***To Make FIDO Server and Relying Party more Robust….***

Currently we have a fully-fledged FIDO Server, following all FIDO specified protocols properly and in working condition. The current FIDO server is implemented using JAVA as the main language and the code is built using GRADLE, which creates a WAR file at the end. This WAR file is hosted on an Apache tomcat server 7 to FIDO server functionalities.

The Relying Party website has two parts: A Frontend UI (designed using Angular + Bootstrap + HTML + CSS) and a Backend Server (designed using Node.js and Express). It basically follows the ***MEAN*** (**M**ongoDB, **E**xpress, **A**ngular and **N**ode) rule.

With all this implementation, we have to take care of 3 different technologies, i.e., JAVA, Node.js and Angular.js with Bootstrap, which will be difficult as we progress with adding more functionalities to FIDO server as well as Relying Party website. Also currently during authentication, the web browser (Angular UI), is directly calling the FIDO server API, for FIDO authentication, which if fails, keeps on polling and there is no other way to stop a web browser from polling a JAVA API, unless we close the web browser completely. We feel this is the major reason why the “Too many Connections” error happens on FIDO server side.

To avoid above issue from Relying Party side, we are planning to change the Relying Party website a bit. The front end will use the Angular + Bootstrap technologies (We are considering this, because of easily available readymade templates for angular UI, which will save the work of implementing a new UI and also give good appearance to the website.). The backend server will change completely and we are planning to use a JAVA tomcat server (with new JAVA REST APIs, which will call the actual FIDO server APIs) for doing the backend processing of website.

So we will have 3 things to be maintained in future:

1) Relying Party UI (Angular +Bootstrap code).

2) JAVA server with REST APIs to host the Relying Party website (JAVA code).

3) JAVA server with REST APIs for FIDO server (JAVA code).

***How will the Relying Party website’s JAVA backend server REST APIs will call the actual FIDO server REST APIs that do the FIDO processes (registration, authentication, transaction and deregistration).? (Implementation):***

**A]** FIDO Server API details:

1) **URL**: [https://10.244.48.148:**8443**/fidouaf\_test/v1/public/[API\_Name](https://10.244.48.148:8443/fidouaf_test/v1/public/%5bAPI_Name)]

2) **Methods**: GET and POST

3) **Headers to be set**: content-type: application/json, accept: application/json

4) **Input data**: differs based on API to be called.

Above is the URL that needs to be called whenever a different backend server wants to use FIDO server APIs for FIDO functionalities. The URL will change according to which API needs to be called on FIDO server, based on API Name appended at the end of the main URL, the Method type (can be a GET or POST API), the headers to be set, and the input json data to be passed to the API.

**B]** Relying Party Website Backend JAVA Server API details:

The implementation of this server will be exactly the same as existing FIDO server. I will develop JAVA REST APIs here also (http/https), which in turn will call the actual FIDO server APIs. For now, consider the following example of URL that I’m planning to keep for this RP website server:

1) **URL**: [https://10.244.48.148:](https://10.244.48.148:9443/RelyingParty/public/[API_Name)**[9443](https://10.244.48.148:9443/RelyingParty/public/[API_Name)**[/RelyingParty/public/[API\_Name](https://10.244.48.148:9443/RelyingParty/public/[API_Name)]

***Note****: Notice the change of ports in both the JAVA server APIs. The Main FIDO server will continue running on ports 8080 for http and 8443 for https. Whereas the RP website server will run on ports 9090 for http and 9443 for https.*

2) **Methods**: GET and POST

3) **Headers to be set:** content-type: application/json, accept: application/json

4) **Input data:** differs based on API to be called.

The Relying party website (Angular + Bootstrap UI), will make an AJAX call to its backend server API, i.e. URL: [https://10.244.48.148:**9443**/RelyingParty/public/[API\_Name](https://10.244.48.148:9443/RelyingParty/public/%5bAPI_Name)].

The method that will be invoked on calling the above API on the website’s backend JAVA server will do the following things:

a) Declare the Actual URL of the API to be called on the FIDO Server.

b) Create a HTTP Client object to create POST or GET Call request.

c) Create the input Data and convert it into JSON.

d) Set the headers required for the API to be called on FIDO server.

e) Execute the POST or GET call request.

f) Get the Results in a String and Convert it back into JSON object (if JSON is returned as output of the API called on FIDO server).

g) Extract data out of result JSON and display it on the RP website (Angular + Bootstrap), wherever required.

Research required before starting with the actual implementation:

1) How to go about designing the RP Website UI using readymade templates available online for Angular + Bootstrap?

2) How to make AJAX post/get calls from the RP website (Angular) to its backend JAVA server?

3) How to build this whole RP website with backend JAVA server code into a WAR File (Can we use Gradle here too just like how we build our FIDO server code?)?

**1) How to go about designing the RP Website UI using readymade templates available online for Angular + Bootstrap?**

I searched for online readymade templates for responsive websites and found and option of HTML5 with bootstrap, Angular with bootstrap and JQuery with bootstrap options. I am thinking of going ahead with the simplest HTML5 with Bootstrap option for designing the front end of the Relying Party website.

We will need to take a ready template available online and start to add features to it, for example: login form, FIDO login form, registration form, transaction form, FIDO registration form, etc. Depending on the template we have downloaded and decided to use, we will need to add the above changes. Some templates have the login form inbuilt, whereas some templates don’t. So accordingly will have to add this pages to our website. I have already started with one website template.

**2) How to make AJAX post/get calls from the RP website (Angular) to its backend JAVA server?**

Here I’m thinking of using the standard *$AJAX ().POST* and *$AJAX ().GET* functions just the same way as I did for the Relying Party website that I had first created (The simple website that I had created with no good appearance). In this AJAX calls, I used to call my FIDO server APIs directly. But now the change here will be to call the Relying Party’s own server’s API instead, which in turn will call the FIDO server APIs. The way of calling the Relying Party’s server APIs will not change, and I’m also thinking of keeping the output format for each API that I will write for Relying Party’s server same as the FIDO server APIs output, so that we don’t face the problem of parsing the output data and extracting the required fields. (If required to be shown on UI).

**3) How to build this whole RP website with backend JAVA server code into a WAR File (Can we use Gradle here too just like how we build our FIDO server code?)?**

The UI of the Relying Party website is created using HTML5 with Bootstrap (readymade/downloaded from net), with its CSS and JS script files along with it. I’m planning to create HTML5 project in eclipse with all the UI stuff in there, and will host this website on Apache tomcat 7 server with ports 9443 for https and 9090 for http. This JAVA tomcat server will then interact with the FIDO server running on another tomcat 7 server with different ports, process data, and get outputs which will be then parsed and shown to user on UI (wherever required).

Else, we can always create a WAR and deploy it on the server. To create a War out of the HTML5 project in eclipse, I’m planning to use MAVEN as show below:

1) Install Maven plugins in eclipse

2) Convert existing dynamic project to maven project (Reference: <http://crunchify.com/how-to-convert-existing-java-project-to-maven-in-eclipse/>)

3) Finally create the .WAR file out of the converted maven project (Reference: <http://crunchify.com/how-to-create-a-war-file-from-eclipse-using-maven-plugin-apache-maven-war-plugin-usage/>)