Identifying Opportunities for Web Services Security Performance Optimizations

Robert A. van Engelen and Wei Zhang
Department of Computer Science, Florida State University, Tallahassee, FL32306
{engelen,wzhang}@cs.fsu.edu

Abstract

WS-Security is an essential component of the Web services protocol stack. WS-Security provides end-to-end security properties, thereby assuring the participation of non-secure transport intermediaries in message exchanges, a key advantage in Web-based systems. However, compared to point-to-point secure messaging with TLS, WS-Security has a significant performance penalty. In this paper, we identify several opportunities for optimizing WS-Security.

1. Introduction

It is well known that WS-Security adds significant overhead to XML Web service message processing due to the inherent cost of repeated cryptographic operations on XML message parts. By contrast, point-to-point security protocols, such as TLS, provide fast but limited security options for Web services. TLS requires negotiation and handshake, which makes it more difficult to secure message exchange patterns (MEP) other than request-response. WS-Security on the other hand supports any MEP and allows messaging intermediaries to access non-encrypted message parts, e.g. to obtain message routing information.

Several authors evaluated the overhead of WS-Security. In [2] and also [4] the authors compare the performance of WS-Security operations and choice of signature and encryption algorithms to non-secure messages using various message sizes and complexities. WS-Security optimizations are not considered. In [3] the authors compare the performance of Web Services, WS-Security, RMI and EMI-SSL extensively. In [8] a performance comparison is presented between security mechanisms based on WS-Security, specifically for Grid services. All of these studies conclude that WS-Security adds significant overhead and can slow message processing by as much as a factor 100. However, these papers assume worst-case scenarios and do not identify nor investigate the impact of WS-Security optimizations.

In this paper, we identify opportunities for optimizing WS-Security by examining the overhead of WS-Security operations in more detail. Hot-spots are starting points for optimization, which can be addressed by algorithmic changes and option selections in the WS-Security protocol.

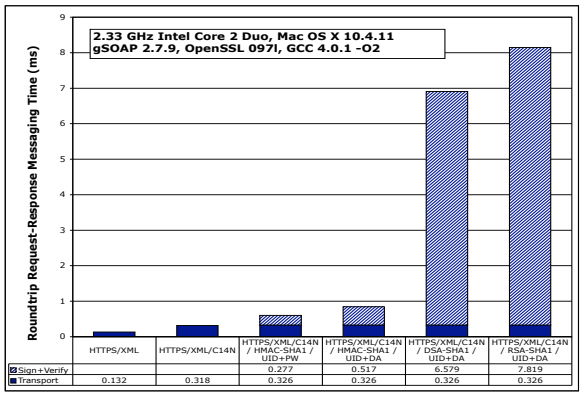

Figure 1. Total time (ms) of canonical (C14N) SOAP/XML request-response messaging over HTTPS with and without HMAC-SHA1, DSA-SHA1, and RSA-SHA1 signatures and UID authentication with passwords (PW) or password-digest authentication (DA).

2. Identification of WS-Security Optimizations

The choice of cryptographic algorithms and associated tokens plays a major role in the overhead of WS-Security operations [3, 6, 7]. From Figure 1 we can see that HMAC outperforms RSA/DSA by an order in magnitude. These results are obtained with a highly-optimized version of gSOAP [9] with OpenSSL. Any optimization in combination with RSA/DSA is pointless, since it will not have much of an impact on improving overall performance.

Thus, HMAC keys are our starting point for further improvement. Well-established key exchange methods can be used to provide a secure mechanism for mutually agreeing on a shared secret HMAC key for signature validation.

To analyze how much overhead WS-Security with HMAC adds to messaging, Figure 2 shows the timing breakdown of a one-way WS-Security signed and authenticated message as a percentage of the total time. XML messaging over HTTPS takes 32.2% (SEND, RECV, TCP/IP, HTTPS). The other timings are 40.4% for WS-Security with HMAC and 26.0% for canonicalization (C14N). While HMAC is fixed, the latter two stages can be optimized.

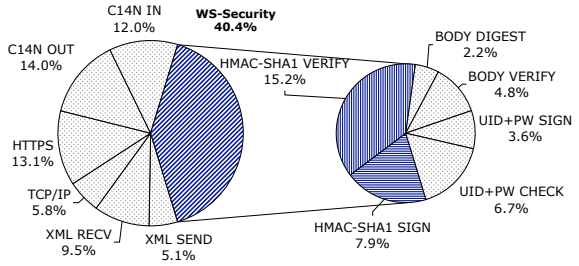

Figure 2. Timing breakdown of WS-Security operations (% of total time).

Note that HMAC-SHA1 signature signing incurs 7.9% overhead of the total time and HMAC-SHA1 verification incurs 15.2% overhead, Together they are the most substantial stages of the HMAC-based algorithm and cannot be easily optimized, unless hardware acceleration is used. The remaining stages (C14N IN/OUT, BODY DIGEST/VERIFY, UID+PW SIGN/CHECK) can be optimized. So we can expect up to 43.3% performance improvement in the best case.

Note that in this example only one element was signed, the SOAP Body. Additionally signed elements will incur a similar cost, so it is important to *keep the total number of signed elements to a minimum*, i.e. sign the SOAP Body root element rather than individual subelements. The cost of signing (BODY SIGN) can be reduced with buffering. The signature in the header is computed by serializing the message body first, which can be buffered to avoid reserialization. In addition, an approach similar to *differential deserialization* [1] can be used to reduce the verification time (BODY VERIFY) when messages contain identical elements that have been signed and deserialized before, assuming they are stored together with their digest in a cache. The deserialized form of a previously parsed signed element can be retrieved from the cache based on its digest value.

XML re-canonicalization (C14N IN) contributes 12.0% to the overhead. In [5] a streaming validation model for signature processing is described to reduce the cost of re-canonicalization at the receiving side (C14N IN). However, only a small cost reduction is obtained with this strategy. By contrast, canonicalization overhead can be completely eliminated for messages that do not require it, since *in most cases, re-canonicalization is unnecessary*. If the message is not reformatted in transit, the C14N OUT and IN stages can be removed from the algorithm to save 26.0% overhead. When messages are potentially reformatted, then on-demand re-canonicalization (C14N IN) and signature verification (BODY VERIFY, HMAC-SHA1 VERIFY) can be performed by the recipient after the message signature verification failed due to reformatting. Suppose a fraction $0 \leq \alpha \leq 1$ of the messages do not require re-canonicalization. Let us define

$$cost_{\mathrm{C14N}} = \text{re-canonicalization time} \ / \ \text{total time} \approx 12\%$$

then the expected message processing cost for the on-demand re-canonicalization is given by

$$cost = \alpha(100\% - cost_{\mathrm{C14N}}) + (1-\alpha)(200\% - cost_{\mathrm{C14N}})$$

giving a cost reduction $cost < 100\%$ if $\alpha > 100\% - cost_{C14N}$. The lower the re-canonicalization cost the higher the success rate α to a gain a performance benefit. For example, given $cost_{C14N} = 12\%$, a performance gain of up to 12% is achieved as long as 88% of the messages do not require re-canonicalization.

User authentication (UID+PW) can be expensive when password digest methods are used that require hashing, password lookups, and temporary storage of nonces to protect against replay attacks. Figure 1 shows that 10.3% overhead is spent on verifying 10,000 user names with cached nonces in a linear linked list search.

To conclude, in this paper we identified performance hotspots of WS-Security processing and suggested existing and new approaches to optimize several WS-Security stages.

References

[1] N. Abu-Ghazaleh and M. Lewis. Differential deserialization for optimized SOAP performance. In *proceedings of the ACM/IEEE conference on Supercomputing*, pages 21–31, Los Alamitos, CA, 2005. IEEE Computer Society.

[2] S. Chen, J. Zic, K. Tang, and D. Levy. Performance evaluation and modeling of Web services security. In *Proceedings of the IEEE International Conference on Web Services (ICWS'07)*, pages 431–438, 2007.

[3] M. B. Juric, I. Rozman, B. Brumen, M. Colnaric, and M. Hericko. Comparison of performance of Web services, WS-Security, RMI, and RMI-SSL. *Journal of Systems and Software*, 79(5):689–700, 2006.

[4] H. Liu, S. Pallickara, and G. Fox. Performance of Web services security. In *13th Annual Mardi Gras Conference*, Baton Rouge, Lousiana, USA, Feburay 2005.

[5] W. Lu, K. Chiu, A. Slominski, and D. Gannon. A streaming validation model for SOAP digital signature. In *In 14th IEEE International Symposium on High Performance Distributed Computing (HPDC-14)*, 2005.

[6] S. Makino, K. Tamura, T. Imamura, and Y. Nakamura. Implementation and performance of WS-Security. *Int. J. Web Service Res.*, 1(1):58–72, 2004.

[7] A. Moralis, V. Pouli, M. Grammatikou, S. Papavassiliou, and V. Maglaris. Performance comparison of Web services security: Kerberos token profile against X.509 token profile. In *ICNS '07: Proceedings of the Third International Conference on Networking and Services*, page 28, Washington, DC, USA, 2007. IEEE Computer Society.

[8] S. Shirasuna, A. Slominski, L. Fang, and D. Gannon. Performance comparison of security mechanisms for Grid services. In *GRID '04: Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04)*, pages 360–364, Washington, DC, USA, 2004. IEEE Computer Society.

[9] R. van Engelen and K. Gallivan. The gSOAP toolkit for Web services and peer-to-peer computing networks. In *proceedings of the IEEE International Symposium on Cluster Computing and the Grid*, pages 128–135, Los Alamitos, CA, May 2002. IEEE Computer Society.