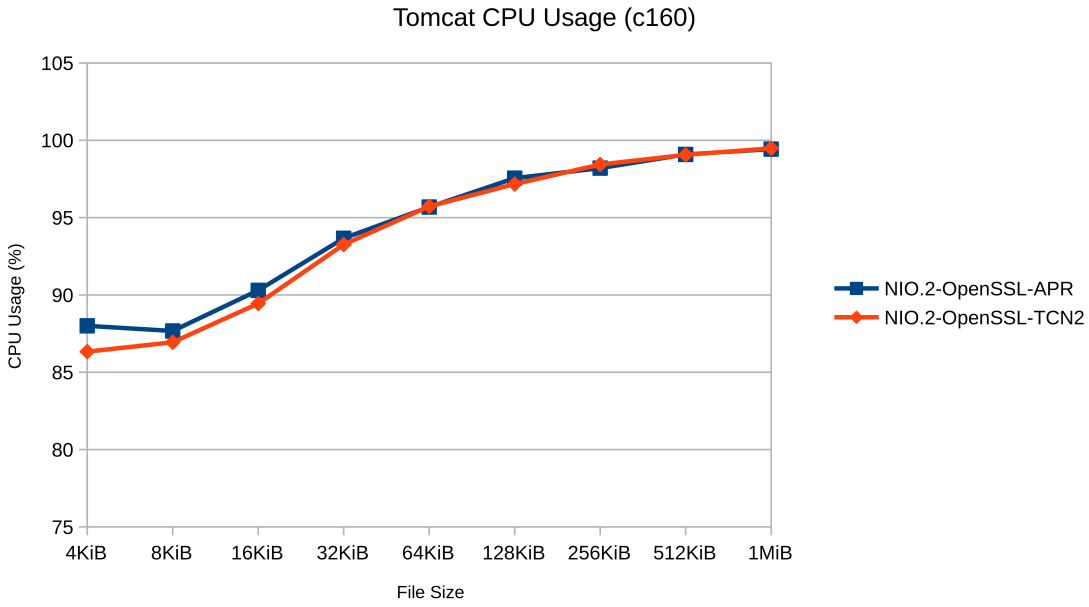


Figure 2: **Diagram showing the difference in CPU usage between Tomcat with Tomcat Native 2 and Tomcat with Tomcat Native.**

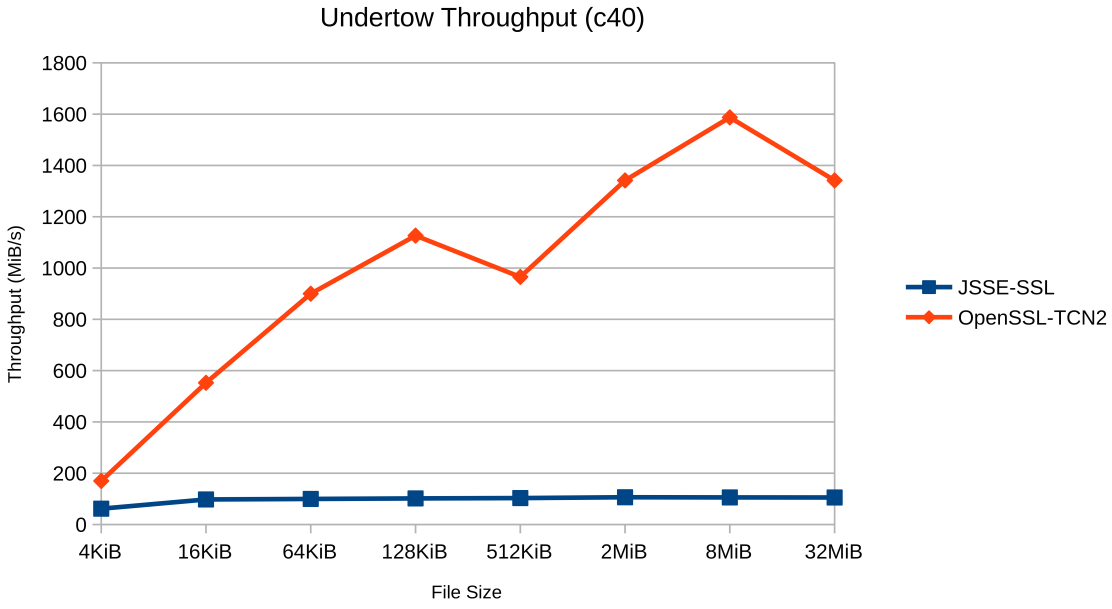


Figure 3: **Diagram showing the difference in throughput between Undertow with Tomcat Native 2 and Undertow with the JSSE SSL Engine.**

6 Issues

During this project, we face two issues described in the following subsections.

6.1 Diverging Philosophies

When we made Tomcat Native 2 Compatible we wanted to have everything in common in the Java code between Undertow and Tomcat. Unfortunately this was not possible due to diverging opinions between the Tomcat and Undertow maintainers. This concerned the way the OpenSSL engine is provided and instantiated in both project. Undertow maintainers want to use the SPI[2], as suggested by JSSE. Tomcat already uses a custom system, which maintainers think is more practical for their usage.

Fortunately, this covered much less code than we imagined although it forced us to rethink some parts of our architecture, in particular how all components would connect with each other.

6.2 Dynamic Loading

Some experiments regarding dynamic loading were performed in the `dynload` branch of our Tomcat Native 2 repository. This would enable Tomcat Native 2 to swap OpenSSL for other TLS/SSL libraries such as LibreSSL or BoringSSL without having to recompile.

Unfortunately developing a functional prototype with dynamic loading was taking too long, to the detriment of the rest of the project. We therefore made the decision to stop working on this feature.

7 Future Works

We completed all the obligatory milestones of the projects. However, some optional tasks were either not started or not finished.

The current code is only a prototype. In the future, tests should be put in place to ensure the code is working as expected in every situations. Tomcat Native 2 should also have a streamlined build process. This should be done when it is eventually merged in Tomcat and Undertow.

7.1 Better integration

Some improvements should be made regarding the integration of Tomcat Native 2 in Tomcat and Undertow, in addition to a better build process.

For example, in Undertow we handle ALPN[3] such that HTTP/2 is preferred, even when the underlying server does not support it.

Moreover, we want Tomcat Native 2 to be an optional run-time dependency. However Tomcat currently requires it both at compilation and run-time. Similarly we needed a

[2]Security Provider Interfaces
[3]Application-Layer Protocol Negotiation

few classes both in Tomcat and Tomcat Native 2, so we duplicated them in both projects. We did this to avoid a circular dependency between both projects, but there is a better way.

7.2 Dynamic Loading

We unfortunately could not find the time to finish dynamic loading as this proved to be more complicated than we anticipated as stated in section 6.

In the future adding Dynamic Loading to Tomcat Native 2 could be a boon for the project.

7.3 SSL/TLS Integration

At the moment Tomcat Native 2 does not support the use of sessions. This was not needed to have a working prototype, especially since sessions were not well supported in Tomcat to begin with. However, support of sessions is required for a complete SSL library.

Support for OpenSSL handshake callback should be implemented down the line. Currently we rely on a hack to make it work.

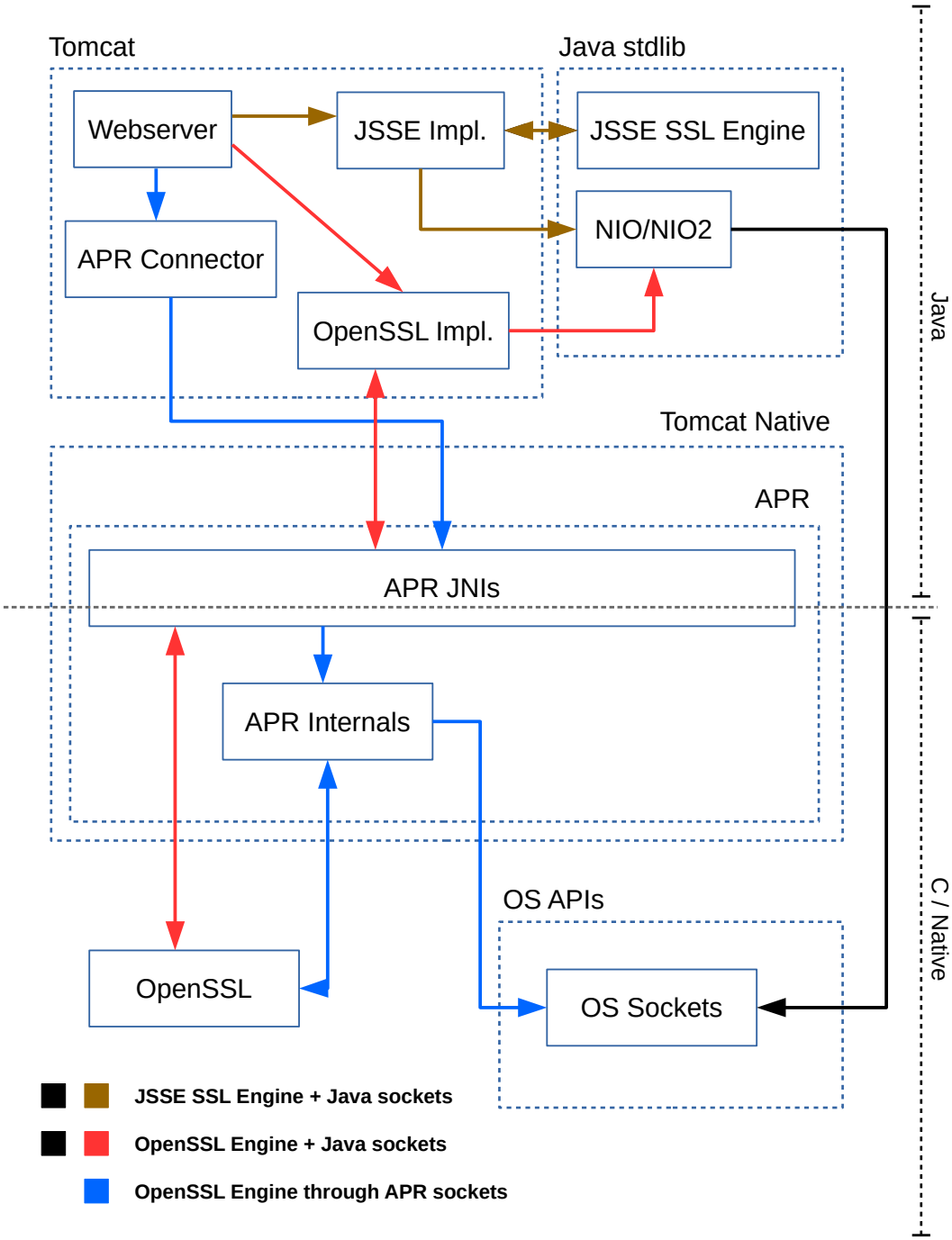


Figure 4: **Diagram showing the different ways to use TLS/SSL in Tomcat.**

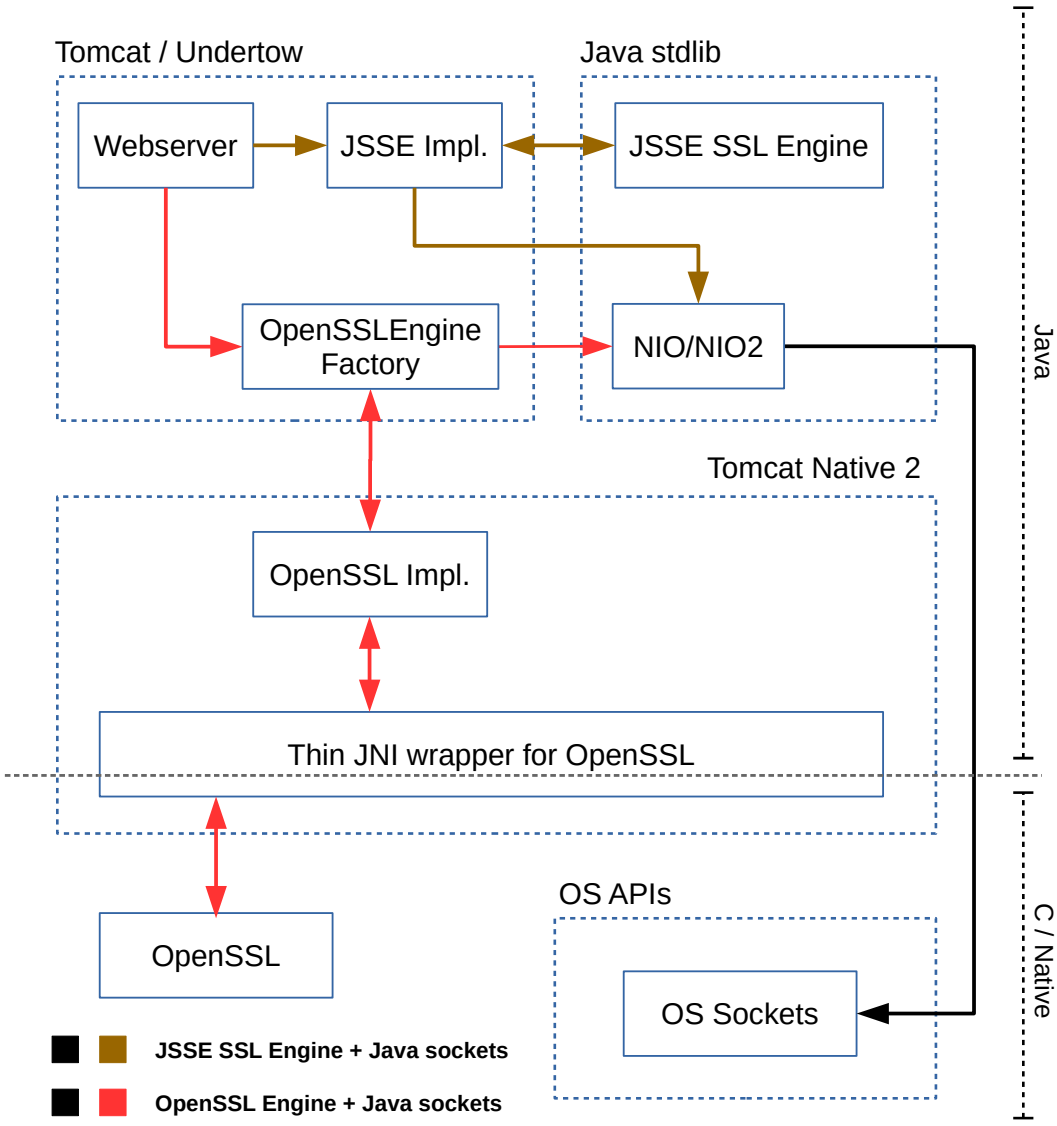


Figure 5: **Diagram showing the end result of our project.**