Table of Contents

Analysis of move to OpenSAML2 library in UNICORE

Introduction

OpenSAML 2 is a big library, which covers handling of SAML assertions and SAML protocols along with support for several SAML profiles. Internally OpenSAML 2 is a set of three closely coupled libraries (build by the same development team):

- opensaml Core SAML functionality: allows to produce SAML objects using API, to marshal them to DOM objects and to unmarshall SAML objects from XML DOM objects.
- java-openws Simple web service (SOAP) stack including WS-Security support.
- xml-tooling XML parsing along with support for XML signatures and XML encryption.

OpenSAML libraries altogether have 172.5 KLOC.

In UNICORE a very small samly2 library is used to provide SAML support. samly2 provides support for producing SAML objects using API, to convert them to and from XMLBeans objects and DOM objects. Also support for XML digital signing is provided. samly2 does not provide any web service support and uses XMLBeans library to manipulate XML easily.

samly2 has 2.8 KLOC.

Even looking at bare number of code lines number (OpenSAML is over 60 times bigger then samly2) we can be sure that differences are big. First of all web service stack in samly2 is not needed as it is provided by other specialized libraries in UNICORE. The 2nd difference is the XMLBeans adoption. It allowed to mostly eliminate all the need for a separate solution which is implemented by xml-tooling. Finally samly2 implements only those parts of SAML specification that are anyhow interesting from the UNICORE PoV, not everything that was defined.

What was analysed

The following actions were undertaken to verify if the exchange of samly2 with OpenSAML makes sense. The order presented here is not the same in which the actions were undertaken, it was modified to provide a more clear evaluation below.

1. OpenSAML library and API was initially evaluated.
2. One typical sequence of operations which is implemented in samly2 was implemented using OpenSAML2 and the results were compared.
3. Tricks and hacks which are programmed in samly2 to work around problems which were observed in various UNICORE components (e.g. due to buggy 3rd party libraries or bugs in a legacy code) were tried to be applied in OpenSAML 2.
4. Dependencies of OpenSAML2 were compared with dependencies which are used in UNICORE.
5. Performance of both solutions was compared, though in a limited way.
6. Effort to maintain samly2 was assessed by checking all commits during the last 12 months and this was compared with the expected effort to integrate and maintain implementation based on OpenSAML2.

Results of evaluation

Generally p. (1) showed (as expected) that **coverage of OpenSAML is complete** and all required SAML functionalities are there. API turned out to be verbose and rather low level. Regular user documentation is mediocre: there are few pages with incomplete examples. JavaDoc documentation is quite good. Generally finding out a possible way to program something with OpenSAML2 is quite easy. Finding *the best way* is more tricky as one must dig through the source code. For an experienced developer it is acceptable.

Test Implementation done in point (2) involved creation of an assertion with an API, setting issuer and subject, adding one attribute, finally signing the assertion. Here code without the last action (signature creation) is compared (it will be explained below why).

OpenSAML implementation required ca 25 lines of code to realize the task. The code which implements it in samly2 is 34 lines. **NOTE**: this compares usage of OpenSAML library with **implementation inside** samly2. To realize this task **using samly2** one needs 5 lines of code.

The next task was to check whether certain **workarounds which can be found in samly2 can be applied in OpenSAML2** based implementation. There is not much of them and only one of those looks problematic. It is related to namespace declaration in XML signature inserted into SAML assertion. It looks that to work around it the OpenSAML digital signature support would have to be ignored, and samly2 code which uses xmlsec API directly (as OpenSAML under the covers) just left as is. This is why in point one comparison of digital signature was not performed.

Another option is to ignore the samly2 workaround (what would cause the older UNICORE Gateways incompatible with new stack). Then OpenSAML2 code to generate signature is nearly 30 lines long. The similar code inside samly2 is around 50 lines. Usage of samly2 to sign the assertion: 1 line.

Now it is possible to summarize line comparison of the test implementation (rounded in any doubtful case in favour of OpenSAML2):

-	Implementation **inside samly2**	Implementation **using samly2**	Implementation **using OpenSAML 2**
Lines of code	84	6	52
Percentage	161%	11%	100%

Dependency analysis showed three potentially severe problems: OpenSAML2 uses Spring Framework core library, version 2.0. In UNICORE a version 3.0 is used. Next problem is that UNICORE is using wss4j library (providing WS-Security support). The latest version of this library still uses OpenSAML 1 library. It is very hard to verify if at runtime some errors can occur, but unfortunately it is possible: OpenSAML 2 uses in many cases the same namespaces (Java package names) as its predecessor. According to OpenSAML web page the versions are fully incompatible. Finally probably the most important issue was observed by accident: OpenSAML 2 uses DOM level 3 API. This is unsupported by XMLBeans library which forms a fundamental base of the whole UNICORE software. It is possible that some very difficult to overcome problems may come up during integration phase.

Performance of the two implementations was compared only briefly. Two implementations doing the same sequence of operations in OpenSAML and samly2 were compared: create assertion, add issuer, subject, one attribute without a value, convert to DOM element. Also another test was run were the code was additionally signing the assertions. 100 of warm up iterations for both libraries were run. The tests evaluated 10000 repetitions of the task (of course on the same decent system, JDK etc.). Several runs proofed that results presented below are stable:

Test	samly2	OpenSAML
without dsig	2.4s	3.8s
with dsig	21.4s	24.1s

We can see that samly2 outperforms OpenSAML, however the difference is small. In fact creation of a digital signature is dominating and as both solutions use the same xmlsecurity library underneath the results make sense. Additionally one more observation was made. OpenSAML library required 1.7s to initialize on the test machine which is a quite powerful workstation (3.17GHz, IC2D). It can be expected that on a less powerful notebook the startup can consume more then 2s. This is not relevant for the server side however for command line utilities the penalty is an issue.

Effort to maintain the samly2 library in the year 2010 was measured by evaluation of SVN commits. There were altogether 17 of them. Out of them 8 were actual bugfixes or improvements; the rest contained only semi-automatic changes (like update of version for a release, change of an updated dependency) or fixes of typos in JavaDocs. The real 8 changes were small (few lines of code). We can roughly estimate that yearly maintenance of the samly2 library takes between 1 to 4 work days.

Conclusion

Comparison of the two rightmost columns in the first table above clearly shows that OpenSAML 2 is too verbose and too low level to be used directly in UNICORE. It must be wrapped to provide a more high-level interface which is now provided by samly2. The natural place to put this wrapping code is the current samly2 library: reusing even some parts of its existing external API will make the integration easier. In another words samly2 would need to be rewritten using OpenSAML2 internally. Such the approach is assumed below.

What we gain?

- It can be estimated (see the first table, columns 2. and 4. and discussion of legacy code support in digital signature code) that after rewriting samly2 using OpenSAML the resulting implementation will be from 600 to bit over 1000 lines shorter then the current samly2 version. This can result in a lower maintenance effort. It is impossible to assess it precisely but even assuming optimistically that effort will be reduced by two (when code size is decreased by 30-40% only), we get from 0.5 to 2 work days less per year.

How much does it cost?

- Effort to reimplement the library can be assessed from 1 PM up. In a pessimistic variant the problems with the conflicting libraries may render the whole task impossible.
- After even a most careful implementation we may assume an increased maintenance for a period of one year, after the wrapping code will be as stable as currently samly2 is.

What do we loose or risk?

- UNICORE will be little bit slower and startup time of UNICORE client will be increased by approx. 2s.
- The exchange of the very popular XMLBeans library which is used internally by samly2 with over 70 KLOC long xml-tooling (being part of OpenSAML) is risky as the latter is for sure not as well tested.
- Depending on solution chosen to workaround conflicting dependency problems we may get random runtime faults.
- Likely we will have to drop support for older UNICORE server installations. Otherwise potential gains are much lower.
- After finding a bug we can't fix it ourself, and bugs in an even presumably very good implementation which has 170k lines of code are quite possible.

Bottom line

Even not looking at "risk" section the potential efforts are incomparable to the effort, meaning that the effort

which is required to perform the development won't pay-off in any realistic time-frame (for sure not in the next 20 years). Adding to this what we are risking or loosing makes this statement even harder.

-- KrzysztofBenedyczak - 21-Dec-2010

This topic: EMI > UNICORESecurityPTSopenSAML
Topic revision: r6 - 2011-01-03 - unknown