Security Vulnerabilities in SAML based Single Sign-On Authentication in Cloud

Kirandeep Kaur, M.E. Research Scholar
Computer Science Engineering
PEC University of Technology
Chandigarh, India
Kiran.bbsbec@gmail.com

Dr. Divya Bansal, Associate Professor
Computer Science Engineering
PEC University of Technology
Chandigarh, India
divya@pec.edu.in

***Abstract*— Cloud computing is introducing numerous changes to one's lifestyle and working pattern for its infinite benefits. Companies have increasingly turned to Software as a Service (SaaS) or Application Service Providers (ASPs) vendors to offer specialized web based services that have huge potential to cut costs and provide specific applications to the users in a very convenient way. However, the security of cloud computing is always a serious issue for numerous potential cloud users, and also a big roadblock for its far-flung applications. One of the major challenges remains to be an integrated authentication mechanism over cloud environments through Single Sign-On. In this paper, the authors report their work of implementing Security Assertion Markup Language (SAML) to enable Single Sign-On (SSO) based authentication in a multiple web application cloud environment. The paper also reports serious vulnerabilities prevalent in such an environment and describes a detection method for the same.**

***Index Terms*— SSO, SAML, Authentication, Confidentiality, Availability, Integrity, Vulnerability**

I. INTRODUCTION

With increasing number of web based systems and applications, end users have to memorize and keep multiple usernames and passwords for each system and application. This poses an additional challenge to developers and support staff as many end users invariably forget credentials to less commonly used applications. The same password is also used for multiple accounts leading to weakening of the security of authentication systems.

Single Sign-On (SSO) [1] protocols attempt to address this issue by allowing a user to enter credentials once to authenticate across multiple systems and applications which is prevalent in today's cloud environment. This is commonly accomplished by having an identity provider that maintains user credentials which are then passed to relying party to authenticate users [2].

For the accomplishment of Single sign-on authentication mechanism, SAML has been adopted over other existing SSO products like Microsoft passport [2], OpenID [3] due to their inherit phishing vulnerabilities and also because they didn't establish a trust relationship between identity provider and service provider causing a malicious service provider to easily configure their authentication mechanism to redirect the user to their own identity provider. These can be further designed to be visually identical to the legitimate identity provider misleading the cloud user.

Over the years various products have been providing support for web-based SSO. These products typically depend on browser cookies to maintain user authentication state information so that re-authentication is not required, each time the user accesses the cloud resources. Since browser cookies are not transmitted between DNS domains, the authentication state information of users in the cookies from one domain is never available to another domain [4].

These products therefore have typically supported cross domain SSO (CDSSO) [4] through the use of proprietary mechanisms to pass the authentication state information within the domains.

The security of a SAML SSO solution critically depends on several assumptions such as trust relationship amongst the involved parties and security mechanisms like the secure transport protocols used to exchange messages. Many security recommendations that are available throughout the SAML specifications are useful in avoiding the most common security pitfalls but are of little help. Therefore, it is very difficult to achieve the needed level of assurance.

II. SECURITY ASSERTION MARK-UP LANGUAGE (SAML)

The Security Assertion Mark-up Language (SAML) is an XML based language designed for making security statements about subjects. SAML assertions are used as security tokens in WS-Security and in REST based Single Sign-On (SSO) scenarios [5]. Several profiles are defined in [6] and the most important profile is the Browser SSO profile, which defines how to use SAML with a web browser.

In SAML based Single Sign-on approach, users authenticate only once to a trustworthy identity provider (IdP). After a successful login of a user, the identity provider issues security tokens on demand. These tokens are used to authenticate to Relying parties (RP) [4].

III. RELATED WORK

The first browser based SSO protocol was Microsoft Passport which originally intended as an authentication mechanism for its Hotmail service, then later reintroduced as a fully featured SSO targeted at online shopping sites. It supported multi factor authentication like mobile device

registration which contained their mobile number and a custom PIN (Personal Identification Number) and several mechanisms for preventing attacks like if a user enters a password incorrectly five consecutive times, .NET Passport automatically block access to account for two minutes [7]. Besides several advantages, Microsoft Passport faces several challenges for their existence as many of the online merchants using the passport service for online purchases moved away from the platform due to flaws that are revealed [8].

Kelly D. Lewis et al. [9] has described the implementation of Security Assertion Markup Language and its capabilities to provide secure single sign-on (SSO) solutions for externally hosted applications. They mentioned that by using SSO solutions, user experience is enhanced by eliminating additional usernames and passwords but there are some major security issues like replay attack which exploits the credentials that are exchanged between asserting party and relying party.

Projects like The Liberty Alliance Protocol [10] and Shibboleth [11] base their protocol on the SAML message standard. But Liberty Alliance Protocol is not a standardized process [12] whereas Shibboleth project is a SAML application for inter-university federation.

Jorg Schwenk, et al. [13] presents an in-depth analysis of SAML frameworks and shows that SAML v 2.0 has critical XML Signature Wrapping (XSW) vulnerabilities. They showed that the application of XML Security heavily depends on the underlying XML processing system. They proposed a formal model by analyzing the information flow inside the relying party and presented countermeasures XML Signature Wrapping (XSW). XSW attacks have first been described by McIntosh, M et al. in 2005 [14].

McIntosh et al. [14] have presented several XSW attacks and discussed receiver-side security policies in order to prevent such exploits. They have however not given a proper solution for this problem.

Wang et al. [15] describes the importance of SSO protocols. This work has analyzed the security quality of commercially deployed SSO solutions. It has shown eight serious logic flaws in high-profile IdPs and RPs, which have allowed an assaulter to login as the victimized user.

While going through existing research work, we found that application of XML security heavily depends on the underlying XML processing system. This processing system involved can have inconsistent views on the same secured XML document, which may result in successful XML Signature Wrapping (XSW) attacks. So after the creation of SAML assertion used for authentication, compliance policy of SAML needs to be checked and non-conformance issues need to be reported and addressed. In this paper, we have implemented authentication mechanism using OpenSAML v2.0 to enable SSO. Then, we have detected serious vulnerabilities in such an environment and proposed solutions to mitigate the same.

A. Comparative analysis of SSO protocols

A number of solutions for browser-based SSO are available: the OASIS *Security Assertion Markup Language* (SAML) 2.0 [4], Microsoft Passport [2], the Liberty Alliance project [12], the Shibboleth Initiative [11], and OpenID [3] are the most popular.

An adoption of SAML for SSO authentication on the basis of comparison on some important factors has been described in table I [16]:

TABLE I Comparison of SSO protocols

	OpenID	**SAML**
SP Initiated SSO	Yes	Yes
IdP Initiated SSO	No	Yes
IdP Discovery	Configured per user	Configured per account
Just in time provisioning	Indirectly via back channel	Directly
Performance	Slower	Faster
Implementation	Simpler	Complex
Positioning	Consumer	Enterprise

As it is clear from table I that, SAML has advantages over OpenID in terms of faster performance, User provisioning for the end user as compared to OpenID. Further, SAML is enterprise based which means that an organization needs to have single Identity provider for an application and all users in that application can sign in with their e-mail address and username whereas OpenId is user centric which means that every user has its own OpenID registered in that application. Because of the above mentioned advantages SAML has been adopted by organizations.

IV. Solution Approach/Proposed Scheme

In this section, we introduce the SAML framework that describes how SAML assertion is created that is used for authentication mechanism. Following are the points that give a brief idea about how process flow:

- Create SAML assertion using OpenSAMLv2.0 between service provider and identity provider.
- Identify vulnerabilities and how adversary can exploit it.
- Analyze the usage of SAML assertion and detect possibilities of inserting malicious content to exploit those vulnerabilities.
- By using Common vulnerability Scoring System (CVSS) [18] tool, check how vulnerable the application is.
- On the basis of detection of vulnerabilities, build different approaches to overcome.
- Again use tool to check vulnerabilities and compare score created before and after hardening.

SAML framework (see Fig 2) can be explained as:

We develop an assertion which is created by using open source products available like we used OpenSAML between identity provider and service provider. Following we describes step by step procedure that how assertion is created by using OpenSAML and mechanism that are used while authenticating the user who accessing the resources related to Service provider. It is Java-based and can be implemented in Java application environments with relative ease through the use of JSP tags and servlet filters.

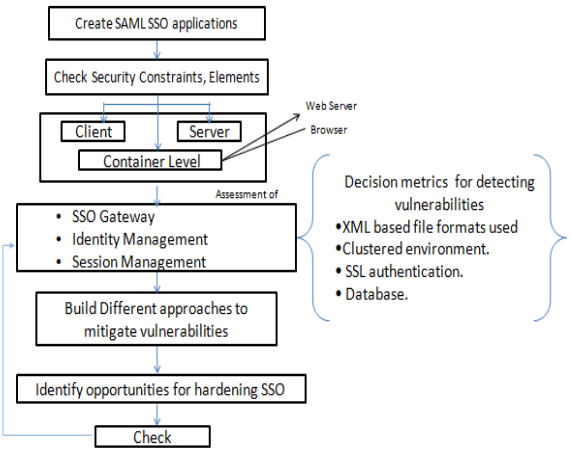

Fig.2. Overview of SAML Framework

A. Analysis of SAML structure

The structure of SAML assertion has been described in Fig.3.The issuing time of the assertion is defined in **saml:IssueInstant.** The **saml:Issuer** element specifies the IdP that is making claims in the assertion. **saml:Subject** defines the principal about whom all statements are made.

<saml:Assertion Version ID IssueInstant>
<saml:Issuer>
<ds:Signature>?
<saml:Subject>?
<saml:Conditions>?
<saml:Advice>?
<saml:AuthnStatement>*
<saml:AuthzDecisionStatement>*
<saml:AttributeStatement>*
</saml:Assertion>

Fig.3. SAML assertion structure
("?" represent zero or one occurrence; "*" represent zero or more occurrence)

To protect integrity of claims that are made by the Issuer, whole **saml:Assertion** element must be protected with a digital signature that must follows the XML Signature standard. SAML specification requires that **saml:assertion** must be referenced by signature element, with an enveloped XML Signature (see Fig. 4) [13].

B. OpenSAML Vulnerability

After creation of SAML assertion, identify vulnerabilities and how adversary can exploit it, attacker may register as a user of an Identity Provider IdP. The adversary then receives, through normal interaction with IdP, a valid signed SAML assertion making claimed attacker. The attacker now adds additional claims like evil assertion about any other subject S, and submits the modified document to RP [13].

But glitches in the Apache Xerces library which execute a schema validation process of each incoming XML message creates problem in the processing of XML elements that are defined with *xsd: any* and moreover the content of the elements which are defined *<xsd: any processContents=''lax"*> are processed incorrectly [17] so it is possible to insert elements with arbitrary and also duplicated Ids inside an XML message. This helps us to create a good position for our wrapped content.

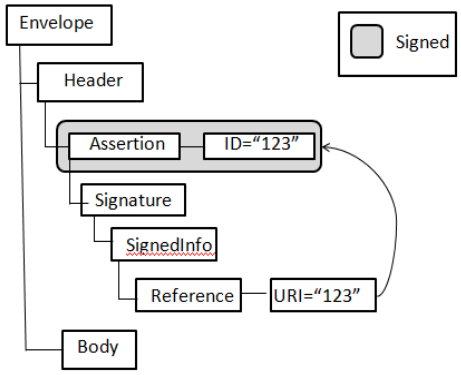

Fig.4. SAML assertion is placed into a root element (header) and signed using an enveloped signature [13]

Implementations of Apache Xerces for Java and C++ handled elements differently. As we are using Java application so we take into account Java implementation. In Java, legitimate assertion has to be placed within or after the evil assertion. In short, if two elements with the same identifiers (Ids) values occurred in an XML message, the XML security library detected only the last element in the message whereas in the case of C++, it will detect first element in the message. This gave a chance to attacker to use this extension for injection of an evil element.

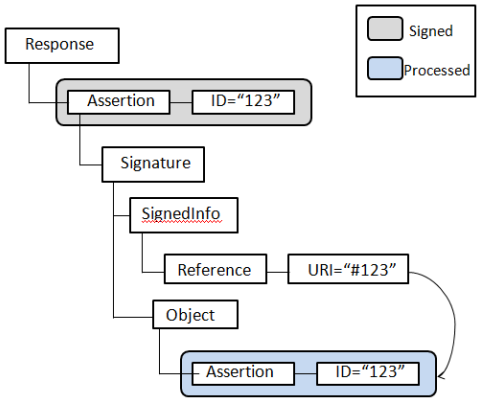

Fig. 5. XSW attack on OpenSAML library [13]

C. Common vulnerability Scoring System

The National Vulnerability Database (NVD) [18] is the U.S. government repository of standards based on vulnerability management data that provides CVSS scores for almost all known vulnerabilities. CVSS is an open source framework designed to provide end users with an overall composite score representing the severity and risk of a vulnerability. It benefits all persons concerned with information security for calculating score for IT vulnerabilities. It is platform and technology independent [18].

These are used to generate both numeric score ranging from 0(least severe) to 10 (Critical) and this indicates the

severity of the vulnerability [18]. A CVSS Score is calculated which is based on 3 sub-scores:

a) Base Metrics,
b) Temporal Metrics and
c) Environmental Metrics

To calculate the severity of vulnerabilities that exists in our framework, we use CVSS base score metrics to calculate the rate as base score metrics is mandatory where as other two metrics are unique to any user's environment. The CVSS Base Score indicates built-in and key characteristics of vulnerability that are constant over time [18].

Equations for calculating CVSS Base score

BaseScore = (0.6 * Impact + 0.4 * Exploitability – 1.5) * f(Impact) (1)

Equation(1) is calculated by the use of three metrics i.e. Impact, Exploitability and f(Impact) and these further depend on sub metrics which are described in the following equations:

Impact = 10.41 * (1 – (1 – ConfImpact) * (1 –IntegImpact) * (1 – AvailImpact)) (2)

In Equation (2), **ConfImpact** describes the degree of breach of information to an unauthorized user or organization. **IntegImpact** specifies the amount of modification that has been done to information when an attacker succeeds in launching an attack. **AvailImpact** reflects accessibility loss of resources due to a successful attack by an assaulter [18].

The possible values for these metrics are: None, Partial, Complete [18] which depicts the possible danger that could affect individuals and/or the organization if resources were inappropriately accessed, used, or disclosed.

Exploitability = 20 * AccessComplexity * Authentication * AccessVector (3)

In equation (3), **AccessComplexity** measures the degree of complexity of target system for launching an attack with possible metrics as high, medium or low [18]. **Authentication** defines how many times an adversary needs to authenticate to reach a target system for injecting an attack with possible metrics as single instances or multiple instances [18]. **AccessVector** describes the possibilities where an attacker can launch an attack with metrics as local, adjacent network or Network [18].

f(Impact) = 0 if Impact=0; otherwise value of **f(Impact)** would be 1.176 (4)

As (4) depend on the value of Impact that is calculated above. Vulnerabilities with a base score are categorized as:

- Critical(7.0-10.0)
- Major(4.0 – 6.9)
- Minor(0-3.9)

V. Result and Discussion

We tested a SAML based Single Sign-On authentication system and proposed mechanism to detecting the severity of vulnerability by deploying in following computing environment.

- Computer environment: Windows 8
- Platform: JAVA
- Web server: Apache-7.0.35
- Installation: OpenSAML v2.0
- Tool : CVSS

Our main aim was to design a secure SAML based Single Sign-on authentication for traffic that routes for accessing external hosted web applications. As we discussed earlier also that SAML is an XML based framework designed for making security statements about user in form of SAML request and response assertions. Moreover XML framework is more prone to vulnerabilities due to its properties of extension elements which in turn lead to XSW attacks. We perform such attack in our framework and make use of CVSS base score [18] for calculating the severity of the vulnerability on the basis of metrics as defined in previous section. A suitable approach has been applied i.e. Data tainting method to suppress such vulnerability from our existing vulnerability and then again use CVSS tool to check how vulnerable the system is even after applying mitigation methods. (See fig 6)

Value of CVSS base score which is calculated when vulnerability exists and when suppress method has been applied on framework is:

Base score before: 6.2
Base score after: 2.7

Fig 6 indicates CVSS base score which in return define the severity of vulnerability. In this red bar indicates score of framework when simulation of XSW attack occurred and it is of medium severity as defined by CVSS experts where as blue bar indicates score of framework when method to fix such attack has applied and it is of low severity which shows we are successful in achieving secure Single Sign-On authentication mechanism.

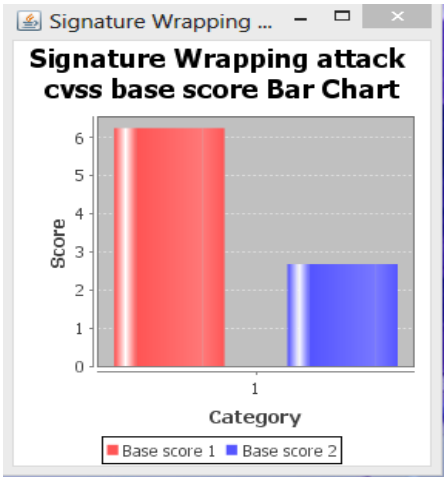

Fig. 6. CVSS Base score

VI. Conclusion

With the use of SAML, organizations can easily and securely share identity based information of the user with other web applications which are integrated with this particular organization. User experience is improved by getting rid of multiple usernames and passwords which in return lower administrative costs.

We found occurrence of new XSW attacks even if the web application is complying with the standard recommended by OASIS technical team. The main reason for the possible attack was that the applications of XML security heavily depend on the underlying XML processing system. These processing modules i.e. Signature validation and business logic involved can have inconsistent views on the XML document which may result in XSW attacks. In order to overcome these attacks, we apply different countermeasures after analyzing the vulnerabilities meanwhile, providing awareness related to SSO specifications to implementers for their specification as well as continuously patching and improving upon their solution as new vulnerabilities are discovered i.e. inevitable.

We can even extend this security measures for federated identity management systems using multi-factor authentication which may include biometric identification that includes different biometric traits like fingerprint, face, palm etc. By using multi-factor authentication, we can hereby increase the security of information or resources both internally and externally.

ACKNOWLEDGMENT

This work is done in Cyber Security Research Center (CSRC) located at PEC University of Technology. The authors would like to thank Government of India, Ministry of Communications and Information Technology, Department of Information Technology, New Delhi, for funding the Project "**Development of Cloud Based Framework for Delivering Security as a Service**", under which this research work has been done.

REFERENCES

[1] San-Tsai Sun, Eric Pospisil, Ildar Muslukhov, Nuray Dindar , Kirstie Hawkey, Konstantin Beznosov Investigating User's Perspective of Web Single Sign-On:Conceptual Gaps, Alternative Design and Acceptance Model. In ACM Transactions on Internet Technology on January 9th, 2012

[2] P. McKiernan. 2002. Addressing Online Identity: Understanding the Microsoft Passport Service. In Information Security Technical Report, Vol 7, No. 3 (2002) 65-80.

[3] D. Recordon and D. Reed. OpenID 2.0: a platform for usercentric identity management. In Proceedings of the Second ACM Workshop on Digital Identity management, Alexandria, Virginia, USA, (2006) pp. 11–16.

[4] OASIS. Security Assertion Markup Language (SAML) v2.0 "http://www.oasis-open.org/ committees/tc_home.php?wg_abbrev=security", April 2005.

[5] Cantor, S., Kemp, J., Philpott, R., and Maler, E. Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS Standard, 15.03.2005.

[6] Cantor, S., Kemp, J., Maler, E., and Philpott, R. Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS Standard, 15.03.2005.

[7] Rolf Oppliger. 2004. Microsoft .NET Passport and identity management. Information Security Technical Report, 9(1):26-34.

[8] Rolf Oppliger. 2003. Microsoft .NET Passport: A Security Analysis. Computer, Vol 36, No. 7 pp. 29-35.

[9] Kelly D. Lewis, James E. Lewis et al. Web Single Sign-On Authentication using SAML. In IJCSI International Journal of Computer Science Issues, Vol. 2, 2009

[10] OASIS Identity Federation.LibertyAlliance Project http://www.projectliberty.org/resources/specifications.php, 2004.

[11] Internet2. Shibboleth Project http://shibboleth.internet2.edu/, 2007.

[12] J. Hodges and T. Wason. Liberty architecture overview, 2003.

[13] Jorg Schwenk, Marco Kampmann, Juraj Somorovsky, Andreas Mayer and Meiko Jensen et al. On Breaking SAML: Be Whoever You Want to Be.

[14] Mcintosh, M., and Austel, P. XML signature element wrapping attacks and countermeasures in Workshop on Secure Web Services (2005).

[15] Wang, R., Chen, S., and Wang, X. Signing Me onto Your Accounts through Facebook and Google: a Traffic-Guided Security Study of Commercially Deployed Single-Sign-On Web Services. In IEEE Symposium on Security and Privacy (Oakland), IEEE Computer Society (May 2012).

[16] Hodges, J Technical Comparison: OpenID and SAML - Draft 07 "http://identitymeme.org/doc/draft-hodges-saml-openid-compare-07a.html" .

[17] THE Apache Software Foundation. Apache Xerces "http://xerces.apache.org"

[18] NVD Common Vulnerability Scoring system "http://nvd.nist.gov/cvss.cfm"