Session Based Authentication

Session

A session is a series of interactions between two communication end points. One end point requests a connection with another specified end point and if that endpoint replies agreeing to the connection, the end points take turns exchanging commands and data. The session begins when the connection is established at both ends and terminates when the connection is ended.

Open Systems Interconnection ([OSI](http://searchnetworking.techtarget.com/definition/OSI)), the session layer manages the setting up and taking down of the association between two communicating end points that is called a connection. A connection is maintained while the two end points are communicating. Some connections and sessions last only long enough to send a message in one direction. For Internet applications, each session is related to a particular [port](http://searchnetworking.techtarget.com/definition/port), a number that is associated with a particular upper layer application.

Session is nothing but a small amount of data that is storing at the server side. We can handle sessions in different approaches.

1. HttpSessions

2.Cookies

3.URL

4.Hidden Form

1.HTTP Sessions

A session token is a unique identifier that is generated and sent from a server to client to identify the current interaction session. The client usually stores and sends the token as an HTTP cookie and/or sends it as a parameter in GET or POST queries. The reason to use session tokens is that the client only has to handle the identifier - all session data is stored on the server linked to that identifier.

2.Cookie:

A cookie is information that a Web site puts on yourhard disk so that it can remember something about you at a later time. It is information for future use that is stored by theserver on theclient side of aclient/server communication. A cookie records your preferences when using a particular site. Using the Web's Hypertext Transfer Protocol (HTTP), each request for a Web page is independent of all other requests.

Ex. If we take a gmail account, here save your password is there. Suppose if we select the option once means we need not to give our password everytime.

3.URL

URL stands for Uniform Resource Locator. It is the address of a web page. Each page has its own unique web address(URL). This is how your computer locates the web page that you are trying to find.

4.Hidden Form

Hidden forms are like Text Fields.The difference is that the hidden field does not show on the page. Therefore the visitor can't type anything into a hidden field, which leads to the purpose of the field. To submit information that is not entered by the visitor.

Authentication

* The process of identifying an individual, usually based on a [username](http://www.webopedia.com/TERM/U/username.html) and [password](http://www.webopedia.com/TERM/P/password.html). In [security systems](http://www.webopedia.com/TERM/S/security.html), authentication is distinct from [authorization](http://www.webopedia.com/TERM/A/authorization.html) *,* which is the process of giving individual su[ccess](http://www.webopedia.com/TERM/A/access.html) to system objects based on their [identity](http://www.webopedia.com/TERM/I/identity.html). Authentication merely ensures that the individual is who he or she claims to be, but says nothing about the access rights of the individual.
* It is a secure means to recognize a user when accessing government services online. The user remains anonymous, and the authentication process just confirms that the credential is valid and is in the hands of the owner of the credential. Credential authentication does not reveal or confirm identity to the government website. It receives only a message confirming that your credential (username/password) was successfully validated by your bank or by the government.

Authentication Methods:

1.Name and password Authentication

2.Session based Name and Password Authentication

Session-based name-and-password authentication includes additional functionality that is not available with basic name-and-password authentication. A session is defined as the time during which a Web client is actively logged onto a server with a cookie. To specify settings that enable and control session authentication, you edit the Web Site document or the Server document, depending on your configuration.

Sections of Session Based Authentication

1. Single server selection

2. Multi-server selections.

* The single server option causes the server to generate a cookie that is honored only by the server that generated it, while the multi-server option generates a cookie that allows single sign-on with any server that shares the Web SSO configuration document.
* To use session-based authentication, Web clients must use a browser that supports cookies. Domino uses cookies to track user sessions.

Benefits of session based Name and Password Authentication:

#### Customized HTML log - in form

#### Idle session timeout

#### Maximum user sessions

* Internet password Management
* Multi server session based authentication
* Using client certs with password protection enabled for private key access provides a bit stronger validation because not only does a client need to have access to the private key but also the password to use it.

Client Side Authentication

* The session management is server responsibilty. When the session is created, a session token is generated and sent to the client (and stored in a cookie).
* After that, in the next requests betwen client and server, the client sends the token (usually) as an HTTP cookie. All session data is stored on the server, the client only stores the token.
* security measure would be mainly used to further restrict who would be granted permission to communicate with our servers.
* The user would first have to authenticate with a traditional username and password, then would present it is client certificate to the server who would simply verify it's legitimacy as well as it is appearance in it is list of known certificates.
* Firebase makes authentication easy. It can integrate with your existing login server, or authenticate users with only client-side code. It has built-in functionality for email & password, and third-party providers such as Facebook, Twitter, GitHub, and Google.

The process of providng a user's identity is called authentication. Firebase provides a full set of authentication options out-of-the-box.

When a user authenticates to a Firebase app, the following things happen:

1. Information about the user is returned in callbacks on the client device. This allows you to customize your app's user experience for that specific user.
2. The user information returned contains a uid (a unique ID), which is guaranteed to be distinct across all providers, and to never change for a specific authenticated user.
3. The value of the auth variable in your app's [Security and Firebase Rules](https://www.firebase.com/docs/security/guide/securing-data.html) becomes defined. This variable is null for unauthenticated users, but for authenticated users it is an object containing the user's unique (auth.uid) and potentially other data about the user. This allows you to securely control data access on a per-user basis.
4. Firebase apps have built-in support for logging in with email & password, social login providers such as Facebook, Google, Twitter, and GitHub, and single-session anonymous login. Apps that use Firebase's built-in auth services can handle user login entirely with client-side code, saving you time and the headache of operating your own backend.

Using client-side authentication in conjunction with Janrain can enable your users to sign into your page without a page refresh. Page refreshes can be resource intensive and time consuming, and so asynchronous client-side authentication is often the best way to go.

So, here's my code to provide a client certificate to an ASIHTTPRequest prior to a UIWebView request:

-(BOOL)webView:(UIWebView \*)webView shouldStartLoadWithRequest:(NSURLRequest \*)request navigationType:(UIWebViewNavigationType)navigationType { SecIdentityRef identity = NULL; SecTrustRef trust = NULL; NSData \*PKCS12Data = [NSData dataWithContentsOfFile:[[NSBundle mainBundle] pathForResource:@"test.cert" ofType:@"pfx"]]; [self extractIdentity:&identity andTrust:&trust fromPKCS12Data:PKCS12Data]; NSURL \*serverUrl = [NSURL URLWithString:URL\_SECURE\_SERVER]; ASIHTTPRequest \*firstRequest = [ASIHTTPRequest requestWithURL:serverUrl]; [firstRequest setValidatesSecureCertificate:NO]; [firstRequest setClientCertificateIdentity:identity]; [firstRequest startSynchronous]; return YES; }

Ex. If we take a Online ticket reservation System, for a particular time only the session is in active. After that the session will be automatically closed and the text session timed out to be displayed.

Implementation:

<https://auth0.com/docs/quickstart/native-mobile/ios-swift/python>

These are the methods to implement the client side requests.

### 1. Adding the Auth0 dependencies

2. Configuring your Swift project to use an ObjC library

### 3. Configure Auth0 Lock for iOS

### 4.Register Native Authentication Handler.

The above link is used to know the methods and how to use this methods in client side authentication.

Result:

Client side Authentication provides more security for the users. It can done by using the various methods like NsURLConnection. Client side authentication is very important in the case of client side projects.