Securing WEB API Based on Token Authentication

[1]Pravat Sahoo, [2]Nilesh Kumar Janghel, [3]Debabrata Samanta

[1]Software Engineer, Thought Focus
[2,3]Dayananda Sagar College of Arts, Science and Commerce, Bangalore, India

Abstract— An application program interface (API) is an arrangement of schedules, conventions, and devices for building programming applications. Fundamentally, an API determines how programming segments ought to collaborate. Furthermore, APIs are utilized when programming graphical UI (GUI) parts. A decent API makes it less demanding to build up a program by giving all the building pieces. A software engineer at that point assembles the pieces. In token based authentication user first send his user-ID and password to Authorization Server Using client application. If user is existing in database then authorization Server send one token to client. Client use this token for next request. Every time client is validated based on token not his login credential. So, system not access database every time.

Index Terms—Token Authentication, API, Server.

I. INTRODUCTION

The token-based authentication system is to let users to input their credential in order to receive a token which permits them to access a definite resource - without using their credential. Once their token has been attained, the user can offer the token - which bids access to a specific resource for a time period - to the remote site. In alternative words it adds one level of indirection for authentication -- rather than having to certify with username and password for every protected resource, the user authenticates that means once (within a session of restricted duration), obtains a time-limited token reciprocally, and uses that token for additional authentication throughout the session. "Web Services" is the name of an arrangement of models and instruments for summoning programming parts remotely. All correspondence between the part and any applications that conjure it is as ASCII documents containing XML and sent through a standard convention, for example, HTTP. The segment would thus be able to be summoned by applications from anyplace on the Web (or intranet), with no requirement for them to know any subtle elements of how the part is conveyed. Authentication protects any system from anonymous user. It's confirms user identity. This process helps Compare user login information with existing database. Sometime user can authenticate By Active directory (AD). Authentication establishes authorship of system. Authentication starts when user tries to get the information from some system. First, user must valid his access credential. When user tries to login any system user must enter his credential, which is must be provide each user. However, this type of authentication can be bypass by hackers. Authentication doesn't stop to user access particular resource, this is done by authorization.

II. ADVANTAGES

A. Cross-domain / CORS

Cookies + CORS don't play well across dissimilar domains. A token-based tactic enables one to make AJAX calls to any server, on any domain because one uses an HTTP header to convey the user information.

B. Stateless (a.k.a. Server-side scalability)

It is not at all required to keep a session store, the token is a self-reliant unit that sends all the user info. The remining stays in cookies or local storage on the user side.

C. Decoupling

One is not bond to any specific validation scheme. The token might be produced anywhere, hence your API can be accessed from anywhere with a solitary way of validating those calls.

D. Extensibility (Friend of A Friend and Permissions)

It allows one to create applications and allows to share permission to others i.e. it provides careful way of giving permission to the third-party applications.

E. CSRF

As one is not depending on cookies, one doesn't require defending against cross site requests. It would not be attainable to sib your web site, generate a POST request and re-use the prevailing verification cookie and again using the existing verification cookie as there is none.

III. BASIC ARCHITECTURE

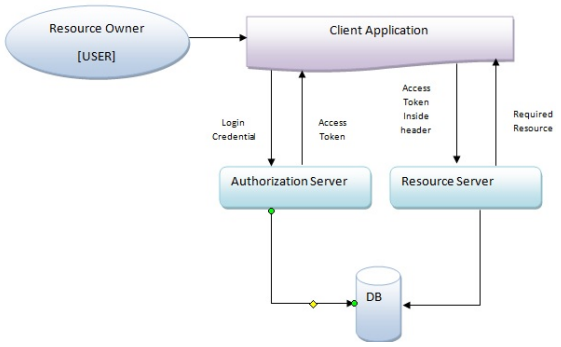

Figure 1: Basic Architecture of Token Based Authentication

User: who use system and who is authenticating by token.

Client: User is communicating to Authorization Server by client app.

Authorization Server: This server job is like cop, this server authenticate user using DB and generate Token for client.

Resource Server: This is server doing main functionality job of application. If client have proper token this server response
based on client request.

Database: This store user credential and other data.

IV. COOKIE BASED AUTHENTICATION

Cookie based authentication is old approach. In this approach we need to maintain state. The record of authentication must be store in server side and client side. Servers keep tracking active session in DB and client side created session identifier.

Flow of cookie based authentication:

- User enters his credential.
- Server verifies the credentials and store in DB.
- Create a cookie with new session ID in browser.
- For subsequent requests, the session ID cross verified again if it's correct proceed the request.
- Once user Logout session destroyed in both side.

V. STEPS

- User trying to access any system.
- User input his accreditation.
- Authenticate process compare user input data with existing record.
- If user is valid its given permission to user for access system

VI. OPENID CONNECT

OAuth 2.0 was composed to permit a wide range of arrangements, however by configuration does not determine how these organizations come to be set up or how the parts think about each other. This is alright in the general OAuth world where one approval server secures a particular Programming interface, and the two are firmly coupled. With OpenID Associate, a typical secured Programming interface is conveyed over a wide assortment of customers and suppliers, all of which need to think about each other to work. It would not be adaptable for every customer to need to know early about every supplier, and it would be significantly more unscalable to require every supplier to think about every potential customer.

OpenID Connect also uses the JSON Object Signing Encryption (JOSE).

OpenID Interface is manufactured specifically on OAuth 2.0 and as a rule is conveyed appropriate alongside (or over) an OAuth foundation.

The OpenID Interface ID Token is a marked JSON Web Token (JWT) that is given to the user application close by the general OAuth get to token. The ID Token contains an arrangement of cases about the verification session, including an identifier for the client , the identifier for the personality supplier who issued the token , and the identifier of the user for which this token was made . Moreover, the ID Token contains data about the token's legitimate lifetime and in addition any data about the verification setting to be passed on to the user, for example, to what extent back the client was given an essential validation system. Since the organization of the ID Token is known by the user, it can parse the substance of the token specifically and acquire this data without depending on an outside support of do as such. By applying a couple of straightforward checks to this ID token, a user can shield itself from a substantial number of basic assaults.

VII. SAMPLE CODE

After you install this bundle “Install-Package Microsoft.Owin.Security.OAuth -Version 2.1.0"

open cs file"Startup" and call the new technique function "ConfigureOAuth" as the principal line inside the strategy "Configuration", the implemntation for this technique as beneath:

```
public class Startup
  {
    public void Configuration(IAppBuilder app)
    {
      ConfigureOAuth(app);
          //Rest of code is here;
    }
    public void ConfigureOAuth(IAppBuilder app)
    {
      OAuthAuthorizationServerOptions
OAuthServerOptions = new
OAuthAuthorizationServerOptions()
      {
        AllowInsecureHttp = true,
        TokenEndpointPath = new
PathString("/token"),
        AccessTokenExpireTimeSpan =
TimeSpan.FromDays(1),
        Provider = new
SimpleAuthorizationServerProvider()
      };

      // Token Generation

app.UseOAuthAuthorizationServer(OAuthServerOption
s);
      app.UseOAuthBearerAuthentication(new
OAuthBearerAuthenticationOptions());
    }
  }
```

We've determined the expiry for token to be 24 hours, so if the client endeavored to utilize a similar token for validation following 24 hours from the issue time, his demand will be rejected and HTTP status code 401 is returned.

VIII. CONCLUSION

When you use "authentication token", the easy presentation of that token by the client grants access. If we store the tokens "as is" in our server's info, then hacker might get a glimpse at your info. A validation token isn't a fixed password, it is an arbitrary value which was created and recollected by a system, with no human cerebrum engaged with the procedure. In that capacity, in the event that you created the token legitimately (no less than 128 bits, got from a cryptographic-ally secure PRNG), at that point there is no requirement for salts, simply utilize a plain hash technique (even MD5 would be fine there). This will be more effective.

REFERENCES

[1] D. Benslimane, S. Dustdar, A. Sheth, "Services Mashups: The New Generation of Web Applications", IEEE Internet Computing, vol. 10, pp. 13-15, 2008.

[2] T. Dierks, The Transport Layer Security (TLS) Protocol Version 1.2 Internet RFC 5246, August 2008.

[3] H. Krawczyk, M. Bellare, R. Canetti, HMAC: Keyed-Hashing for Message Authentication Internet RFC 2104, February 1997

[4] SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, NIST Computer Security Division (CSD), 2014.

[5] P. W. Handayani, A. N. Hidayanto, I. Budi, "Business process requirements for Indonesian Small Medium Enterprises (SMEs) in implementing Enterprise Resource Planning (ERP)", International Journal of Innovation Management and Technology, vol. 4, no. 1, pp. 93-97, February 2013.

[6] A. A. Rumanti, K. J. Syauta, "Determining strategies based on strategic position analysis in Small and Medium Enterprises", International Journal of Information and Education Technology, vol. 3, no. 4, pp. 442-447, August 2013.

[7] T. Shiroma, T. Nagata, Y. Taniguchi, M. Nakamura, S. Tamaki, "Extension of openID connect for utilizing attributes", INFORMATION, vol. 19, no. 2, pp. 705-715, 2016.

[8] R. Fielding, J. Reschke, "Hypertext Transfer Protocol (HTTP/1. 1): Semantics and Content", IETF, June 2014.

[9] Ien Liao, Chengchi Lee, Minshiang Hwang, "A password authentication scheme over insecure networks", Journal of Computer and System Sciences, vol.72(4), pp.727-740, 2006.

[10] Mousumi Paul , Debabrata Samanta, and Goutam Sanyal," Dynamic job Scheduling in Cloud Computing based on horizontal load balancing",International Journal of Computer Technology and Applications (IJCTA) , Vol. 2 (5), pp. 1552-1556, 2011, ISSN: 2229-6093

[11] Syed K Ahmed Khadri, D Samanta, and Mousumi Paul, "Approach of Message Communication Using Fibonacci Series: In Cryptology," Lecture Notes on Information Theory, Vol. 2, No. 2, pp. 168-171, June 2014.

[12] Syed K Ahmed Khadri, D Samanta, Mousumi Paul," Secure Approach for Message Communication", International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE), pp. 3481-3484, Vol. 2, Issue 9, September 2013.

[13] Syed K Ahmed Khadri, D Samanta, Mousumi Paul," Novel Approach for Message Security", International Journal of Information Science and Intelligent System (IJISIS), pp. 47-52,Volume 3, Number 1, 2014.

[14] P.J. Guo, T. Zimmermann, N. Nagappan, B. Murphy, "Characterizing and predicting which bugs get fixed: an empirical study of Microsoft Windows", Proceedings of the 32nd International Conference on Software Engineering, pp. 495-504, 2010.

[15] Syed K Ahmed Khadri, D Samanta, Mousumi Paul," Message Encryption Using Text Inversion plus N Count: In Cryptology", International Journal of Information Science and Intelligent System (IJISIS), pp. 71-74, Volume 3, Number 2, 2014.

[16] M. Bellare, D. Pointchecal, and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," Proceedings of Advances in Cryptology-2000, pp. 139-155, 2000.

[17] W. Diffie and M. Hellman, "New directions in cryptography," IEEE Transactions on Information Theory, Vol. 22, pp. 644-654, 1976.S. Chen, B. Mulgrew, and P. M. Grant, "A clustering technique for digital communications channel equalization using radial basis function networks," IEEE Trans. on Neural Networks, vol. 4, pp. 570-578, July 1993.

[18] J. U. Duncombe, "Infrared navigation—Part I: An assessment of feasibility," IEEE Trans. Electron Devices, vol. ED-11, pp. 34-39, Jan. 1959.

[19] C. Y. Lin, M. Wu, J. A. Bloom, I. J. Cox, and M. Miller, "Rotation, scale, and translation resilient public watermarking for images," IEEE Trans. Image Process., vol. 10, no. 5, pp. 767-782, May 2001.