

Assignement#1

sdh140430 (Swapnil Hasabe)

Question1:

1. Introduction about Dropbox Application:


1.1 Dropbox is application which allows users to upload files that could then be shared over the cloud by the same user or possibly by other users, after a password or other authentication is provided.

1.2 Users can create a special folder on their computers, which Dropbox then synchronizes so that it appears to be the same folder (with the same contents) regardless of which device (different computer, tablets, smart or other networked device) is used to view it. User also can access these files via Dropbox website and mobile apps

2. How Dropbox shares Information Securely in the Cloud:

2.1 Dropbox keeps the **security** of users information is at **highest priority**

2.2 **Two-step verification**: User can enable a **login authentication** feature to add another layer of security to their account by choosing to receive **security codes by text message** or via any Time-Based One-Time Password apps.



Enter your phone number

2.3. **Encryption to provide confidentiality**: Dropbox keeps files in encrypted form. It uses 256-bit Advanced Encryption Standard (**AES**) algorithm. Only authenticate user can decrypt those files.

2.4. **Dropbox uses Secure Sockets Layer (SSL)/Transport Layer Security (TLS)** to protect data in transit between **Dropbox apps and their servers**; higher Advanced Encryption Standard (AES) is used to provide encryption.

2.5. People (Non Dropbox Users) only who have a like to a file can view **public files**.

2.6. **By default**, anything user store in his/her Dropbox is **private**—only authentic user can access it.

2.7. If you're the **owner** of a shared folder, you can **unshare** it at any time. Non Dropbox users in a folder do not have the ability to unshare it.

2.8. If you own a shared folder, you can prevent other members from adding, editing, or deleting files in the folder by setting view only permissions in that application.

Question 2:

a) **SAML**: is the standard that encompasses profiles, bindings and constructs to achieve Single Sign on (SSO), Federation and Identity Management.

Security Assertion Markup Language (SAML) support single-sign- on feature.

Token or Message format: SAML deals with XML as the data construct or token format.

Transport: SAML has Bindings that use HTTP such as HTTP POST Binding, HTTP REDIRECT Binding etc. But there is no restriction on the transport format. You can use SOAP or JMS or any transport you want to use to send SAML tokens or messages.

Scope: Even though SAML was designed to be applicable openly, it is typically used in Enterprise SSO scenarios. E.g. enterprise to cloud scenarios

Identification of user: SAML identifies user on signature request.

Example: Jobvite Application is analytics-driven recruiting software accelerates hiring with an applicant tracking system, social recruiting, and mobile solution.

Step1: When any user tries to log in Jobvite application (Online or via API), Jobvite validates username and checks user profile's setting. If user profile has "Uses single sign on" user permission, then Jobvite does not authenticate the username with password. Instead, a web service call made to user's single sign on service, asking it to validate username with password. E.g. Facebook does that work for Jobvite application.

Step2: Facebook verifies user's identity. (Authentication)

Step3: Facebook provides SAML assertion statement to Jobvite behalf of user. When Jobvite receives SAML assertion statement, it allows users to log in.

SAML boils down to attribute exchange through the creation of trust relationships between IdP's and SP's.

Overview of how SAML plays Role in Cloud:

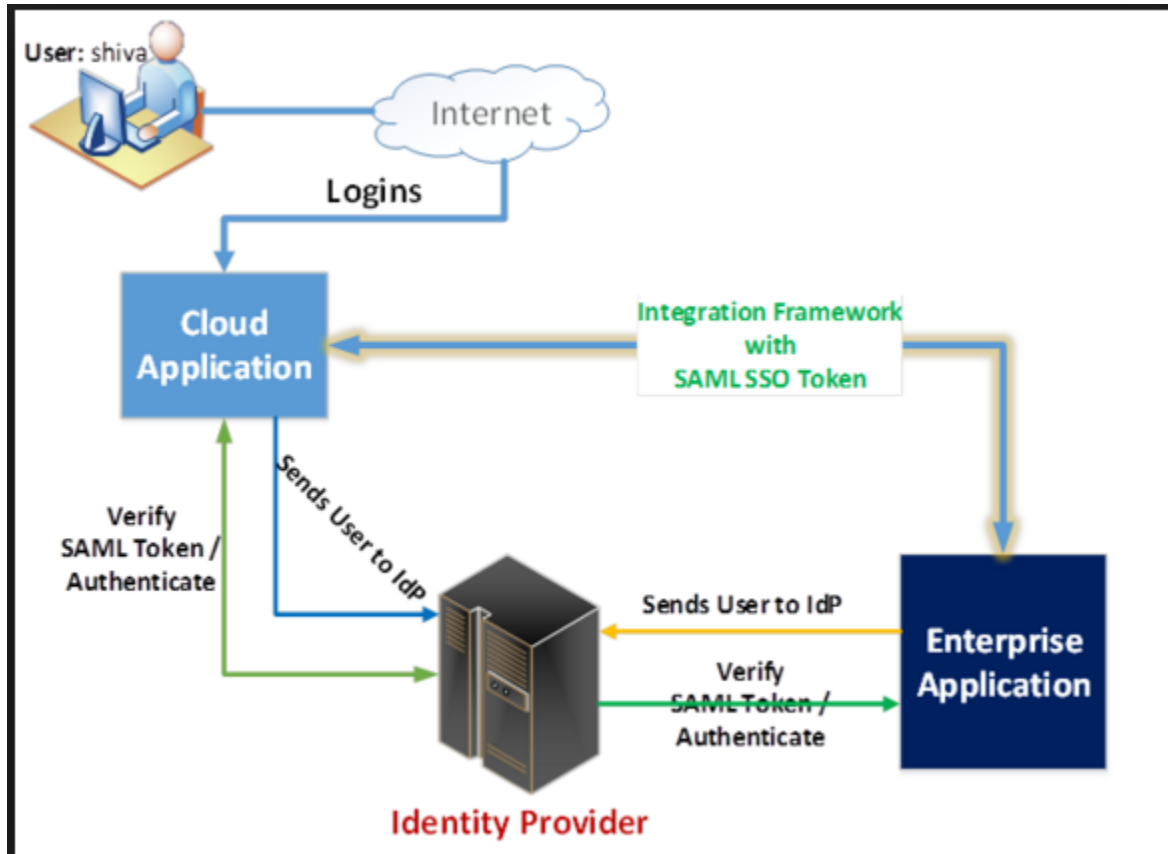
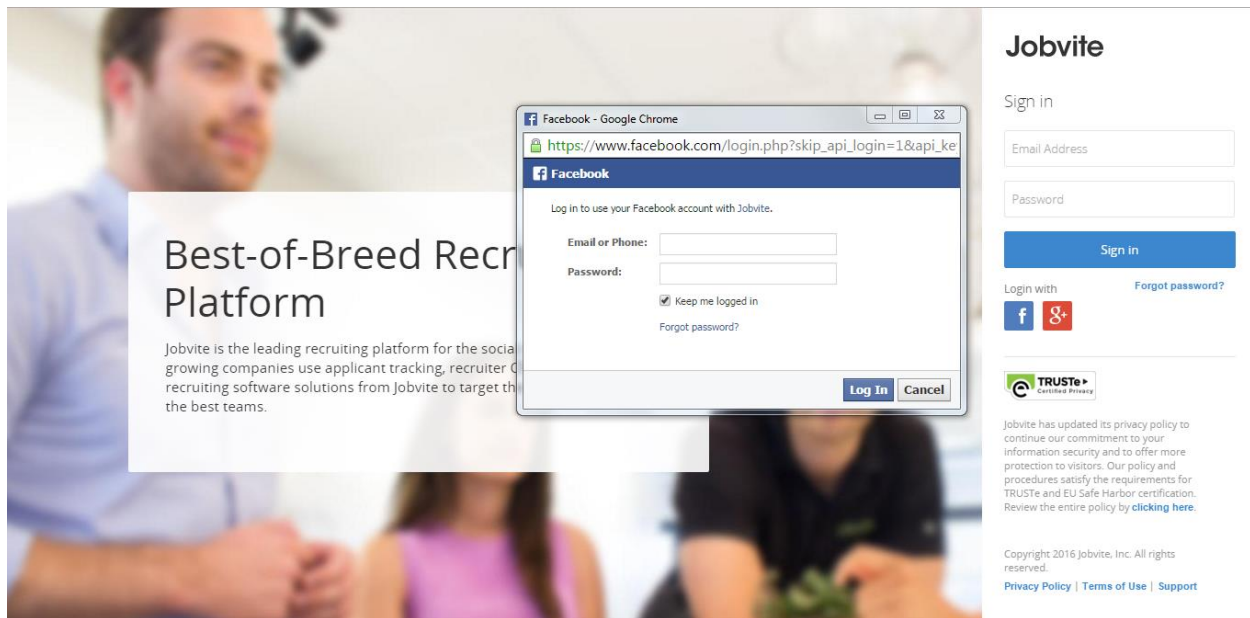
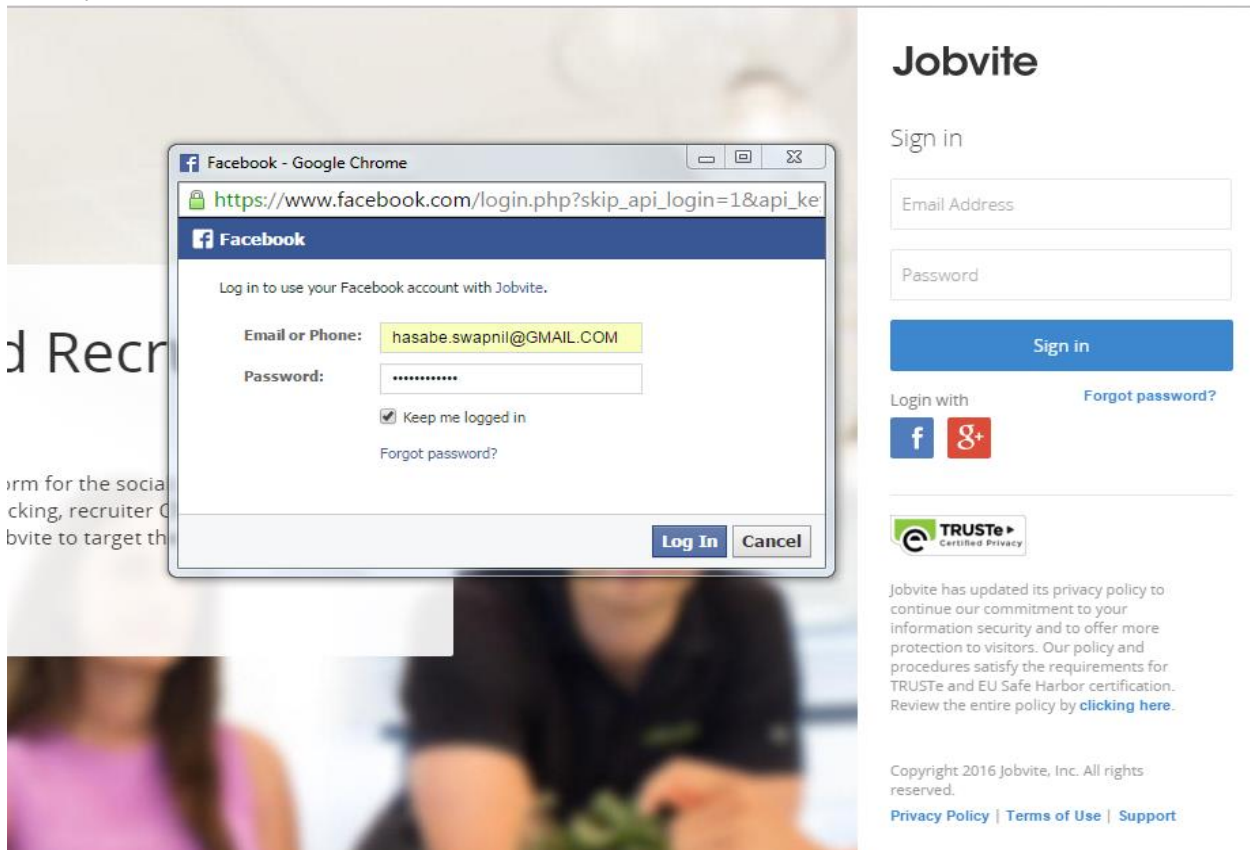


Illustration:

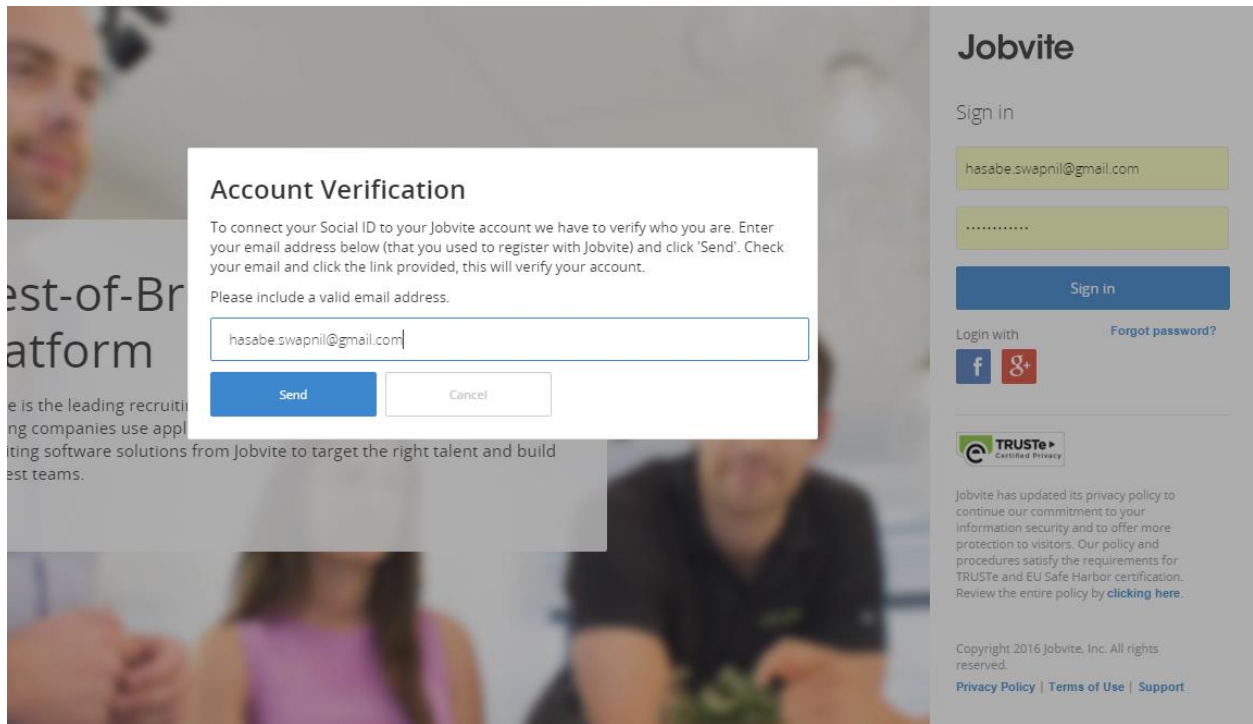
I want to log into Jobvite Application. I can use Single sign on feature. I will give facebook credentials and will log into Jobvite. Facebook does authentication of user for Jobvite



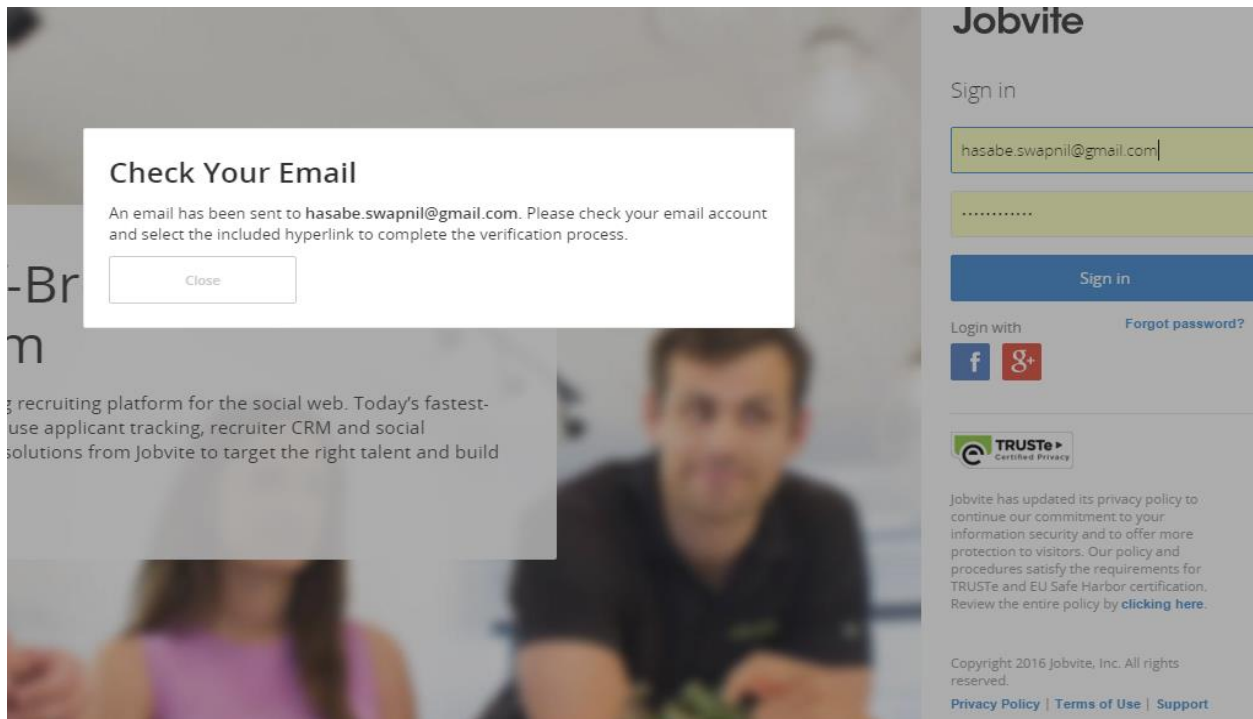
Put my facebook credentials

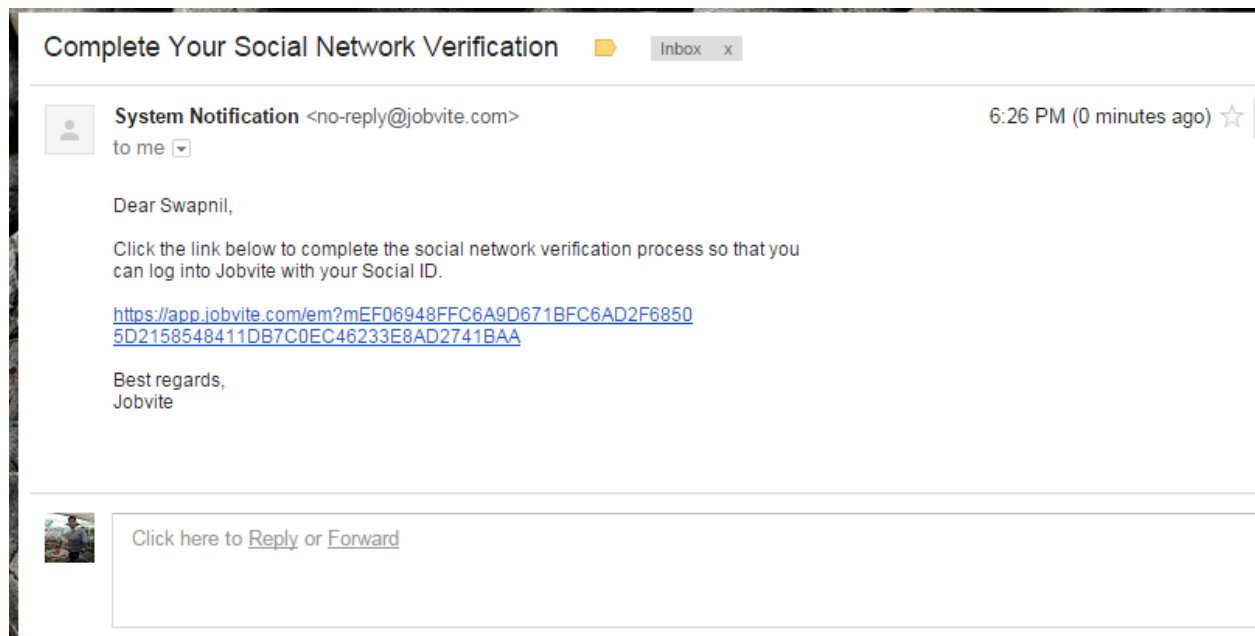


Asking for Account Verification.Prompted to enter email address

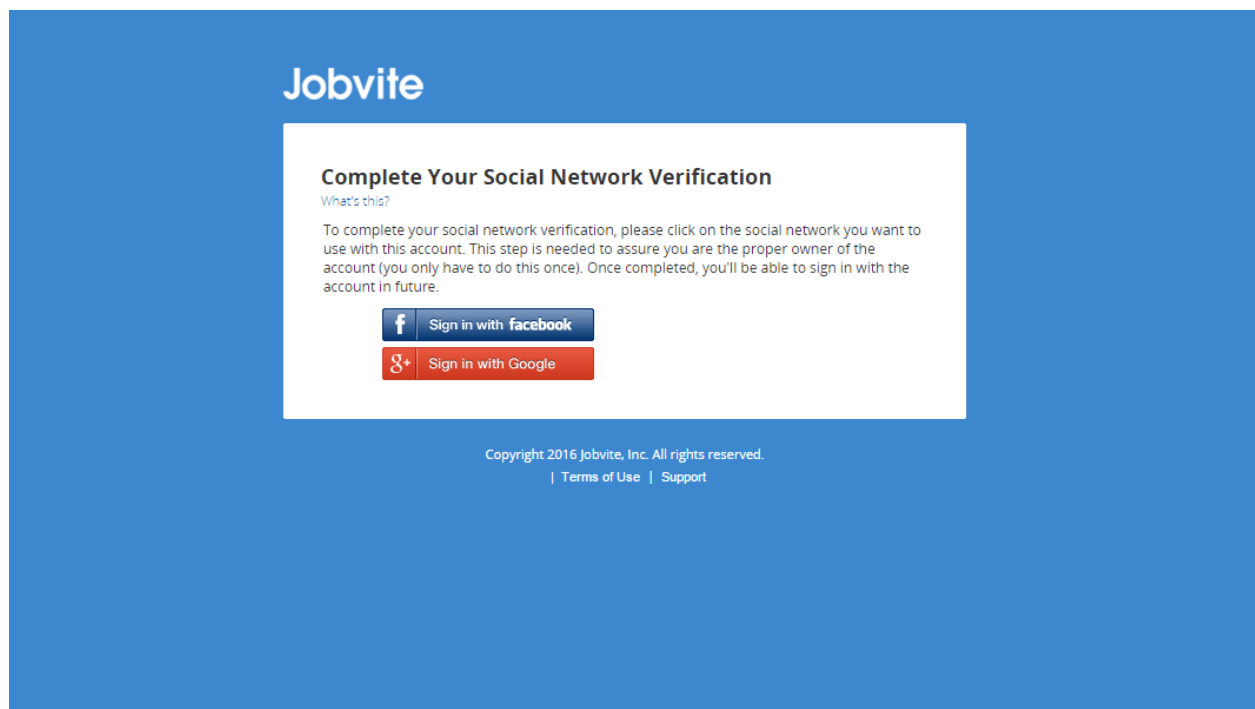











User receives link in email account.





In this way, facebook does Authentication of user using SAML. Because it provides SAML assertion Certificate to Jobvite behalf of User.



Jobvite			
Applications			
1 - 20 of 97			
Company	Title	Status	Date
 Zynga	Software Engineer (Slots)	Not Selected	02/18/2016
 fitbit	Software Engineer - New Grad	New	02/12/2016
 Square	Software Engineer	Not Selected	02/12/2016
 okta	Software Engineer in Test - Core Technology	Not Selected	01/18/2016
 Trustwave	Software Engineer (Java)	Not Selected	01/12/2016
 bina	Software Engineer Spring 2016 Grads	New	01/11/2016
 yext	Software Engineer - Fresh Grads!	In Process	12/29/2015
 Shutterstock	Frontend Engineer, University Graduate	Not Selected	12/29/2015
 Shutterstock	Software Engineer, Intern & New Grad	New	12/29/2015

Finally, User got logged into Jobvite using SAML.

b) XACML

Question 1: Source: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1625374>

Modeling permissions in a (U/X) ML world

Goal of Application: SECTET project cluster a model driven approach for end2end security in the context of inter-organizational workflows. We focus on the aspect of authorization. Authorization in a SOA basically concerns the question if a subject is permitted to call a specific web service.

How this application uses XACML: Refer fig.6 in this paper; it represents 5 rules regarding this application. So, these rules are written in XACML format. Eventually, application will use this XACML format to get required result. E.g. A physician is allowed to check the medical records of every patient.

In this example, application uses the rule 5 in XACML format and gives authority to physician to check the medical records of every patient.

In rule 1: XACML authorizes a person having patient ID to make an appointment only for week days. It means XACML is defining role here. This is simply access controlling of person.

In rule 4: XACML authorizes a nurse to read medical records of every patient for her specialization only. It denies permission when patient disease is marked as confidential.

<pre> <PolicySet PolicySetId="RPS:PhysicianRole" Combining Algorithm = „deny-overrides“ <PolicySetIdReference>NPPS:PhysicianRole</PolicySetIdReference> <PolicySet Combining Algorithm = „permit-overrides“ <PolicySetIdReference>PPPS:PhysicianRole</PolicySetIdReference> <PolicySetIdReference>DenyPolicy</PolicySetIdReference> </PolicySet> <Target> Role Definition </Target> </PolicySet> </pre>	<pre> <Target> <Subjects> <Subject> <SubjectMatch MatchId="string-equal"> <AttributeValue DataType="string"> PhysicianRole </AttributeValue> <SubjectAttributeDesignator AttributeId="urn:someapp:attributes:role" DataType="string"/> </SubjectMatch> </Subject> </Subjects> <Resources> <AnyResource/> </Resources> <Actions> <AnyAction/> </Actions> </Target> </pre>
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Left Figure:

It shows an example RPS for PhysicianRole. The PolicySetId points to the associated role, the combining algorithm is set to "deny-overrides" which is meant to logically enforce the precedence of negative permissions over the positive permissions. In our case we assume that negative permissions were specified for PhysicianRole in the Access Model. They are referenced through NPPS: PhysicianRole inside a Element. Positive permissions are specified in an inner policy. Notice that the combining algorithm is set to permit-overrides, requiring only one out of many PPPS to return a positive result.

Right Figure:

It shows the element block inside the RPS specifies the Role Definition. According to XACML syntax, the role name (e.g., PhysicianRole) is specified with an element given inside the element block as shown in Figure 9. It makes the RPS applicable to any XACML Request possessing the role PhysicianRole. The element searches for a particular attribute which in our case is a role. This subject attribute is matched to an attribute in the request context. The elements are irrelevant for RPS identification.

In this way, XACML Application can be used in cloud.