International Institute of Information Technology, Hyderabad

Central Authentication Service

A Single Sign-On Approach to Web-based Authentication

Ejaz Ahmed – 200601028
Sanrag Sood – 200601078

Faculty Advisor:
Saurabh Barjatiya

Introduction:

With number of web applications in the institute rising day by day, the problem of Identity Management is taking a bad shape. With the traditional sign-on procedure of accepting username and password at each portal; and each portal maintaining its own database of user credentials, users have to not only keep track of a large number of login credentials for each portal he has an account on, but the process of signing on into each portal is also tedious for user experience.

Initial steps were taken to remove the multiple credentials problem by launching a Simple Authentication Service (SAS) which used the credentials from the mail server to verify user identity. While the approach was neat, it certainly had the drawback that the authentication credentials were sent to the SAS service in clear text. SAS service was also vulnerable to brute-force attacks. Also, since SAS was home-brewed, it was not standards compliant and hence existing apps like Zimbra, Squirrel Mail etc. couldn't be integrated with SAS.

To further establish the concept of One Identity, One Password, an LDAP server was setup. Being a standards compliant interface, any application (web, console or desktop) could now be integrated with it and same user identity could be used everywhere. However, the problem of repeating the procedure of login on every portal still persisted. Also, since each portal accepted his username and password from him in plain text, the assurance that the portal developer wasn't storing his username and password and using it maliciously was absent. Also, directly integrating a web application with LDAP requires a basic knowledge of LDAP protocol to every developer.

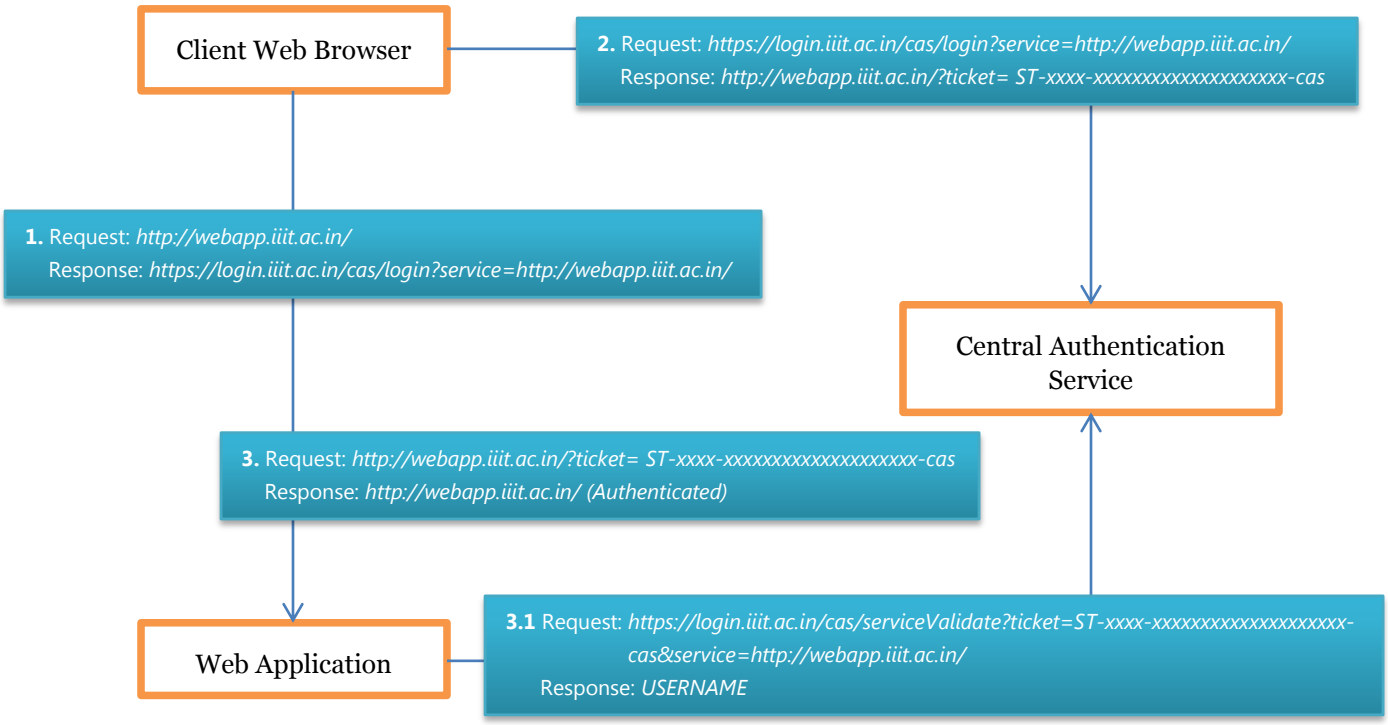
To support rapid application development as well as to remove the aforementioned problems, a Central Authentication Service was required which could implement a Single Sign-On application protocol for web applications. A Central Authentication Service would make sure that users submit their credentials only at a trusted site (here https://login.iiit.ac.in/cas) and he would need to sign-on only once. Also, the service needs to be LDAP compliant so that same credentials could be used everywhere.

About Central Authentication Service:

Central Authentication Service is a Single Sign-On protocol for the web. Its purpose is to permit a user to access multiple applications while providing their credentials only once. It also allows web applications to authenticate users without gaining access to a user's security credentials, such as a password.

The CAS protocol involved at least three parties: a client web browser, the web application requesting authentication and the CAS server. When the client web browser first accesses the web application, the web application seeing the session unauthenticated, redirects the user to the CAS server (https://login.iiit.ac.in/cas/login) with a *service=<web app URL>* appended in the query string. At the CAS server, if the user is visiting it for the first time, he's unauthenticated at the CAS server also. The CAS server asks user for his login credentials and verify them by contacting the LDAP server. Once the user is verified, the CAS server sets a secure Ticket Granting Ticket (TGT) cookie in the user's browser. It then

generates a random ticket for the user with a short validity (5 minutes in our case). This ticket is valid for single use only. This ticket is appended to web-app URL received by CAS server in the query string as a value to *service* parameter and the user is redirected to the modified URL.



The user again arrives at the web-application but this time with a ticket. The web application takes the ticket and verifies the ticket with the CAS server by calling the URL (https://login.iiit.ac.in/cas/serviceValidate?ticket=<ticket>&service=<web-appURL>) internally. If the ticket is valid, the CAS server replies with the following response:

```
<cas:serviceResponse xmlns:cas='http://www.yale.edu/tp/cas'>
    <cas:authenticationSuccess>
        <cas:user>USERNAME</cas:user>
    </cas:authenticationSuccess>
</cas:serviceResponse>
```

The web-application then gets the username from response and maintains an internal session for the user. It then redirects the client web-browser to the application URL without a ticket parameter in the query string.

When the user visits another web-app, it also redirects him to the CAS server. But since the user is already authenticated at the CAS server, the CAS server doesn't ask for his credentials but directly

generates a ticket for him. Note that this ticket generated is in no way related to the previous ticket. Once the ticket is generated, the same procedure follows.

When the user clicks on logout in any application, the user session in that application is invalidated, logging him out of that particular application. After that, the web application redirects the user to https://login.iiit.ac.in/cas/logout which invalidates the user's CAS session also. The CAS server is capable of signing out the user from other services also by sending a SAML request to registered services. However this requires heavy modification of existing codes of all the services and hence is not used as of now. It is to be noted here that though logging out of one application logs out the user from CAS also, the applications to which user has already signed in will continue to work and the user has to log out from them manually. However, the user won't be able to automatically sign-in into other apps without providing his credentials again.

Setting Up CAS Server:

The information on setting up the CAS server is available on http://192.168.36.131/wiki under restricted access

Integrating Web-Apps with CAS:

Integrating an existing web-application is very easy. Libraries have been made available for different languages for smooth integration. For PHP developers, the official phpCAS library is maintained by the JA-SIG group which also maintains CAS. For Mod Python developers, a library was developed and is now hosted on http://bitbucket.org/iiit/pyiiit/ under Mercurial Version Control. The documentation to use these libraries is available on http://login.iiit.ac.in/help/ . Libraries for other languages are available at http://www.ja-sig.org/wiki/display/CASC/Home

Reference:

JA-SIG Central Authentication Service (http://www.jasig.org/cas)