A Protocol for Service to Service Authentication using Token Challenge Response

Hiresha Kewalramani  
Pace University  
One Pace Plaza, NY, 10038, USA  
kewalramani.hiresha@pace.edu

Nader Nassar  
Pace university  
Pleasantville, NY,10570, USA  
nnassar@pace.edu

Abstract - Many approaches have been created to ease out the connection between users and servers using SSO, which utilizes OAuth and OpenID Connect. Here, two different servers communicate with each other to provide service to the user without having to create multiple accounts. Ex. Google and Facebook accounts are used to authenticate the users for the use of third-party services. In this paper we have proposed a new method for services to securely communicate by authorizing and authenticating one another by utilizing handshake methodology for requesting service, sending tokens and receiving the service after authentication of the tokens.

Keywords- authentication, authorization, token, pseudo-random numbers

I. BACKGROUND

Two-factor authentication is widely used technology with every device and service to protect its users from different types of attacks from hackers by providing users a second layer of security to their devices, accounts and sensitive information from being mishandled. As said there are pros and cons for every action. In the same way, even 2FA has its cons with respect to the existing technology. Due to obvious reasons smartcards, biometrics, and hardware tokens cannot be used for service to service authentication as there is no process for a service to read a physical token or scan biometrics to authenticate and authorize another server. Therefore, we are limited to using Type 1 (something you know) authentication process but by using a challenge response methodology for authenticating services by providing them authorized access for the requested service.

II. PROPOSED PROTOCOL

The goal of our research is based on bundling authorization and authentication in one protocol that could be easily adopted in service to service communication or within a micro-services ecosystem. Our approach is using the commonly known challenge response methodology that is a defacto in manual user authentication. Except in our proposal, we are using a set of expired tokens that have been previously communicated between the two services. The service provider (SP) will authenticate the service requester (SR) only when the provided tokens were valid and received in the correct order proves to be authentic. If, the provided token is not validated or received in an incorrect order by the SP that will mean either the service requester is malicious or there had been some error in the service while providing the token due to an attack or vulnerability. Thus the authentication process will fail.

The architecture diagram is based on two main components that communicate among themselves to request or receive service from each other. These two components are Service Requester and Service Provider described below in Figure1.

Figure1. High-Level Architecture Diagram

A. Requester Side Token Management

- Service Requester (SR) Server communicates with the SP server to establish a connection to receive the service from the provider. It also communicates with its Token Factory database when requested to fetch a token.
- Token Management Engine is responsible for generating new tokens and providing expired tokens as requested by the service provider. This engine is capable of storing or generating both valid and expired tokens based on the token identifier. It consists of:
  - New Seed - It is the seed that is received from the Service Provider when the Service Requester request for the service. This new seed is fed to PRNG to generate a pool of random tokens.
  - PRNG – Pseudo-Random Number Generator, generates a pool of random strings using the received new seed for the connection for the SP from whom the service is requested.
  - Valid/Time-stamped Token Pool – The pool of strings are further time-stamped and marked as valid and stored in the Token Factory
  - Expired Token Pool – This pool contains tokens which have been previously used for earlier connections made while connecting with the SP. Expired tokens are used as a method of authenticating the SR to SP during the challenge response.
  - Expired Token Request Verifier – Verifier verifies the token that SR provides to the SP asked during the challenge response. If the provided token is verified as a valid token by the SP, then SR is authenticated to receive service.
• Token Factory – It stores all the valid/time stamped tokens and expired tokens and provides them to the Token Management Engine when required.

B. Request Governance
- Service Provider Ingress/Egress is responsible for communication with the Service Requester. It also communicates with the Seed Factory to generate the proper challenge for the requester.
- Request Governance Engine
  - Generates New Seed for the requester – It generates a new seed when the SR requests for the service. This seed is then, sent to the SR where it is fed into the PRNG to generate a string of tokens.
  - Validate Token Requester – The validated time-stamped tokens in the Token Management Engine are then sent to the Validate Requester token in Request Governance Engine for validation, that tokens have been received from the same SR who has requested for the service.
  - Generates challenges for the Requester – During the connection request, it generates challenges for the requester (based on asking for expired tokens).
  - Validate Requester Expired Tokens – Once the SR answers the challenges by providing correct expired token asked by the SP, then answer is validated by the Validate Requester Expired Token.
- Seed Factory – It stores the new seed and expired token for each service requester(s) who request for service from the SP.

III. IMPLEMENTATION

A. TERMINOLOGY

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>SR</td>
<td>Service Requester</td>
</tr>
<tr>
<td>SP</td>
<td>Service Provider</td>
</tr>
</tbody>
</table>

In this paper, we are covering two scenarios only: (1) Initialization Phase and (2) Steady Phase

B. INITIALIZATION PHASE

According to scenario (1) for the services to communicate the Service Requester (SR) will have to request the service from the Service Provider (SP). On the initial connection request from SR, the SP will generate a seed ‘S’ , where S will be a set of seeds

\[ S \Rightarrow \{ s_1, s_2, s_3, ..., s_n \} \]  

and through a seed a set of random pool of timestamped tokens will be generated (timestamped tokens range from \[ T(t_1, t_2, t_3, ..., t_n) \]), which are a function of seed) for a particular SR and stored in its database. Now, when SR requests for service, the SP will respond by challenging the SR by asking for a random token, if the token provided by the SR is valid and acknowledged by the SP the SR will receive access to the service.

\[ SR \xrightarrow{\text{requestes connection}} SP \]  

\[ SP \rightarrow S_{SP}, \text{ where } S = \{ s_1, s_2, s_3, ..., s_n \} \]  

\[ T(t_1, t_2, t_3, ..., t_n) = f(s_n), s_n = f(t + \text{Rand}(t)) \]  

C. STEADY PHASE

As per scenario (2), given that both services have already established an initial connection, to be authorized, SR will have to first provide a valid token. This token will be generated while establishing the connection. If the token proves to be valid, then SP will challenge SR to provide a set of expired token in a predetermined order. If the set of expired tokens prove to be valid, i.e. \[ T(n)_{SR} == T(n)_{SP} \] the expired token will be validated and then acknowledged by the SP to grant service to the SR.
\[ SR \xrightarrow{\text{valid token: } t_n} SP \] (2.1)

\[ \text{If } SR = SP, \text{ session established} \] (2.2)

\[ SR \xrightarrow{ex(t_n)} SP \] (2.3)

\[ T_{SR} = (T_m, ex(t_n)) \] (2.4)

\[ \text{If } T(n)_{SR} = T(n)_{SP}, \text{ service granted} \] (2.5)

IV. CONCLUSION

In this paper, we introduced a protocol for service to service authentication using a set of active and expired tokens in the challenged-response format. The service is granted to the service requester when active token is used for authorization, and when the service provider requests a specific set of expired tokens from the requester which will be used for authentication. This protocol combines authentication and authorization between two services while many existing service to service protocols use tokens for authorization only.

REFERENCES


