ISSN:0975-9646

A Survey on Web Authentication Methods for Web Applications

Ch.Jhansi Rani [1]
Asst. Professor/Dept of CSE
KKR&KSR Institute of Technology & Sciences
Guntur,A.P

SK.Shammi Munnisa [2]
Asst. Professor/Dept of CSE
KKR&KSR Institute of Technology & Sciences
Guntur,A.P

Abstract-One of the common problems facing our web application user is the thorny issue of trust and security. The vast majority of consumers are concerned about the safety of their credit card and personal details. People simply don't trust the Web, fearing that their transactions might not be safe. Not only are consumers concerned, the prospect of online credit card fraud also has an adverse effect on potential online shoppers. Increased trust in the safety of online dealings has numerous benefits, of which increased revenue and profitability is the most important. There are real challenges– and significant opportunities – for e- tailers like you to deliver the same level of trust and personalization over the Internet as offered by real shops. This paper presents the critical role of authentication for web application and different types of authentication methods as well as the business benefits which flow from creating trust on the web application.

Keywords: role of Authentication, Authentication in web applications

1. INTRODUCTION

Web browsers can connect to Web Logic Server over either a Hyper Text Transfer Protocol (HTTP) port or an HTTP with SSL (HTTPS) port. The benefits of using an HTTPS port versus an HTTP port are two-fold. With HTTPS connections:

i. All communication on the network between the Web browser and the server is encrypted. None of the communication, including the user name and password, is in clear text.
ii. As a minimum authentication requirement, the server is required to present a digital certificate to the Web browser client to prove its identity.

If the server is configured for two-way SSL authentication, both the server and client are required to present a digital certificate to each other to prove their identity.

2. WEB AUTHENTICATION METHODS:

HYPER TEXT TRANSFER PROTOCOL AUTHENTICATION

HTTP authentication is a method for the client to provide a username and a password when making a request.

This is the simplest possible way to enforce access control as it doesn't require cookies, sessions or anything else. To use this, the client has to send the Authorization header along with every request it makes. The username and password are not encrypted, but constructed this way

i. username and password are concatenated into a single string: username: password
ii. this string is encoded with Base64
iii. the Basic keyword is put before this encoded value

Example for a user named rani with password sec

The drawbacks of using HTTP Basic authentication

i. The username and password are sent with every request, potentially exposing them - even if sent via a secure connection
ii. connected to SSL/TLS, if a website uses weak encryption, or an attacker can break it, the usernames and passwords will be exposed immediately
iii. there is no way to log out the user using Basic auth
iv. Expiration of credentials is not trivial - you have to ask the user to change password to do so

USER NAME AND PASSWORD AUTHENTICATION

Web Logic Server performs user name and password authentication when users use a Web browser to connect to the server via the HTTP port. In this scenario, the browser and an instance of Web Logic Server interact in the following manner to authenticate a user

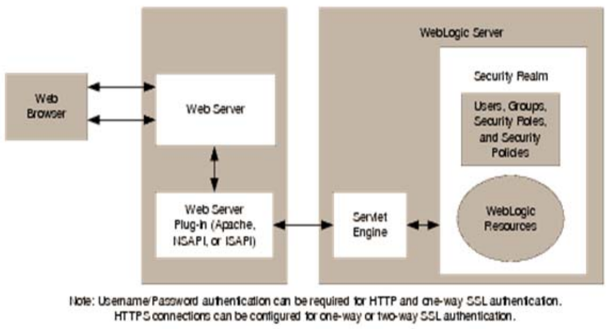
1. A user invokes a Web Logic resource in Web Logic Server by entering the URL for that resource in a Web browser. The http URL contains the HTTP listen port, for example, http://myserver:7001.

   The Web server in Web Logic Server receives the request.
2. The Web server determines whether the Web Logic resource is protected by a security policy. If the Web Logic resource is protected, the Web server uses the established HTTP connection to request a user name and password from the user.
3. When the user’s Web browser receives the request from the Web server, it prompts the user for a user name and password.
4. The Web browser sends the request to the Web server again, along with the user name and password.
5. The Web server forwards the request to the Web server plug-in. Web Logic Server provides the following plug-ins for Web servers:
   i. Apache-Web Logic Server plug-in
   ii. Netscape Server Application Programming Interface (NSAPI)
   iii. Internet Information Server Application Programming Interface (ISAPI)
   iv. The Web server plug-in performs authentication by sending the request, via the HTTP protocol, to

Web Logic Server, along with the authentication data (user name and password) received from the user.

6. Upon successful authentication, Web Logic Server proceeds to determine whether the user is authorized to access the Web Logic resource.
7. Before invoking a method on the Web Logic resource, the Web Logic Server instance performs a security authorization check. During this check, the server security extracts the user's credentials from the security context, determines the user's security role, compares the user's security role to the security policy for the requested WebLogic resource, and verifies that the user is authorized to invoke the method on the WebLogic resource.
8. If authorization succeeds, the server fulfills the request.

Figure 1 Secure Login for Web Browsers


WebLogic Server
Web Browser
Web Server
Security Realm
Users, Groups, Security Roles, and Security Policies
Web Server Plug-In (Apache, NSAPI, or ISAPI)
Servlet Engine
WebLogic Resources
Note: Username/Password authentication can be required for HTTP and one-way SSL authentication.
HTTPS connections can be configured for one-way or two-way SSL authentication.

The drawbacks of using username/password Authentication:

i. Password data stores as a weak point
ii. Passwords should be reset early and often
iii. Good passwords can be hard to remember

COOKIES

When a server receives an HTTP request in the response, it can send a Set-Cookie header. The browser puts it into a cookie jar, and the cookie will be sent along with every request made to the same origin in the Cookie HTTP header.

To use cookies for authentication purposes, there are a few key principles that one must follow.

Always use Http Only cookies

To mitigate the possibility of XSS attacks always use the Http Only flag when setting cookies. This way they won't show up in document cookies.

Always use signed cookies

With signed cookies, a server can tell if a cookie was modified by the client.

The drawbacks of using Cookie Authentication:

i. Need to make extra effort to mitigate CSRF attacks
ii. Incompatibility with REST - as it introduces a state into a stateless protocol

TOKENS

Nowadays JWT (JSON Web Token) is everywhere - still it is worth taking a look on potential security issues.

First let's see what JWT is!

JWT consists of three parts:

i. Header, containing the type of the token and the hashing algorithm
ii. Payload, containing the claims
iii. Signature, which can be calculated as follows if you chose HMAC SHA256: HMACSHA256(base64UrlEncode(header) + "." + base64UrlEncode(payload), secret)
iv. The drawbacks of using Tokens Authentication:
v. Need to make extra effort to mitigate XSS attacks

DIGITAL SIGNATURES

Either using cookies or tokens, if the transport layer for whatever reason gets exposed your credentials are easy to access - and with a token or cookie the attacker can act like the real user.

A possible way to solve this - at least when we are talking about APIs and not the browser is to sign each request. How does that work?

When a consumer of an API makes a request it has to sign it, meaning it has to create a hash from the entire request using a private key. For that hash calculation you may use:

i. HTTP method
ii. Path of the request
iii. HTTP headers
iv. Checksum of the HTTP payload
v. and a private key to create the hash

To make it work, both the consumer of the API and the provider have to have the same private key. Once you have the signature, you have to add it to the request, either in query strings or HTTP headers. Also, a date should be added as well, so you can define an expiration date.

The drawback of using Signature Authentication:

i. cannot use in the browser / client, only between API

ONE-TIME PASSWORDS

One-Time passwords algorithms generate a one-time password with a shared secret and either the current time or a counter:

i. Time-based One-time Password Algorithm, based on the current time,
ii. HMAC-based One-time Password Algorithm, based on a counter.

These methods are used in applications that leverage two-factor authentication: a user enters the username and password then both the server and the client generates a one-time password .implementing this using not p is relatively easy.

The drawbacks of using One Time Password Authentication:

i. with the shared-secret (if stolen) user tokens can be emulated
ii. because clients can be stolen / go wrong every real-time application have methods to bypass this, like an email reset that adds additional attack vectors to the application

CONCLUSION

To achieve this confidentiality, authentication of web is necessary. Many schemes proposed on authentication and some of the significant ones are discussed in this paper. While most authentication schemes are focus only on the security while other major compelling challenges for authentication is to provide proper scalability. It is therefore essential from the literature that an effective implicit authentication method which minimizes the computation cost and verification and eliminates the need of shared key if you have to support a web application only, either cookies or tokens are fine - for cookies think about XSRF, for JWT take care of XSS.If you have to support both a web application and a mobile client, go with an API that supports token-based authentication. If you are building APIs that communicate with each other, go with request signing.

REFERENCES

[1] Mudassar Raza, Muhammad Iqbal, Muhammad Sharif and Waqas Haider, "A Survey of Password Attacks and Comparative Analysis on Methods for Secure Authentication", World Applied Sciences Journal, vol. 19, pp. 439-444, Jan. 2012.

[2] Web Application Security Statistics, "http://projects.webappsec.org/w/page/13246989/WebApplication SecurityStatistics."

[3] C. Onwubiko and A. P. Lenaghan, "Managing Security Threats and Vulnerabilities for Small to Medium Enterprises", in Proc. IEEE Intellgienc and Security Informatics, 2007, p. 244-249.

[4] Italo Dacosta, Mustaque Ahamad, Patrick Traynor, "Trust No One Else: Detecting MITM Attacks Against SSL/TLS Without Third Parties",in Proc. 17th European Symposium on Research in Computer Security, Italy, 2012, p. 10-12..

[5] Syverson, P, "A Taxonomy of Replay Attacks", in Proc. CSFW7 '94, 1994, p. 187-191..

[6] MySpace Samy Worm, "http://namb.la/popular/tech.html," 2005.

[7] Carlisle Adams, Guy-Vincent Jourdan, Jean-Pierre Levac and François Prevost, "Lightweight protection against brute force log in attacks on web applications ", in Proc. PST '10, 2010, p. 181-188.

[8] Elie Bursztein, Matthieu Martin, John C. M, "Text-based CAPTCHA Strengths and Weaknesses", in Proc. CSS '11, 2011, p. 125-138.

[9] Junghyun Nam, Kim-Kwang Raymond Choo, Juryon Paik, Dongho Won, "An Offline Dictionary Attack against a Three-Party Key Exchange Protocol", IEEE Communication Lett., Vol. 13, pp. 205-207, Mar. 2009.

[10] Adrian J Duncan, Sadie Creese, Michael Goldsmith, "Insider Attacks in Cloud Computing", in Proc. TrustCom '12, 2012, p. 857-862.

[11] N. Jovanovic, E. Kirda, and C. Kruegel, "Preventing cross site request forgery attacks," in SecureComm'06: 2nd International Conference on Security and Privacy in Communication Networks, 2006, pp. 1 –10.

[12] A. Jesudoss and N.P. Subramaniam, "A Taxonomy of Authentication Techniques for Web Services", International Journal ofEngineering Research and Technology, Vol. 3 2014, pp. 271-275.

[13] Z. Mao, N. Li, and I. Molloy, "Defeating cross-site request forgery attacks with browser-enforced authenticity protection," in FC'09: 13 th International Conference on Financial Cryptography and Data Security, 2009, pp. 238–255.

[14] Chun-Ying Huang, Shang-PinMa, Kuan-TaChen, "Using one-time passwords to prevent password phishing attacks", Journal of Computer and Network Applications, Vol. 34, Issue 4, pp. 1292-1301, Jul. 2011.

[15] J. Weinberger, P. Saxena, D. Akhawe, M. Finifter, R. Shin, and D. Song, "A Systematic Analysis of XSS Sanitization in Web Application Frameworks," in ESORICS'11: Proc. of 16th European Symposium on Research in Computer Security, 2011.