Scientific Journal of Impact Factor (SJIF): 3.134
e-ISSN (O): 2348-4470
p-ISSN (P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 3, Issue 1, January -2016

Survey on approaches to secure SSO mechanism

Nidhi Thacker[1], Prof. Gordhan B. Jethava[2]

[1]*Department of Computer Engineering, PIET, Vadodara*
[2]*Department of Information & Technology, PIET, Vadodara*

Abstract — *Internet and web applications have grown exponentially and have become an essential part of day-to-day living. But level of security that this Internet provides has not grown as fast as the Internet applications. As web applications become more and more widespread, users must handle an increasing number of authentication credentials to establish security contexts with web applications. Single Sign-On Mechanism is the most popular authentication mechanism and is used by most of companies now a days. There are many single sign-on protocols available for implementing it. These single sign-on protocols suffer from an authentication flaw that allows a malicious service provider to impersonate the user. In this survey paper, these types of attacks on single sign-on are explained and also approaches to prevent these attacks are explained with advantages, disadvantages and future scope.*

Keywords- *Single Sign-On, XSS, OpenID, SAML, MTM, Relying Party*

I. INTRODUCTION

With wide spreading of distributed computer networks, it has become popular to allow users accessing various network services offered by distributed service providers [1], [2]. Consequently, user authentication (also called user identification) [3], [4] plays a crucial role in distributed computer networks to verify if a user is legal and then can be granted to access the services requested.

The goal of a single sign on platform is to eliminate individual sign on procedures by centralizing user authentication and identity management at a central identity provider. Therefore, users are relieved from the huge burden of registering many online accounts and remembering many passwords.

Single sign-on (SSO) is an authentication mechanism that uses a single action of authentication to permit an authorized user to access all related, but independent software systems or applications without being prompted to log in again at each of them during a particular session [5]. Once the user login, the SSO system generates authentication information accepted by the various applications and systems. SSO helps in reducing password fatigue from different user name and password combinations and IT costs due to lower number of IT help desk calls about passwords. SSO is used by Google, Facebook and in many commercial firms and educational institutes. There are single sign-on protocols which are used to implement single sign on solution.

As SSO is advantageous still SSO protocols are suffering from authentication flaws by malicious service providers which will impersonate the user. So user privacy is sometimes at risk. To prevent these flaws there are many approaches are available and these approaches is working towards securing single sign-on mechanism.

The rest of this paper is organized as follows. Section II presents basic operation of single sign-on mechanism.. Section III presents possible attacks on SSO. Approaches to secure SSO are discussed in section IV. Section V presents comparison of different approaches for security of SSO. Conclusion is presented in section VI.

II. WORKING OF SSO

Fig. 1 shows basic operation involved in Single Sign-On mechanism. It includes different domains like authentication domain and others are secondary domains. In authentication domain user will send request to access any application to secondary domain. In the authentication domain, SSO application is running. So whenever the use's request is arrived host of the application will refer SSO application for authentication. Then user is asked for credentials and then valid user is authenticated and authentication information is transferred to that application's domain. Host will verify and then give access to user to that application. Now if user wants to access another application which is already registered with SSO application then directly user can access that application. Due to SSO user need not to log in again to that application during particular session.

SSO basically depends on other infrastructures like authentication system, requires interface with web server and identity management or registration. The SSO application maintains a session or Ticket Granting Ticket (TGT) for user. So for another application login, SSO application directly transfers this ticket to that domain. For example Google is providing Google apps service which is based on SAML Single Sign-On. Using the SAML model, Google acts as the service provider and provides services such as Gmail and Start Pages. Google partners act as identity providers and control usernames, passwords and other information used to identify, authenticate and authorize users for web applications that Google hosts [9].

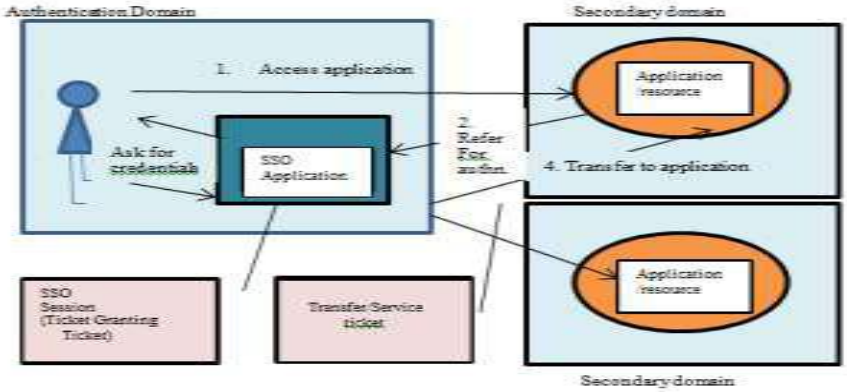

Figure 1. Basic operation of SSO

III. POSSIBLE ATTACKS ON SSO

3.1 Man-in-the-middle attack by malicious relying party

A malicious web site pretends to be benign and initiates the SSO process, i.e., an attacker simply sends a request using the application ID of the benign website to spoof the identity of it. Man-In-The-Middle Attack by a Malicious Relying Party in which a malicious RP could send an authentication response that has been issued for the RP to another RP to impersonate an end user even if TLS is utilized [8].

3.2 Cross-Site Scripting(XSS) attack

In a typical XSS attack, a hacker inject his malicious JavaScript code in the legitimate website. When a user visit the specially crafted link, it will execute the malicious JavaScript. This vulnerability will allow attackers to do phishing attacks, steal accounts and even worms. Access tokens can be stolen on most (91%) of the evaluated RPs, if an adversary could exploit an XSS vulnerability on any page of the RP web site. A Cross-Site Scripting attack that have been identified in the SAML-based SSO for Google Apps and in the SSO available in Novell Access Manager v.3.1 [6].

IV. APPROACHES TO SECURE SSO

Possible approaches to preserve privacy using single sign-on mechanism are divided into two categories:

4.1 Approaches to prevent man-in-the-middle attack by malicious relying party

4.1.1 Establishing a dedicated bidirectional communication channel between Relying Party (RP) and Identity Provider (Idp)

This approach is actually practically relevant in solving a pressing problem in the area of user authentication, privacy, and web security. The life-cycle of a bidirectional, authenticated, secure communication channel is over the following three steps: (i) establishing the channel, (ii) using the channel to communicate securely, and (iii) destroying the channel. In establishing a channel it uses javascript socket to communicate and execute handshake protocol during particular session. After establishing the channel it is used to send and receive messages between client-side RP and client-side Idp. In the process of destroying a channel, socket.close() is called. When either the IdP or RP calls the close method, the other side is notified and must close the channel as well.Using single sign-on mechanism all different RPs will communicate with proxy and proxy will communicate with legacy Idp [10].

4.1.2 On Cryptographically Strong Bindings of SAML Assertions to Transport Layer Security

One approach is to bind the SAML assertion and the SAML artifact to the public key contained in a TLS client certificate. Another approach is to strengthen the Same Origin Policy of the browser by taking into account the security guarantees TLS gives. By binding the SAML assertion to cryptographically derived values of the TLS session that has been agreed upon between client and the service provider, this approach provides anonymity of the browser while allowing Relying Party and Identity Provider to detect the presence of a man-in-the-middle attack. The idea is the following: If the browser adds a value derived from the master secret of a certain TLS session to his SAML Assertion request, and if this derived value is present in the SAML assertion, the RP may deduce that the other endpoint of the current TLS channel (i.e. the browser) requested this assertion from the Idp [11].

4.2 Approaches to prevent Cross Site Scripting (XSS) attack on single sign-on

4.2.1 Applying Pattern Filtering Approach

Various approaches to defend against attacks (that use XSS vulnerabilities) are available today but no single approach solves all the loopholes. To defend against persistent XSS attacks, a simple task has to be performed for input filtering: any data from the input must be transformed or filtered in a way that it is not executed by a browser if sent to it. To avoid XSS developers must sanitize the user's input before storing it in the database.

Figure 2. Conceptual model of the persistent XSS filter [12]

The solution includes:
Filtering Event Handlers: Event handlers are JavaScript codes that are not added inside the <script> tags, but rather inside the HTML tags and when any event occurs, the function that is assigned to an event handler runs.
Filtering Data URI: Data URI is a self-contained link that contains document data and metadata entirely encapsulated in the URI.'data' URI, being entirely self-contained, does not include a filename. The use of some keywords in the user input has been blacklisted in web application - keywords like for instance, JavaScript, alert, script, round brackets, double quotes, and colon.
Filtering insecure keywords: An insecure keyword is in a list of known bad data to block illegal content from being executed. then, will be removed or repla Some of the insecure keywords are document.cookie, document.write, window.location, innerHTML, parent Node <applet, <embed, <script.
Filtering Character Escaping: Preventing XSS attacks means to substitute every special character used in these attacks Method is to escape dangerous characters by using the &# sequence followed by its character code.

4.2.2 Encoding HTML responses

To help prevent XSS attacks, an application needs to ensure that all variable output in a page is encoded before being returned to the end user. Encoding variable output substitutes HTML markup with alternate representations called *entities*. The browser displays the entities but does not run them. For example, <script> gets converted to <script&gt. When a web browser encounters the entities, they will be converted back to HTML and printed but they will not be run [13].

4.2.3 Server-side approach using reverse proxy

This approach is presented to protect users against XSS attacks that offers the same level of protection as previous work done to prevent XSS attack, but without the necessity for client-side modifications. To avoid the disadvantage of involving the enduser, a Web browser is positioned on a reverse proxy and XSS filter before the server.

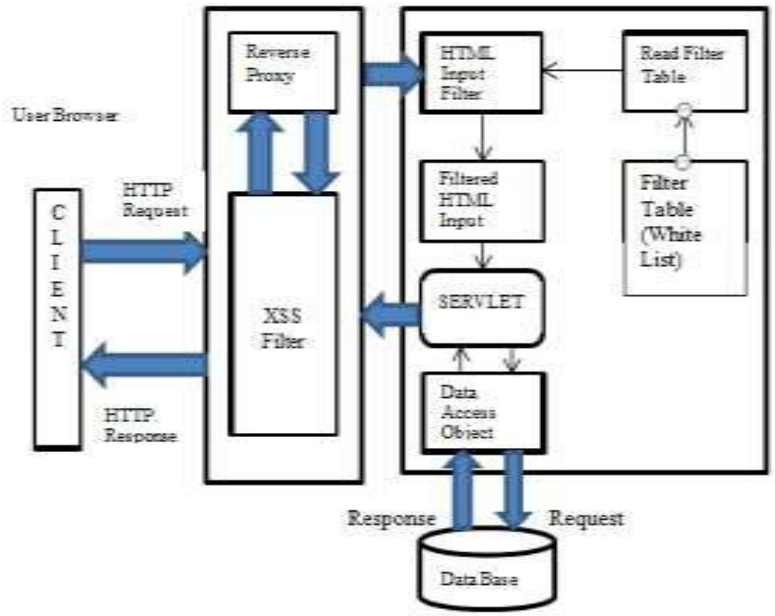

Figure 3. Architecture for preventing Cross-Site Scripting in Server Side [14]

A JavaScript detection component, which, given the Web server's response and request, is capable of determining whether script content is present or not. A reverse proxy installed in front of the Web server, which is used to getting incoming HTTP request parameter from the user and outgoing HTTP response parameter from the server and subjects them to analysis by the JavaScript detection component. A XSS filter component, which is used to clean harmful script from the HTTP request and HTTP response. A HTML Input filter component is located in front of servlet component ,which is used to inspect escape comments, balance HTML tags, remove blanks space, protocol attributes from the incoming HTTP request and encode this parameter. A Data Access Object (DAO) component, by using data access objects instead of accessing the data source directly, the type and implementation of the actual data source is decoupled from its usage. This allows moving from one data source to a different data source without having to change the business logic.

4.2.4 **Noncespaces: Using randomization to defeat cross-site scripting attacks**

Noncespaces, a technique that enables web clients to distinguish between trusted and untrusted content to prevent exploitation of XSS vulnerabilities. Using Noncespaces, a web application randomizes the (X)HTML tags and attributes in each document before delivering it to the client. As long as the attacker is unable to guess the random mapping, the client can distinguish between trusted content created by the web application and untrusted content provided by an attacker. To eliminate the client-server semantic gap and to adapt to differing security needs, the browser enforces a configurable security policy. The policy specifies the browser capabilities that each type of content can exercise. In this way, malicious content injected by an attacker is restricted to the capabilities allowed to untrusted content by the policy.

Figure 4. Noncespaces overview [15]

The server delivers an (X)HTML document annotated with trust class information and a policy to the client. The client accepts the document only if it satisfies the policy. Noncespaces is simple. The server need not sanitize any untrusted content. This avoids all the difficulties and problems with sanitization.

4.2.5 BIXSAN: Browser Independent XSS Sanitizer

The existing solutions to XSS attack include use of regular expressions to detect the presence of dynamic content and client side filtering mechanisms such as Noscript and Noxes tool. The drawbacks of these solutions are low fidelity and disallowing of benign HTML. In order to overcome these drawbacks BIXSAN, a Browser Independent XSS SANitizer for prevention of XSS attacks is proposed. A Browser Independent XSS SANitizer. BIXSAN comprises of JavaScript Tester (modified browser) to detect the presence of JavaScript, a Parse tree generator to avoid the anomalous behaviour of the browser, identification of static tags for allowing benign HTML, a complete HTML parser. BIXSAN was tested on all the popular browsers, viz., IE, Opera, Firefox and Netscape navigator. From the experiments conducted it was found to reduce the anomalous behaviour of browser [16].

4.2.6 Complementary Character Coding Approach

A new approach to character level dynamic tainting which allows efficient and precise taint propagation across the boundaries of server components, and also between servers and clients over HTTP. In this approach, each character has two encodings, which can be used to distinguish trusted and untrusted data. Notably, it offers a precise protection against persistent cross-site scripting attacks, as taint information is maintained when data is passed to a database and later retrieved by the application program. In this approach, two encodings are used for each character, standard characters and complement characters. Untrusted data coming from users is encoded with complement characters, while trusted developer code is encoded with standard characters. Components are modified to enforce security policies, which are characterized by sets of allowed tokens, for which user input characters should not be permitted. Each complement aware component enforces its policy by using full comparison to match sensitive tokens during parsing. Elsewhere they use value comparison to preserve functionality [17].

4.2.7 SWAP (Secure Web Application Proxy)

SWAP (**S**ecure **W**eb **A**pplication **P**roxy), a server-side solution for detecting and preventing cross-site scripting attacks. SWAP comprises a reverse proxy that intercepts all HTML responses, as well as a modified Web browser which is utilized to detect script content. SWAP can be deployed transparently for the client, and requires only a simple automated transformation of the original Web application. SWAP operates on a reverse proxy, which relays all traffic between the Web server that should be protected and its visitors. The proxy forwards each Web response, before sending it back to the client browser, to a JavaScript detection component, in order to identify embedded JavaScript content. In the JavaScript detection component, SWAP puts to work a fully functional, modified web browser, that notifies the proxy of whether any scripts are contained in the inspected content. If no scripts are found, the proxy decodes all script IDs, effectively restoring all legitimate scripts, and delivers the response to the client. If the JavaScript detection component, on the other hand, detects a script, SWAP refrains from delivering the response, but instead notifies the client of the attempted XSS attack. The proxy prevents each malicious response from being delivered to the client, and thus effectively inhibits the attack to be carried out on the client's browser [18].

V. COMPARISON OF APPROACHES

Approach	Advantage	Disadvantage	Future Scope
Dedicated bidirectional communication channel between RP & Idp	Authentication latency is less. It is safe from malicious RP.	With an overall small latency overhead of about 650ms, and a clear bottleneck in this prototype implementation in the generation of the private and public key is there.	Overhead of communication channel is still not considered. This approach can be used without the need of SSL.

Bindings of SAML Assertions	This provides anonymity of the browser so RP and Idp can know about man-in the middle attack.	This approach assumes that some RP process only self-signed certificates.	It can be applied to email protection or secure HTTP cookies.
Pattern filtering approach	Filtering of insecure keywords, event handlers, URIs, etc are done in this method.	Proper escaping mechanisms should be used at the right places.	This approach can also be applied to non-persistent and other types of XSS attack.
Encoding HTML responses	Web browser itself will identify entities and convert to HTML and then will not allow to run.	This approach is limited to some specific entities.	More markups can be added and make more generalized.
Reverse Proxy	This approach does not involve client side modifications.	Reverse proxy should be placed before XSS Filter and should be maintained.	This proxy can be provided in-built with the system.
Noncespaces	Policy is used to restrict content injected by attacker.	Specific policy for injected contents should be described.	This approach can be improved using other randomization techniques.
BIXSAN	This provides solution with high fidelity and allows benign HTML to run.	This approach is limited to some web browsers.	This approach can be used with Google chrome web browser.
Complementary Character Coding	Taint information is maintained when data is passed to a database then can be retrieved back.	2% of worst case overhead is there.	This solution can be extended to use complementary Unicode.
SWAP	SWAP can be deployed transparently for the client, and requires only a simple automated transformation of the original Web application.	The solution is limited to JavaScript.	SWAP can be used in current browsers and SWAP performance can be improved.

VI. CONCLUSION

Single sign-on is an authentication mechanism and the goal of a single sign on platform is to eliminate individual sign on procedures by centralizing user authentication and identity management at a central identity provider. Due to increasing use of single sign-on mechanism in most companies, more and more research work is done by different researchers in many different directions. From last many years, research on prevention of authentication flaws on single sign-on is done. The different approaches for preventing security attacks on single sign-on are discussed in this work. A brief comparison of prevention approaches with its advantages, disadvantages and future scope are explained.

REFERENCES

[1] A. C. Weaver and M. W. Condtry."Distributing Internet services to the network's edge." IEEE Trans. Ind. Electron., 50(3): 404-411, Jun. 2003.
[2] L. Barolli and F. Xhafa. "JXTA-OVERLAY: A P2P platform for distributed, collaborative and ubiquitous computing." IEEE Trans. Ind. Electron., 58(6): 2163-2172, Oct. 2010.

[3] L. Lamport,. "Password authentication with insecure communication. Commun.", ACM, 24(11): 770-772, Nov. 1981.
[4] W. B. Lee and C. C. Chang."User identification and key distribution maintaining anonymity for distributed computer networks". Computer Systems Science and Engineering, 15(4): 113-116, 2000.
[5] V. Radhaa, D. Hitha Reddya. "A Survey on Single Sign-On Techniques", Procedia Technology 4 (2012 134 – 139 2212-0173 © 2012 Published by Elsevier Ltd. doi: 10.1016/j.protcy.2012.05.019 C3IT-2012
[6] Alessandro Armando, Roberto Carbone, Luca Compagna Jorge Cuellar,Giancarlo Pellegrino, and Alessandro Sorniotti, " From Multiple Credentials to Browser-based Single Sign-On: Are We More Secure?", Springer(2011)
[7] David Orrell, Eduserv Athens EuroCAMP. "Authentication Systems and Single Sign-On (SSO)", 7-9 November 2005, Porto, Portugal
[8] Yuto Iso Graduate School of Meiji University, Japan, Takamichi Saito Meiji University, Japan, " A Proposal and Implementation of an ID Federation That Conceals a Web Service from an Authentication Server", (2015 IEEE 29th International Conference on Advanced Information Networking and Applications)
[9] Creative Commons Attribution 3.0 License. "SAML Single Sign-On (SSO) Service for Google Apps", Last updated January 6, 2015 https://developers.google.com/googleapps/sso/saml_reference_im plementation?hl=en,
[10] Yinzhi Cao, Yan Shoshitaishvili, Kevin Borgolte, Christopher Kruegel, Giovanni Vigna, and Yan Chen, "Protecting Web-Based Single Sign-on Protocols against Relying Party Impersonation Attacks through a Dedicated Bi-directional Authenticated Secure Channel", A. Stavrou et al. (Eds.): RAID 2014, LNCS 8688, pp. 276–298, 2014, Springer International Publishing Switzerland 2014
[11] Florian Kohlar, J¨org Schwenk, Meiko Jensen, Sebastian Gajek. "On Cryptographically Strong Bindings of SAML Assertions to Transport Layer Security", Volume 3, Issue 4. Copyright © 2011. 16 pages, International Journal of Mobile Computing and Multimedia Communications (IJMCMC)
[12] Imran Yusof, Al-Sakib Khan Pathan, "Preventing Persistent Cross-Site Scripting (XSS) Attack By Applying Pattern Filtering Approach", (2014)IEEE
[13] Usha Ladkani." Prevent cross-site scripting attacks by encoding HTML responses", IBM corporation 2013,30 July 2013
[14] A. Duraisamy, M. Sathiyamoorthy, S. Chandrasekar. "A Server Side Solution for Protection of Web Applications from Cross-Site Scripting Attacks", International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278 - 3075, Volume-2, Issue-4, March 2013
[15] Matthew Van Gundy, Hao Chen. "Noncespaces: Using randomization to defeat cross-site scripting attacks", computers & security 31(2012) 612e628, www.sciencedirect.com
[16] Sharath Chandra V. and S. Selvakumar." BIXSAN: Browser Independent XSS Sanitizer for prevention of XSS attacks" ACM SIGSOFT Software Engineering Notes, September 2011 Volume 36 Number 5,DOI: 10.1145/2020976.202099 http://doi.acm.org/10.1145/2020976.2020996
[17] Raymond Mui and Phyllis Frankl, "Preventing Web Application Injections with Complementary Character Coding", V. Atluri and C. Diaz (Eds.): ESORICS 2011, LNCS 6879, pp. 80–99, 2011.Springer-Verlag Berlin Heidelberg 2011
[18] Peter Wurzinger, Christian Platzer, Christian Ludl, Engin Kirda, and Christopher Kruegel, "SWAP: Mitigating XSS Attacks using a Reverse Proxy", *SESS'09*, May 19, 2009, Vancouver, Canada 978-1-4244-3725-2/09/$25.00 © 2009 IEEE

