Traffic Evaluation of a Claim-based Single Sign-On System with Focus on Mobile Devices

Mateusz Khalil
Fraunhofer Fokus
Berlin, Germany
Mateusz.khalil@fokus.fraunhofer.de

Yacine Rebahi
Fraunhofer Fokus
Berlin, Germany
Yacine.rebahi@fokus.fraunhofer.de

Simon Hohberg
Fraunhofer Fokus
Berlin, Germany
Simon.hohberg@fokus.fraunhofer.de

Pascal Lorenz
University of Haute Alsace
Colmar, France
Lorenz@ieee.org

Abstract— **The work on Web services security is not that new; however, the provided solutions either are not efficient, or applicable only to some special cases. As an example, the standardization group *OASIS* specified a huge amount of standards, which are unofficially called *WS-**. Unfortunately, not all of these standards are implemented by modern frameworks and their applicability to mobile Web services is also questionable. As the WS-* technologies are very useful in realizing single sign on (SSO) solutions, we discuss, in this paper, an implementation prototype for a single sign-on system with WS-* standards and evaluate the resulting network traffic while focusing on mobile devices. To be more precise, two implementations (one based on NetBeans and another one developed by the authors themselves) were achieved and their corresponding results were compared. It appears that the authors' implementation performs better that the NetBeans one and reduces the traffic generated from 85% to 50%.**

Keywords-Web traffic; Single Sign-On; Mobile devices; SOAP; OASIS; WS-*.

I. INTRODUCTION

A crucial benefit for the emerging Web services' architectures is the ability to deliver integrated, interoperable and secure solutions. Ensuring the protection of Web services from attacks and misuse through the enforcement of comprehensive security models is critical.

The work on Web services security is not that new, however the provided solutions either are not efficient, or applicable only to some special cases. In the literature, there are some suggested security mechanisms, namely, Security Assertion Markup Language (SAML) [1], which is an XML-based standard for exchanging authentication and authorization data between security domains, that is, between an identity provider and a service provider. SAML was specified by the OASIS [2] Security Services Technical Committee and was intended to solve the *Web Browser Single Sign-On* (SSO) problem. Single sign-on solutions are abundant at the intranet level (using cookies, for example) but extending these solutions beyond the intranet has been problematic and has led to the proliferation of non-interoperable proprietary technologies.

The standardization group *OASIS* [2] specified a huge amount of standards, which are unofficially called *WS-**[3]. Unfortunately, not all of these standards are implemented by modern frameworks and their applicability to mobile Web services is also questionable. As the WS-* technologies are very useful in realizing single sign on (SSO) solutions, we discuss, in this paper, two implementation prototypes (one based on the NetBeans [16] technology and the other one not) for a single sign-on system with WS-* standards and evaluate the resulting network traffic while focusing on mobile devices.

The outline of the rest of this paper is as follows. Section two provides an overview of the techniques needed by the subsequent parts. Section three discusses the proposed approach and section, four presents the experimental results. Finally, section five concludes the paper.

II. BACKGROUND

A. Single Sign-On

Single sign-on (SSO) is an identity management model where the user needs to provide his credentials only once and will stay authenticated against the realm, which may include multiple services within the concerning *Circle of Trust*. There are many different implementations of SSO with prominent representatives like Kerberos [4] or OpenID [5]. Main parts of a SSO system in a decoupled claim-based scenario are Client (C), Relying Party (RP) providing a service and Identity Provider (IdP) in the role of a STS. Usually, the first step is a service request of C to RP. Then RP forwards the Client's request to a trusted IdP, because

 ISBN: 978-1-61208-199-1

RP requires identity information from C in order to check a defined authorization policy. To prove authorization C needs to provide a service token to RP. This service token is provided by the IdP when C authenticates using its credentials. Figure 2 shows a sequence how such a service token is requested. In the sequence in the figure, C first requests a security context token (SCT) sending a request security token (RST) message containing C's credentials. The STS (IdP) answers by sending a request security token response (RSTR) containing the SCT if C could be authenticated. Now C can use this valid SCT to request the service token from the STS (IdP) in the next step. Hereon, the STS validates the SCT and provides the service token and any requested attributes. With this service token and the attributes C is able to request the service (RP), so that the service token is validated successfully and the service policy is fulfilled. Finally, the service (RP) responds to the request.

The SCT's and the service token's integrity are commonly protected by a signature. In some solutions the token is not forwarded by C, but is delivered directly from STS (IdP) to the service (RP). Because there is no need for C to authenticate again after C has received a SCT from STS (IdP) until it has expired and C can use this SCT for further requests to STS (IdP), this mechanism is called single sign-on.

B. Related Work

WS-* [3] contains many specifications defined by the standardization group OASIS. The specifications deal with security issues as well as reliability, transactions and others. In this paper, we focused on the key security-related specifications *WS-Security*, *WS-Trust*, *WS-SecureConversation* and *WS-SecurityPolicy* [2]. These specifications describe how SOAP-messages [6] need to be built with the objective to maintain interoperability. In an SSO system: C, RP and IdP may be developed independently but can interact because of the defined languages specified in WS-*.

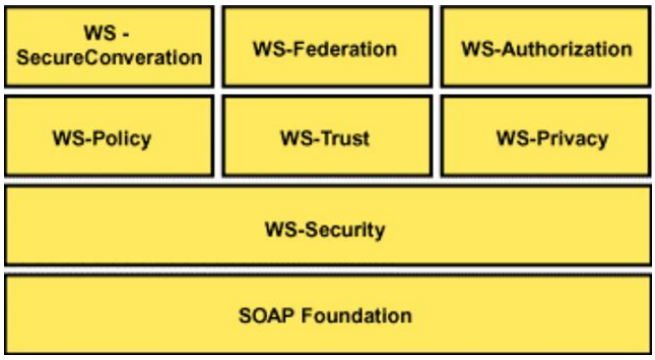

Figure 1: WS secure conversation

WS-Security defines how authentication and authentication information can be stored in the SOAP-Header. Basically other standards and specifications can be embedded into WS-Security.

WS-Trust is a specification which can be regarded as a Web service definition which may validate, create, renew or delete tokens. This Web service is commonly called Security Token Service (STS). The token is used as a proof of authentication (and may be limited in time). It may also contain authorization information.

SOAP is stateless which leads to inefficient usage of bandwidth if all security information needs to be transmitted in every SOAP-message. This problem is solved by **WS-SecureConversation** which provides methods for creating and protecting security sessions. Once a security context has been established, tokens, claims and key information can be stored within the concerning security context.

WS-SecurityPolicy describes a language for the definition of security requirements. For instance a valid security policy may require that the SOAP-body needs to be encrypted using a certain algorithm, expected token types and how the response message needs to be secured. The main benefit is that this policy can be published, which means that the consumer may adapt himself to the requirements. Another aspect of the policy driven development is its readability and efficiency concerning development effort.

Finally, the XML security recommendations consisting of **XML-Encryption** and **XML-Signature**, which were introduced by W3C [7], specify how to encrypt and sign any elements within an XML document. This implies that *XML-Encryption* and *XML-Signature* can be applied in SOAP-messages as well.

III. Our Approach

Web services have many benefits and can be used in various scenarios. Nowadays, most security dependent Web services are secured by SSL only. For instance we may have a production chain where many producers participate in. Assuming that parts of the SOAP message are confidential and have to be readable by one certain producer. SSL encrypts on transport level but an encryption on message level is required. This is where WS-* and XML security standards come into play. These technologies are also very useful in order to realize a single sign on (SSO) system. Especially in the production chain example, identity management of all participating parties has to be considered.

As not all of these standards are implemented by modern frameworks, a description of critical features provided in *Apache Axis2*, *Apache CXF* and *Glassfish Metro* is crucial.

A. Frameworks

Apache Axis 2 [8], Apache CXF [17] and Metro Glassfish [9] are popular Web service frameworks. They act as SOAP engines and have additional libraries which provide security functionalities. We have summarized which specifications

 ISBN: 978-1-61208-199-1

and features are supported by these frameworks. In our opinion *Metro Glassfish* in combination with *NetBeans* is the most effective choice (we refer to Table 1). NetBeans [16] is an extensible integrated development environment (IDE) providing necessary tools for developing desktop, enterprise, Web and mobile applications. Metro is simply an open source Web service stack that is a part of the Glassfish project.

B. Use Case

We created a simple Web service with *NetBeans* which was secured by WS-* technologies using *Metro Glassfish*. In particular WS-Security (UsernameToken Profile) WS-Trust, WS-SecureConversation, WS-SecurityPolicy and WS-MetadataExchange are used. The identity management applied the described decoupled claim-based SSO architecture. In our experiment, we assume that a user authenticates himself by supplying his username and password token to the identity provider (IdP) (see Figure 2), who will create a SAML-Assertion describing the authentication status of the user and hand out a symmetric key which will be used to secure the connection to the Relying Party (RP). Finally, this assertion token is used to call a policy-constrained Web service.

Furthermore, we implemented ourselves a SSO system in the Ubipol project (see section V) using the WS-* technologies. In this project, we used the same architecture as already described.

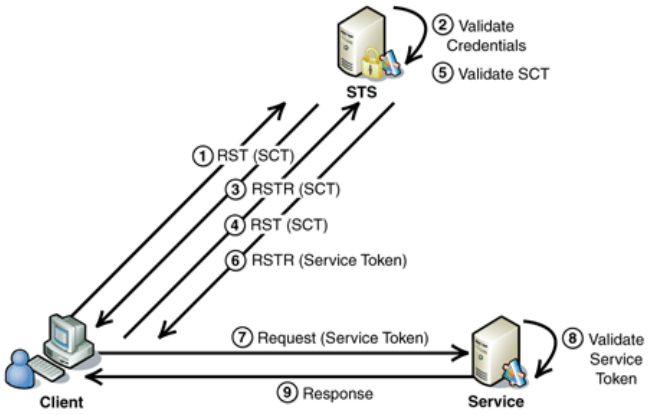

Figure 2: WS secure conversation

IV. Experimental Results

We have analyzed the traffic which was generated by decoupled claim-based scenarios. First of all, a simple application has been employed, which makes use of the *Metro* Web service stack implementation (*Metro 2.1.1*). *WSIT* is part of *Metro* and provides methods for creating reliable, transactional and secure Web services using standards specified by the standardization group OASIS.

Second, we created our own implementation and compared the traffic of both scenarios.

No compression is used for the traffic analysis in order to inspect the distribution of the traffic regarding the Web service security standards. We will distinguish between the first and second (or further) Web service call made by the client. All packets which are necessary for succeeding the request are regarded as one package. This may involve additional traffic to other recipients, in particular IdP. All Web service security relevant XML elements were considered and their generated traffic has been assigned to the related standard based on the qualified name which itself and only its content from the same standard are part of the counted size.

In the first scenario, the first user's Web service call is accompanied by 4.5 times more traffic than the further ones (see Figure 3). This is caused by bootstrapping the Web service, which contains the policy and *WSDL* retrieval, but also the token request to the IdP needs to be done. The former aspect explains, why *WS-Trust* and *WS-MetadataExchange* are not represented in following Web service calls, because they are part of the initial token exchange process.

All further requests to RP require less traffic, because all relevant keys have been established and put into a security context. As defined in *WS-SecureConversation*, derived keys are used for protecting the transported content. In the first request, the security standards use 85% of the traffic. This reduces in any further requests to 33%.

In the second scenario, where we analyzed the traffic of our own implementation, the traffic of the first request is also much higher than in any subsequent requests. Anyhow, the overall package size is much smaller since our implementation does not use any *Metadata-Exchange* and also the transferred data of *WS-Security* is slightly lower. Therefore the ratio between security data and payload transported is much better. In the first request, the security data creates 54% of the total package size and in further requests lowers to only 8% (see Figure 4).

Even though modern mobile networks (3G and 4G) solve bandwidth problems, high-latency issues still exist. In the bootstrapping process the client request is forwarded to the IdP, which implies an additional expensive Web service call. When dealing with low traffic Web services the initial waiting time may be inconvenient due to the collateral traffic and the mentioned further forward to the IdP.

V. Security in Mobile Based ePArticipation

eParticipation refers to the ICT mechanisms for the citizens to express their opinions in order to influence political, economic or social decisions. Recently, the rapid advance in mobile computing technologies also facilitated the emergence of mParticipation (mobile participation) to allow citizens to be involved in Policy Making Processes (PMPs) even on the move. The UbiPOL project [10] aims to

 ISBN: 978-1-61208-199-1

develop a new governance model in which citizens can participate in policy making processes in the middle of their everyday life overcoming spatial and time barriers. The core of the governance model is a ubiquitous participation platform that motivates its users to be involved in PMPs. Depending on the status of the relevant policies, the citizen may add his own opinion on his handheld device which will deliver the opinion to the opinion base of the relevant government agency. The collected citizen opinions for each site objects will be connected with relevant policy objects to be used for policy making process.

One of the main objectives of UbiPOL is to develop a framework that

1. Ensures citizens privacy in filtering citizen opinions
2. Secures the communication between the mobile device and the Ubipol platform
3. Ensures the anonymity of the user in case of opinion casting. This means the association between the opinions records and the user identity must remain unknown
4. Prevents multiple opinions casting
5. Manages the Ubipol platform users identities and regulates the access to it according to the user role

In the UbiPOL project, Fraunhofer Fokus is in charge of the items (2), (3), (4) and (5). Contrary to the work discussed in the literature about anonymous voting ([11], [12]) where some complex algorithms are suggested, we came out in this project with a simple Web-based architecture that addresses the above issues while taking into account the limitations of the mobile devices. The architecture is already specified and is being implemented and tested on Android capable mobile devices. For the time being, simple credentials (user name, password) are used for authentication and voting tickets acquisition. In the near future, we will enhance the framework with the use of the new German eID cards.

VI. CONCLUSION AND FUTURE WORK

This paper provided an introduction in state of the art specifications for securing Web services. Then a scenario has been described where these technologies were used in order to implement a single sign-on system. Based on two implementations (one using NetBeans and another one developed by the authors), a traffic analysis has been performed and showed that WS-* technologies can produce high overhead (more than 85% of the entire bandwidth) in low-traffic Web service especially when using *Metadata-Exchange*. In the future, we will further optimize our realization for mobile devices by investigating the security standards and group them into profiles that can be used for appropriate scenarios.

ACKNOWLEDGMENT

This work was achieved within the UbiPol project [10]. UbiPol is supported by the European Community under the FP7 ICT Work Programme (call: ICT-2009.7.3 (a)).

REFERENCES

[1] SAML, "Security Assertion Mark-up Language". Link http://saml.xml.org/saml-specifications, Access: March 2012

[2] OASIS: "Advancing open standards for the information society", link: http://www.oasis-open.org/, Access: March 2012

[3] WS-Trust, link: http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/ws-trust.html, Access: March 2012

[4] KERBEROS: The Network Authentication Protocol, link: http://Web.mit.edu/kerberos/, Access: March 2012

[5] OpenID, link: http://openid.net/foundation/, Access: March 2012

[6] SOAP messages, link: http://www.w3.org/TR/soap/, Access: March 2012

[7] W3S, link: http://www.w3.org/, Access: March 2012

[8] Apache Axis 2, link: http://axis.apache.org/axis2/java/core/, Access: March 2012

[9] Metro Glassfish, link: http://en.wikipedia.org/wiki/GlassFish_Metro, Access: March 2012

[10] The Ubipol project, link: http://www.ubipol.eu/, Access: March 2012

[11] A. Y. Lindell, "Anonymous Authentication", Aladdin Knowledge Systems Inc, Bar-Ilan University, Israel, 2006. Link: http://www3.safenet-inc.com/blog/pdf/AnonymousAuthentication.pdf, Access: March 2012

[12] K. Sako; S. Yonezawa; and I. Teranishi; "Anonymous Authentication: For Privacy and security", NEC Journal of Advanced Technology, Special Issue on Security for Network Society, Vol. 2, No. 1, 2005.

[13] Project U-Prove, link: http://www.fokus.fraunhofer.de/de/fokus_testbeds/secure_eidentity-lab/projekte/u_prove/index.html, Access: March 2012

[14] Is Microsoft's U-Prove the answer to better online privacy, link: http://www.networkworld.com/community/blog/microsofts-u-prove-answer-better-online-priva, Access: March 2012

[15] NetBeans IDE integration, link: http://glassfish.java.net/public/netbeans/index.html

[16] NetBeans, link: http://en.wikipedia.org/wiki/NetBeans

[17] Apache CXF, link: http://cxf.apache.org/, Access: March 2012

 ISBN: 978-1-61208-199-1

Feature	Axis 1.x	Axis2	CXF	Glue	JBossWS	XFire	Metro@GlassFish	OracleAS 10g
WS-Addressing	X	X	X	X	X	X	X	with BPEL
WS-Atomic Transaction	X	X					X	
WS-Business Activity		X						
WS-Coordination	X	X					X	
WS-Eventing		X			X			
WS-Metadata Exchange		X [10]	X				X	
WS-Notification	X	X [12]	X	?		?		
WS-ReliableMessaging	X	X	X				X	
WS-Policy		X	X				X	X
WS-Secure Conversation		X	X				X	
WS-Security Policy		X	X				X	
WS-Security	X	X	X	X	X	X	X	X
WS-Trust		X	X				X	
WS-Transfer		X						
WSDL 1.1 Support	X	X	X	X	X	X	X	X
WSDL 2.0 Support		X						

Table 1: Comparison of the different framework

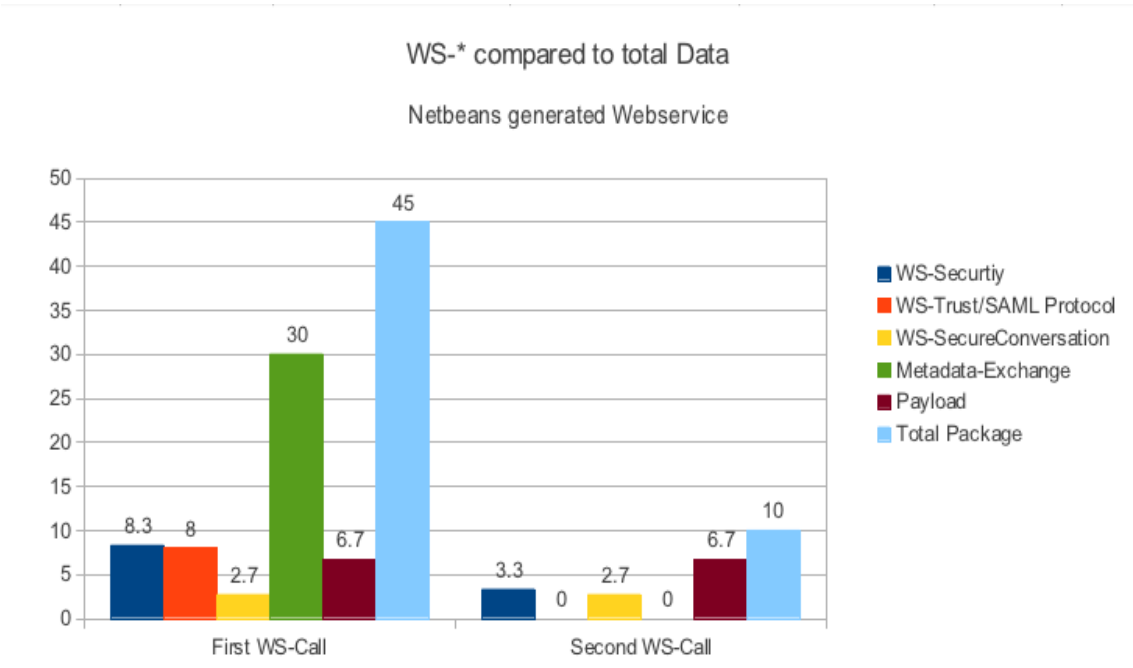

Figure 3: WS-* Compared total data

 ISBN: 978-1-61208-199-1

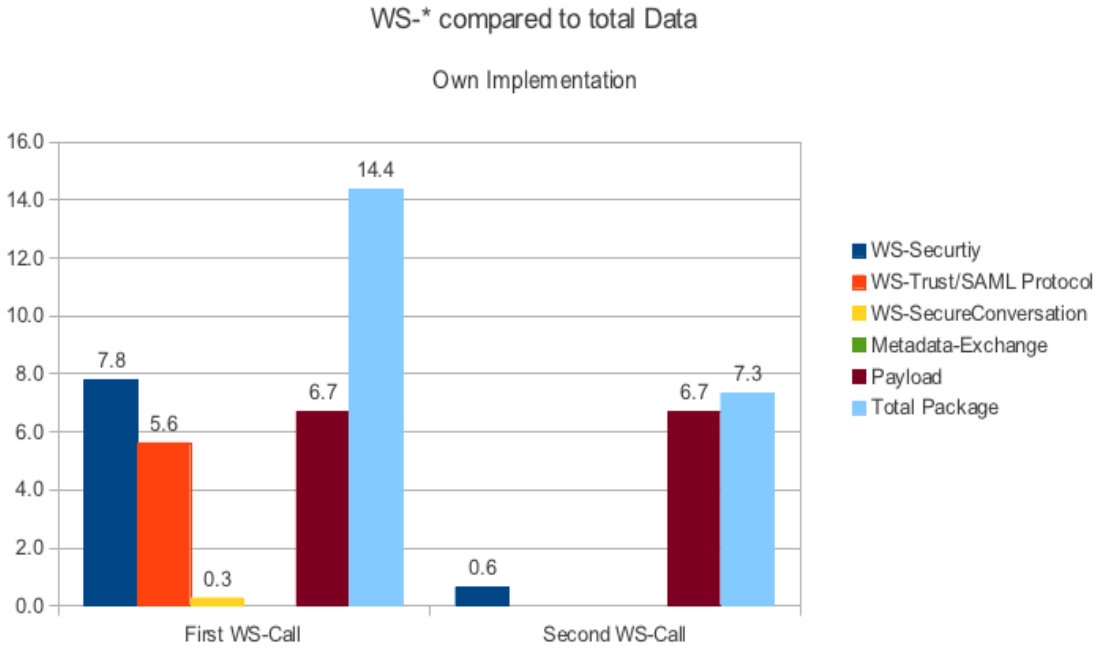

Figure 4: Our WS-* implementation compared to total data

 ISBN: 978-1-61208-199-1