UNIVERSITY OF GOTHENBURG

Authentication and Authorization for Mobile Devices

Bachelor of Science Thesis in Software Engineering and Management

NAVID RANJBAR
MAHDI ABDINEJADI

Authentication and Authorization for Mobile Devices

NAVID RANJBAR
MAHDI ABDINEJADI

Examiner: HELENA HOLMSTRÖM OLSSON

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden June 2012

Authentication and Authorization for Mobile Devices

Navid Ranjbar
Department of Computer Science and Engineering
University of Gothenburg
Gothenburg, Sweden
navid@student.gu.se

Mahdi Abdinejadi
Department of Computer Science and Engineering
University of Gothenburg
Gothenburg, Sweden
m.abdi@student.gu.se

***Abstract*— Nowadays market demand forces companies to adapt to mobile technology. For an enterprise company, this change will bring up security challenges. In this article, we investigate authentication and authorization aspects of security. We conduct a case study in Volvo IT in order to extract their requirements regarding to authentication and authorization of their current and future mobile applications. Also we investigate three security protocols: OAuth, OpenID and SAML to find out to what extent they can satisfy the challenges and requirements.**

Keywords-component; Authentication; Authorization; Mobility; Mobile Devices; OAuth; OpenID; SAML.

I. INTRODUCTION

Mobile technologies are enabling new form of customers and business applications (Teng & Helps, 2010). Mobile businesses on the other hand are boosting enterprise by providing the easy access to the services for third party business partners, consultants and customers (Fitzgerald, 2009). Enterprises are adopting mobile technology and providing to market various applications to increase their functioning competence by offering customers and employees greater access to the real-time information (Unhelkar & Murugesan, 2010).

With advancement of mobile technology, new form of customers and applications emerged which produces new challenges for companies. The challenges mainly caused by integrating to mobile services; in order to get adapted to mobile advancement, companies should expand the boundary of their internal services through Internet. This change will enable mobile devices to get access to services, which are usually available inside the company through native mobile applications.

One of the main concerns with mobile technology and mobile devices is how enterprise companies should deal with the challenges of security. User experience in mobile devices could be completely different than what enterprise used to deal internally. Also enterprise companies are constantly dealing with the trade-off between making their application easier to use and making them more secure.

Volvo is one of the companies that are in the process of coping with the mentioned challenges. They are looking on different security aspect like authorization and authentication of mobile users to their backend systems. Thus they want to investigate different protocols in industry that handle these challenges. In this article we are going to research three protocols, which can be implemented for mobile security in enterprise environments. OAuth is an authorization protocol, which is in its nature considering mobile client and application as well as web application and platforms. Also OpenID is a web based authentication protocol that enables users to get authenticated through third party identity providers as well as in-house identity providers. SAML is another protocol, which deals with the transforming authentication and authorization information between security domains. We are mainly trying to answer the question of: How OAuth, OpenID, and SAML can help mobile devices in getting secure access to resources and services from Volvo backend servers?

The remainder of this report is structured as follows: in section II, we are going to describe what are OAuth, OpenID, and SAML? How do they work? Section III is mainly about the research method we used to address the problems of our case study and which steps we took to gather our data. Section IV is the place that we present our analyzed data that we collected during the research period. In section V we map Volvo requirements with the capabilities of OAuth, OpenID and SAML and try to figure out in what extent these protocols can satisfy the requirements of Volvo. Finally, we are going to conclude and discuss about possible future works at section VI.

In this article we are going to address only OAuth 2.0, OpenID 2.0 and SAML 2.0.

II. KNOWLEDGE BASE

Information considered as an asset for individuals and organizations; so, they try to protect their information assets from any posing threat (Todorov, 2007). Security methods help to protect the information. Authentication and

authorization are aspects of security and we mainly focus on them in this article.

A. Authentication

Authentication is the process of proving who really you are. It often happens at first step of interaction between user and operating system or any services. Todorov (2007) mentioned that authentication usually refers to determining and validating user's identity. This process should occur based on one or combination of the three sorts of credentials: something you have, something you are, something you know (Windley, 2005). For example, ID and password combination is something that only an individual knows or certificate is something that an individual have. The most common authentication function is ID and password. Windley (2005) defined strong authentication is an authentication method that required two or more credentials like ID/password combination plus certificate.

B. Authorization

"Authorization is the process of determining whether an already identified and authenticated user is allowed to access information resources in a specific way." (Todorov, 2007)

By authorization systems determine the level of access and right of a user to certain resources or services. Authorization comes after authentication as a result authentication is fundamental for authorization; the reason behind that is user should be proved who is she then system should assign her right and control access. This chain of event always happens sequentially unless user's identity is not important like public services (Windley, 2005).

User can be authenticated by a specific identity; also she can apply for authorization under another identity. Such a request for authorization under another identity to access a service typically refers as Authentication Identity; and if this process happens by an application or service acting on behalf of the user, it called Impersonation. Impersonation can be permanent or temporary. Impersonation is suitable for client/server in the way that servers or applications can access resources on behalf of users. Impersonation also allow limited access to another user's services or resources when a user, application, or service acting on behalf of another user (Todorov, 2007). Also, Boyd (2012) defined *delegated authorization* as transferring access to another user, application, or service, so that acting user, application, or service can perform tasks on behalf of the user.

ACCESS CONTROL

"It is important to understand that access control is not a complete solution for securing a system"
(Sandhu & Samarati, 1994)

It is a process that gives a user, application, or service defined access right while denying others access (Windley, 2005). In other words, access control limits what a user is able to do, as well as service or application executing on behalf of her; so, it would prevent any execution, which end up with security breach in some degree (Sandhu & Samarati, 1994).

C. OAuth

This section we define common terms and actors in OAuth authorization protocol as well as brief description of important OAuth flows of work.

1) What is OAuth

OAuth is an open protocol to allow secure API authorization in a simple and standard method from desktop, web and mobile applications. OAuth allows a user to grant access to an application to perform on behalf of user; this application can only perform the authorized tasks.

a) Access Token

Access token is a credential that used to access services or resources; it is a string representation of access allowance to the resources, which is generated by the grant of resource owner from the authorization server (Recordon et al., 2012).

b) OAuth Roles

There are several key actors according to (Boyd, 2012) in the OAuth as follows:

- Resource server: The server, which is hosting user's data, which is protected by OAuth.
- Resource owner: The owner of data, in other words the user of the application.
- Client: The application which makes API request to get protected resources on behalf of the resource owner.
- Authorization server: This server gets permission from resource owner and issues access token to client for accessing protected resources available on the resource server.

It is necessary that to register applications with the authorization servers since API requests need to be properly identified. The protocol allow this process by the automated means but most of API providers request manual registration through filling out a form on their developers website.

After the registrations get completed, identification provider issues the credentials to the developer. These credential are: Client ID and Client Secret, which are needed in order to make the authorization requests. This client credentials has two main benefits; first in the process of making authorization request and exchanging authorization code for access token, this credentials acts as the means of authenticity of the requests. Second the user experience will improve during the authorization process by showing to the user the name and logo of the application that trying to access to the their resources.

c) Client Types

According to the OAuth Authorization Framework RFC (Recordon et al., 2012), there are three client profiles available for OAuth.

SERVER SIDE WEB APPLICATION

In this profile the OAuth client is on the web server and that web server is making the API calls with the permission of the resource owner with help of a server side programming language. In this profile the user does not have access to the client credentials or the access token.

CLIENT SIDE APPLICATION ON THE WEB BROWSER

An OAuth client, in this case, is running on the web browser, it could be a JavaScript included in web page, a browser extension or a plugin to the web browser (like an adobe flash application). Protocol data and credentials are easily accessible and often visible to the resource owner.

NATIVE APPLICATIONS

It is an OAuth client that is running on a mobile device and recourse owner is using that client to access her resources. Client credentials included in the application, can be extracted, however issued credentials such as access token or refresh token can be protected. Depending on the platform the credential can be protected from different application installed on the device. This profile is the main profile that we are going to cover in this article.

d) Authorization Flows

Boyd (2012) stated that the core OAuth protocol defines four primary "grant types" used for obtaining authorization, also it defines extra mechanisms for additional grant access. But in this article we are going to cover the two main flows, which is related to the mobile applications.

AUTHORIZATION CODE FLOW

This flow is more appropriate for the server-side web applications or the mobile applications that have the application servers, after the resource owner granted access to her data, authentication code is redirected back to the web application or the application server as the query parameter in the URL (Figure 1).

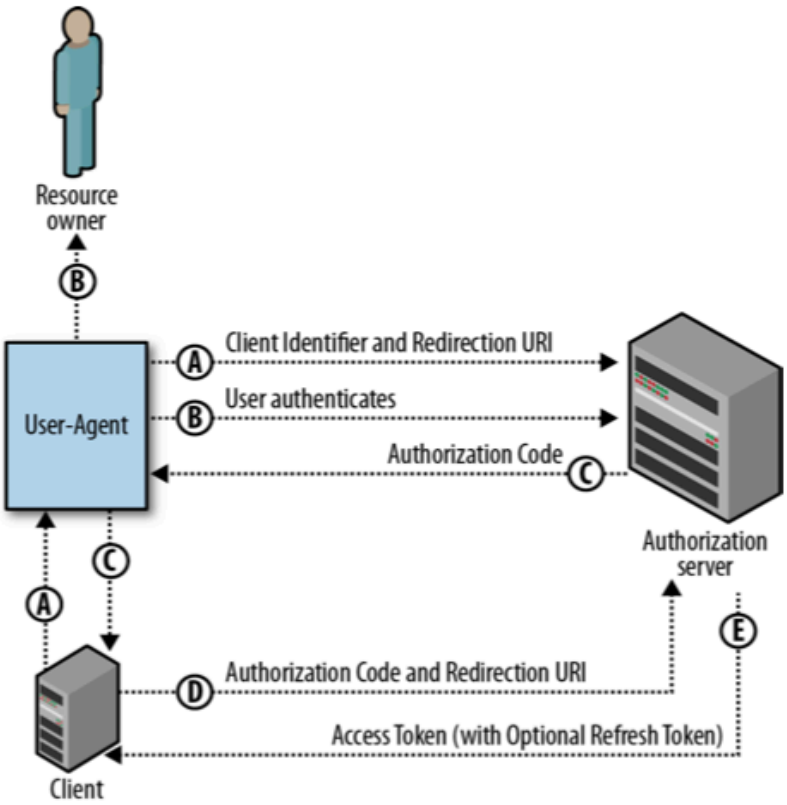

Figure 1. Authorization code flow (Adapted form (Boyd, 2012)

As in figure 1 illustrated, we present the process step by step:

- Step A: Client, which is the application server in this case, sends through the user-agent (Browser/mobile application) client identifier and redirect URL to the authorization server.
- Step B: Authorization server asks for the authentication process and user enters her password.
- Step C: After user get authenticated, authorization server sends back the authorization code to the user agent, then user agent forwards the authorization code to the client (application server).

- Step D: Client (application server) sends the authorization code to the authorization server.
- Step E: Authorization server generates the access token and sends it back to the client.

FLOW PROPERTIES

This flow is more suitable when long lived access tokens are required; this type of tokens has more accountability compared to temporary tokens since credentials would not be required from user as long as the token is valid and the token is only available to the application server.

IMPILICIT GRANT

This grant access is optimized for the client side web applications running in browser or mobile applications, which are not connected to any application servers. The resource owner grant access to the application and new access token is immediately passed back to the application using a hash fragment in the URL (Figure 2).

As in figure 2 illustrated, we describe this flow step by step:

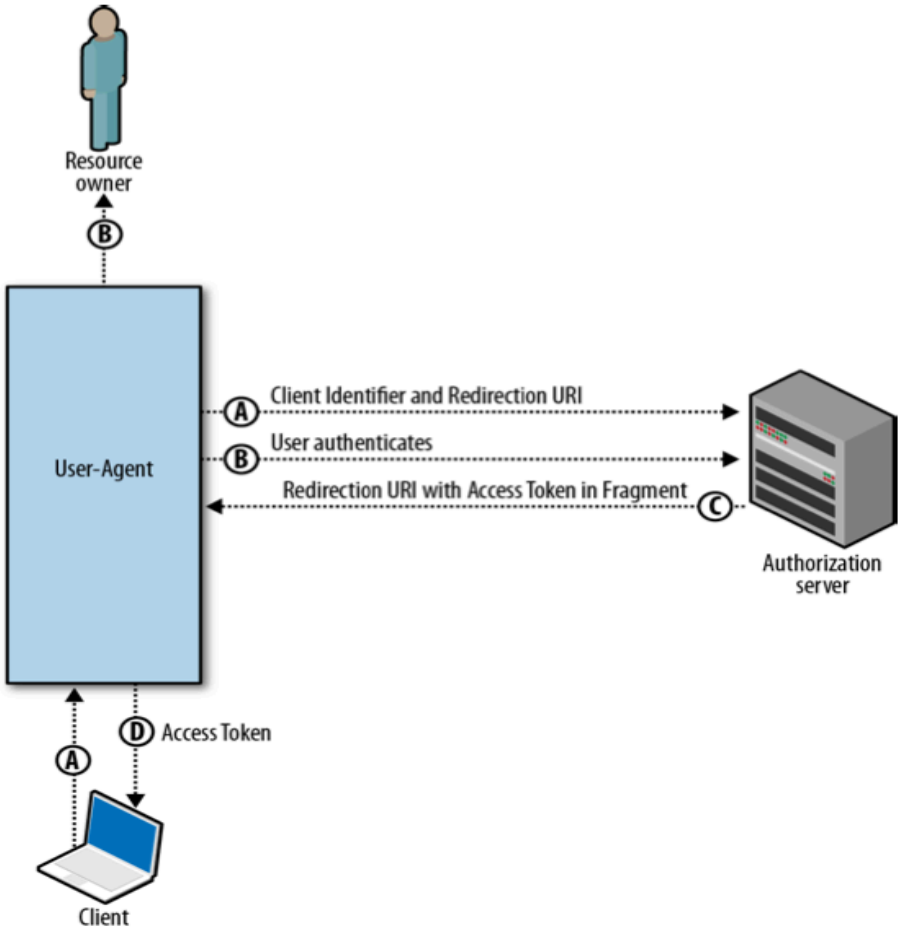

Figure 2. Implicit grant flow (Adapted from (Boyd, 2101))

- Step A: Client, which is mobile device in this case, sends the client identifier and redirect URL to the authorization server.
- Step B: Authorization server asks for the authentication process and user enter her password.
- Step C: Authorization server generates the access token and sends it back to the user agent.
- Step D: User agent forwards access token to client (mobile application) and client should save and optionally protect the access token.

D. OpenID

OpenID is an open protocol for authentication that let users to apply http(s) URL as identity in a specific website and extend same URL identity to multiple OpenID enabled websites. Also, web integrated applications are enabled to operate upon this identity URL for authentication. This means that users have control on their credentials without exposing it to third party (Rehman, 2008). As in figure 3 showed, "Eddie.openid.mydomain.com" is the URL identity used for OpenID sign in.

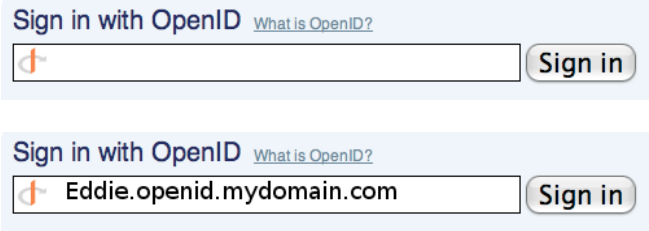

Figure 3. URL based authentication with OpenID

Rehman (2008) mentioned that OpenID empower users to:

- Login to a website or web enabled application without exposing credentials.
- Let websites to ask users information; and give users right to choose which information should be send to third party website over authentication process.
- Determine proper set of information to send to website based on need.
- Provide single sign on (SSO) authentication option in multiple applications and websites inside an organization.
- Incorporate websites and web enabled applications to OpenID framework in a simple way.
- Reduce the cost for managing and maintaining authentication solutions of an organization for consumer.
- Simplify authentication execution for end-user.

OpenID consists of three major components: Customer, Identity Provider, and User Agent (Rehman, 2008). These three components communicate and collaborate during authentication (Figure 4).

OpenID terminology:

- End-user: A person who login to variety of website by OpenID system.
- Consumer or relay party: Is the actual website that End-user wants to login.
- Identifier: is a URL that recognizes End-user.
- Identity Provider or IdP: Is the website that End-user have credentials stored there. This host can identify user based on matching correct username and password.
- User Agent: is the web browser that End-user uses.

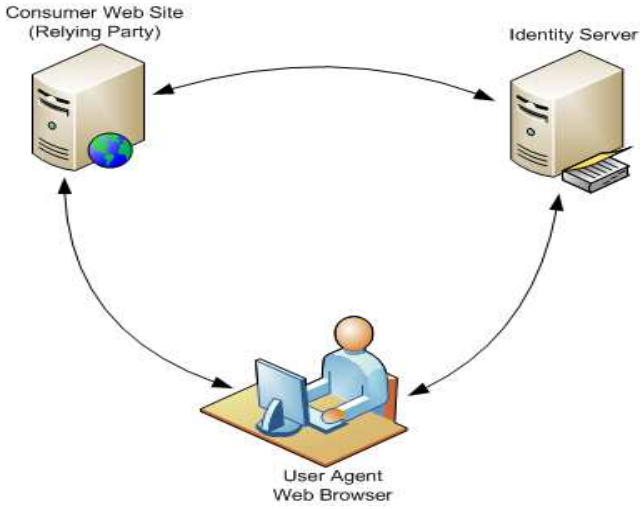

Figure 4. OpenID components (Adapted from (Rehman, 2008))

OpenID have two mode of communication: dumb and smart. In dumb mode, Consumer does not keep the track of state of connection; so, for every login End-user should follow the whole step of authentication and basically repeat them. On the other hand, in smart mode, Consumer maintains the state data and caches shared keys for later login. So, smart mode can effectively reduce traffic of the servers when End-users what to login many times (Rehman, 2008).

Dump mode illustrated in figure 5 (Rehman, 2008):

1. End-user visits Consumer web page and wants to login.
2. The web page requests the Identity Provider URL or simply provides a list of identifiers to choose for End-user.
3. Consumer gets information – a web page – from Identity Provider.
4. Consumer parses the information – HTML embedded – and detects Identity Provider's location. This step is named discovery. Then, Consumer redirects web page to Identity Provider to get assertion data by HTTP GET.

4a. Optionally, Consumer can exchange share key with Identity Provider.

5. End-user logs in to Identity Provider's web page.
6. Identity Provider sends back – by HTTP GET – assertion data with signature to Consumer through browser redirect. It shows authentication success or failure.
7. In case of successful authentication, Consumer directly asks Identity Provider for assertion data. This data can be checked and validated against User Agent data.
8. If assertion information is valid, End-user can login to Consumer website otherwise the Authentication fails.

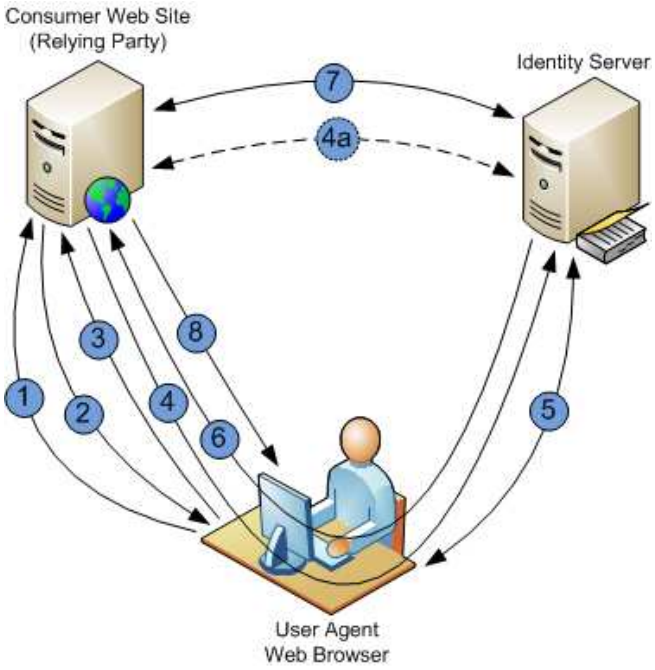

Figure 5. OpenID dumb mode (Adapted from (Rehman, 2008))

SMART MODE:

The smart mode is the same process as dump mode except step seven. At step seven Consumers can check and verify assertion data, based on the shared key on step 4-A. In smart mode, step 4A proceeds when Consumer wants to update caches or obtain it at the first time (Rehman, 2008).

E. SAML

Security Assertion Markup Language (SAML) is an XML based protocol for transaction of authentication and authorization information across domain boundaries. SAML addresses strong trust, high-value transaction, and privacy requirements for identity management. It enables users, applications, and services to communicate via XML messages (Maler & Reed 2008). In the other words, SAML provides a standard for transaction of user security information over insecure networks like Internet between identity provider and service provider domains. It describes set of rules and syntaxes for identity information transaction while it is flexible and customizable (Lewis & Lewis, 2009). As in Figure 6 illustrated, SAML solve the problem of accessing services and resources of a domain by another domain's users and systems; so, user A at domain A authenticates by authentication server of domain A and uses services of domain B. SAML ports the trust that domain A have granted to user A to domain B.

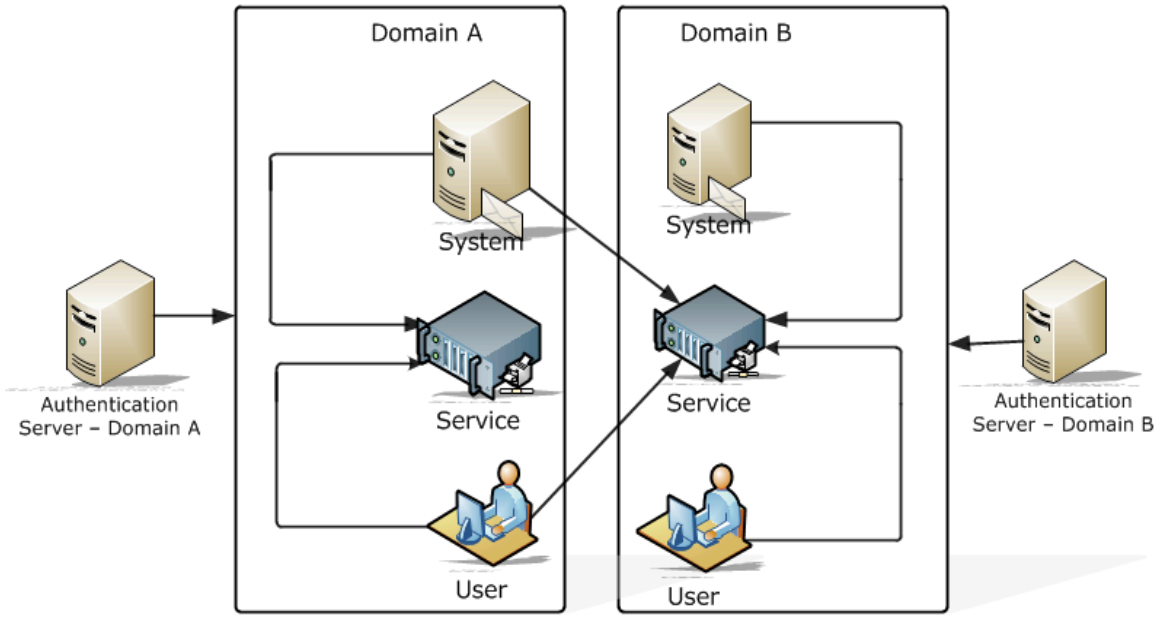

Figure 6. SAML overview

SAML defines three roles that are involved in SAML transactions: asserting party, relying party and subject. Asserting party is the identity provider that provides the user information. Relying party trusts assertion information from the identity provider and provides the services to user. Subject is the user with the identity that is going to be transacted (Lewis & Lewis, 2009).

SAML have four components: assertions, protocols, bindings, and profiles. Each layer of standards is customizable in order to implement specific organizational use-cases. SAML assertions get exchanged between SAML parties (Lewis & Lewis, 2009) - such as transaction between user A and service B in figure 6. Todorov (2007) mentioned that SAML have three assertions:

- **Authentication**: shows user's successful authentication to, authentication authority, it also may contain timestamp, type of credential, and etc.
- **Authorization**: may contain user's permission or access to an object, group membership, or any other information regarding to resources' authorization.
- **Attribute**: contain user's information like email address, telephone, and etc.

SAML describes request and response protocols for communicating between SAML parties; for example, Single Logout Protocol, within SAML, describes the flow of request and response in order to logout from all services by user. Another example is Authentication Request Protocol, which describes how service provider is able to request authentication or attribute assertion statement. SAML bindings map SAML protocols to other network protocols in order to transport assertion between SAML parties; for example, HTTP Redirect Binding relies on HTTP redirect messages to transport SAML assertions. And the last components of SAML are profiles; these components determine how assertions, protocols and bindings will cooperate to provide single sign on. For example Web Browser SSO Profile uses the authentication request protocol with any of HTTP redirect, HTTP POST or HTTP Artifact bindings. Another example is Single logout Profile that uses the Single Logout Protocol; this profile can logout the user form all the service providers using one logout function (Lewis & Lewis, 2009).

III. Research Approach

"A case study examines a phenomenon in its natural setting, employing multiple methods of data collection to gather information from one or few entities (people, groups, or organizations)." (Benbasat et al, 1987)

This research is a case study which we investigate different authentication and authorization methods in the context of mobility and security inside the Volvo IT. We gathered our information from Volvo IT employees. The context of research is bonded to Volvo; however, we tried to generalize and extend the foundings to make them suitable for other enterprise companies.

A. Research Setting

This study is a bachelor thesis, which is done by collaboration of IT university of Gothenburg and Volvo IT during ten weeks of research. Volvo IT was interested in how they can utilize different authentication and authorization protocols for securing their mobile applications in future. In this research we focus our investigation on finding possible relevant requirements regarding to accessing backend services by mobile devices for Volvo IT. Additionally, we did not consider financial and other barriers regarding to implementation of this technology.

B. Research Process

At first step of this research, we focus on the different architecture styles to detect any relation between them and mobile security. But, after extensive research on Service Oriented Architecture, Message Bus Architecture, and Event Driven Architecture, it turned out that there is not an obvious relation between them at implementation level; since these architectural styles are very abstract, any enterprise implementation of these are usually mixed to provide more functional and nonfunctional requirements. So, at this research we tried to bring our arguments and discussions to the context of Service oriented architecture, which is the current comment architecture style at Volvo.

Then, we studied related literature, which are aligned to the area of mobility and security. We started to look for the security protocols related to the authentication and authorization. For limiting our study to fit in the limited time that we had, we chose three protocols (OAuth, OpenID and SAML) which Volvo were interested in.

Then, we interviewed mobile application developers and system architects of Volvo IT in order to discover general requirements of mobile applications and specific requirements of access control in their systems. In the next step we collected the entire requirement together and generated proper requirement list. Finally, we checked capability of OAuth, OpenID and SAML against generated requirements; and by that we conclude if these protocols can add value to Volvo IT or not.

C. Data Collection

Interviews were the source of data that we gathered at Volvo IT. We have conducted four unstructured (Myers & Newman, 2007) interviews at Volvo IT. We had an interview with the mobile developers of Volvo, which gives us insight on major challenges of getting access control over the backend services of Volvo. The other interviews were mostly focused at discovering various scenarios in which these protocols can be suitable for Volvo IT. Since those

interviews were unstructured, we try to take note and gather related data at interview sessions.

D. Data Analysis

Our collected data was analyzed using *Qualitative Data Analysis;* we applied a process of collecting interview notes, analyzing notes, extracting initial requirements, investigation and studying new topics which emerged from previous interviews, plan for the next interview and refining our requirement list after each interview. After every interview the notes were put together in order to sync the outcome on each interview, then we tried to find relevant information; these information led us to a specific requirement and, consequently, to form the initial list of requirements. Also, we extracted new topics and areas, which needed to be investigated to fill the knowledge gap; furthermore, those topics and areas helped us to detect and remove unrelated data and requirements. As a result of this process, we keep the requirements within the scope of research. Finally, after the last interview, we collect all the requirements to form the final requirement list.

E. Research Limitations

Based on the fact that OAuth is a new protocol for handling access control there is not enough published material and article to investigate. Only few large enterprise like Google and facebook already have this technology in place for their systems and this technology, at the time being, is not adapted really well on the enterprise companies except major software developer (OAuth, 2011).

The other limitation for this research was related to the fact that Volvo is not adapted to the mobile industry in a proper way; because of that they do not have guideline and principal documents regarding to mobile developments. As a result, we could not include internal documents of Volvo in our research. So, we faced a hard time extracting the requirements.

Other specification was unclear and the scope was wide at first stage of research. At the beginning, we wanted to find the suitable combination of software architectures and security methods for mobile devices. Given that, we investigated software architectures like Service Oriented Architecture, Message Bus Architecture, and Event Driven Architecture. First, these architectures were wide area to investigate. Second, implementations of those architectures in enterprises were more or less mixture of them. So, our finding indicates that there is not any relation between security methods and architectures style in the case of Volvo.

IV. REQUIREMENTS

In this section we are going to present the requirements that we extracted from data analysis. We divided the requirements to two categories: security and usability requirements.

A. Security Requirements

This section concerned with the requirements that fulfilling them improves the security of the system.

1) Requirement 1 (Limited access rights)

One of the security concerns, mentioned by two Volvo architects, was that a mobile application should provides a method to implement access control to backend services; in the other words, authorizing users. Due to authorization access level for users, some services should be accessible and others should be inaccessible. So, the system should provide a list of services to the mobile application based on the user's access rights.

2) Requriement 2 (Shared mobile device between employees)

There are some organization working with Volvo that using Volvo mobile applications and services. These organizations are using Volvo services on an organization's mobile devices that are shared between their employees. These employees might have variety of access right to use Volvo services; it is a security concern for Volvo to prevent unauthorized user to access resources on behalf of another user.

3) Requirment 3 (Authentication without storing credentials)

One of Volvo IT architects mentioned that it is difficult for Volvo IT to keep all mobile users' credentials in Volvo IT databases. Currently, Volvo has some mobile applications and planning to deploy more applications to public. So, Volvo is seeking for a way to authenticate and authorize users without registering their credentials. One solution is relying on credentials of another company, which is suitable for more public backend services. So, system should be able to authenticate and authorize users based on credentials of either Volvo IT or another company.

4) Requiremet 4 (Authorization of third party applications)

In one of the interviews an architect from Volvo mentioned due to the rapid growth in mobile industry, in future, Volvo wants to facilitate the potential of using third party organizations to develop mobile applications for them. Therefore Volvo needs to find a way to grant third party developers accessing their backend services in a secure way in order to let them utilize Volvo's resources and services.

5) Requrirement 5 (Single logout)

Since there is more than one mobile service, user needs to login to different services. So, whenever user wants to log out, she should log out from all services that she was using. There is a possibility that user may forget to logout from all services; this poses a security risk. So, system should be able

to logout the user from all services when she logout from any service.

B. Usability Requirements

This section concerned with the requirements that fulfilling them improves the user experience of the system, which are, still related to the authentication, authorization and general security issues.

1) Requirement 6 (Long lived access)

Applications should not ask for credentials each time the user opens it. It should have an internal mechanism to save the state of connection with the backend servers. This will prevent annoying password prompt every time the user reopen the application.

2) Requirement 7 (Single sign on)

For a group of applications using the same identity provider, user should receive prompt for credentials only once. If a user opens an application and provides her credentials to the identity provider, she should not asked again for credentials from another application which is using the same identity provider for accessing services on backend. This concept is known as the single sign on.

3) Requirment 8 (Revoke application access)

Typically, there are many applications that are authenticated and authorized by a single user; such a user should be able to cut off any application's access, to her resources whenever she wants.

V. Discussion

In this section we are going to describe whether the above requirements could be addressed by the presented protocols' capabilities or not. This information can be used by Volvo or other enterprise to grasp a better understanding on which areas these protocols can be helpful for them; This understanding help them to make a decision whether or not to utilizing these technologies into their systems. The below requirements are in the same order as they presented in section IV.

A. Limited access rights

Based on the presented information on section II, OAuth is a protocol that covers *delegated authorization*; with this protocol you can authorize a mobile client or web application to use resources or connect to APIs on behalf of the resource owner. However, this protocol is not capable to provide and generate the list of services based on rights of the users. OpenID is an authentication protocol therefore it cannot act as the access control mechanism. Also SAML is a protocol for transferring authentication and authorization between different domains and it can only transfer the authorization decision, which has already been made. This requirement can be met by Role-Based Access Control systems (Windley, 2005) which none of mentioned security methods are capable.

B. Shared mobile device between employees

This requirement is beyond the scope of these protocols, it emphasizes on the security risk of the mobile devices, which can be physically shared. Due to nature of the problem, physical risk of security, OAuth, OpenID and SAML are not the security methods to utilize as the solution. However, biometric authentication (Windley, 2005; Tuyls et al., 2005) may implement in mobile devices to solve the problem; this strong authentication still can be implemented allied with OAuth and SAML.

C. Authentication without storing credentials

This requirement is related to the situation that Volvo wants to have some services available to the public, but they also want to have a sort of authentication without going through the process of registering users. In this case one solution to this problem could trust another identity provider to authenticate the users. In order to do that Volvo should develop their application with the ability to stand as the client concept in OAuth and get the identity credentials like id and email from the external identity provider like facebook or Google. In other words this kind of applications prompt users to give them access to their basic credential through other organizational credential databases (Boyd, 2012).

Another way to address this requirement is through OpenID protocol. OpenID enables authentication from an identity provider, which is running by another organization. Users can enter their OpenID URL, which they obtained, from another organization (i.e. facebook or Google) to enter the applications that has been created by Volvo (Rehman, 2008). This enables the user to login with their preferred identity provider and give to the Volvo the advantage of not storing every user's credentials. However they should consider the reliability of the identity providers that they are going to trust. We are not going to cover that which organization is suitable for Volvo to trust because this matter is out of scope of this research.

D. Authorization of third party applications

The first impact of third party developer to the system is the potential security breach; when a user inputs her credentials to the third party application in order to access Volvo backend services, such credentials may store, in an unsecure way, to mobile device. To avoid this problem OAuth suggest a solution.

OAuth suggest that the third party application, should not have resource owner's credentials unless there is high degree of trust between resource owner and client (Like official applications). If there is not high degree of trust between client and resource owner, client can get an access token to access the resources instead of resource owner's credentials. However, without OAuth, client must access to resource owner's credentials in order to access resources and services; with the access token in OAuth, there is no need to store resource owner's credentials for later authorization

(Recordon et al., 2012). As in_figure 7 illustrated, access token goes to the resource server instead of credentials.

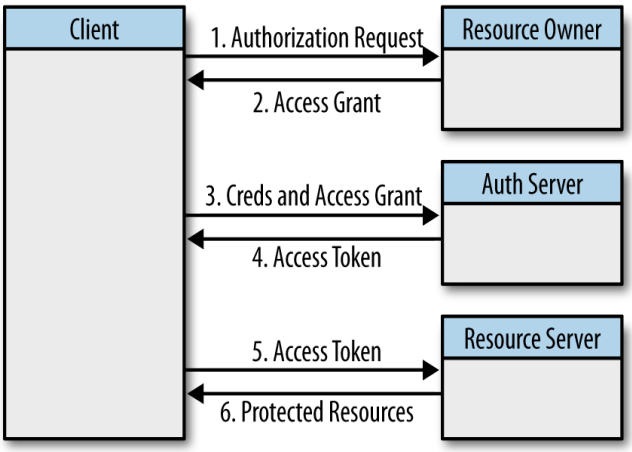

Figure 7. Simple OAuth flow (Adapted from (LeBlanc, 2011))

Furthermore, OAuth suggests setting up an application server in order to improve the security. As we mentioned in OAuth flow section, there are two types of flows - Authorization code and implicit grant - that can be implemented in this case. The Authorization code has significant security improvement compared to implicit grant. This improvements lie in the ability to hide the access token from user agent, since the token is stored on the application server, user agent cannot access it. As a result of that, there is no risk of token exposure by user agent. On the other hand, implicit grant improve efficiency due to deducted number of round trips to get access token. Therefore, usability should be overweight the security to justify implicit grant implementation otherwise Authorization Code should be implemented (Recordon et al., 2012). Because of evolving third party developers that are not fully trusted by Volvo, in the process of mobile application development, Volvo can take benefit of OAuth by implementing Authentication Code flow that provide more security.

E. Single logout

The only protocol that can meet this requirement is SAML. After an application has authenticated to an identity provider, a session may be established between that application and the identity provider. In this case, identity provider may issue assertions to service provider also the identity provider can act as the session authority as well. If a system or user wants to logout, it can be satisfied with single logout profile of SAML. For this protocol <LogoutRequest> and <LogoutResponse> messages are defined (Hughes et al., 2005). In figure 8, sequential requests and responses to logout from all service are illustrated.

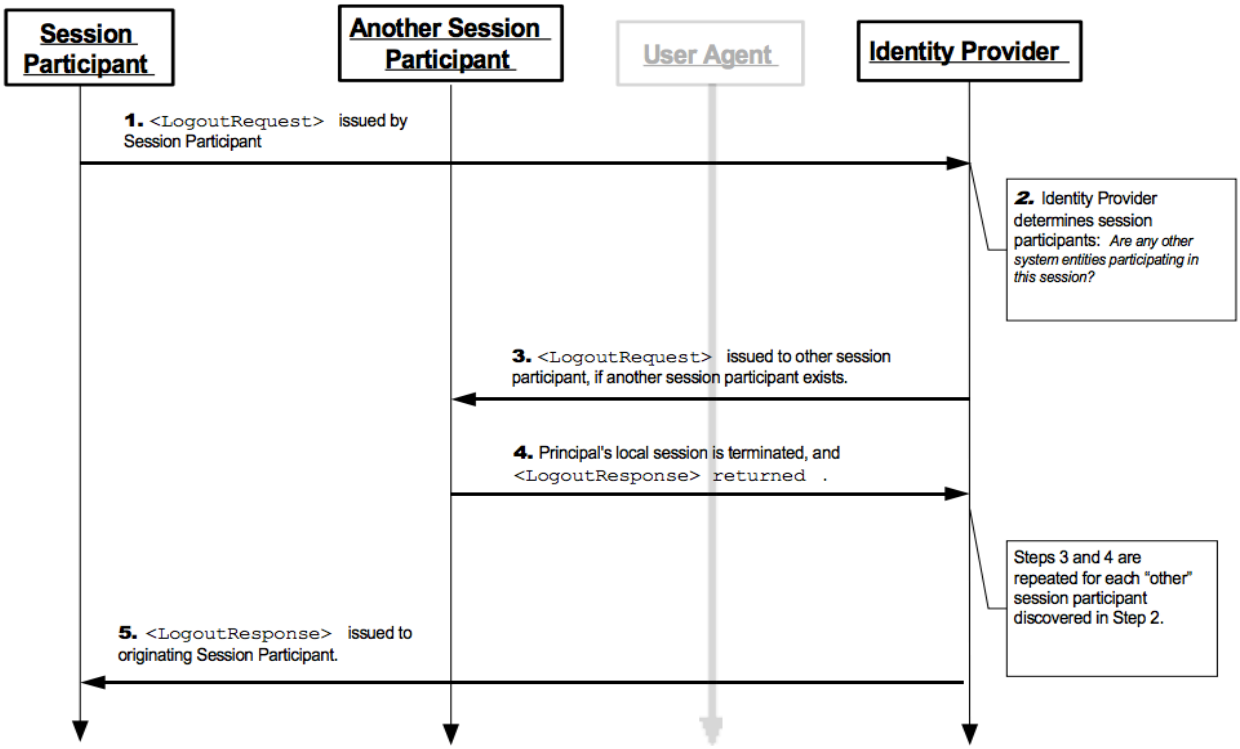

Figure 8. Single logout sequence diagram

F. Long-lived access

OAuth covers this requirement in two different ways; one way is by issuing the access token with long life span, which grant access to the resources and services even an unlimited access. The other way is through refresh token. There is a security problem with the first way and having one-time long-lived access tokens, and that is because the OAuth typically uses bearer tokens without signature to connect to APIs. Also most of the time an OAuth token may provide access to multiple APIs, compromise of a token can give the attacker access to multiple services and could be an extensive security threat for the system.

On the other hand, refresh tokens reduce the security vulnerability of the system; in case a token is compromised then the token is going to expire and new token is going to be replaced by it. This process is accomplished with an HTTP POST request and with sending the refresh token (Boyd, 2012).

G. Single sign on

Both OpenID and SAML provide solutions for SSO. As in section II.D mentioned, OpenID have two modes to handle SSO. In OpenID, Identity provider is the only service that user should sign on; given that, user can access other backend services offering by other consumers without signing on.

SAML have similar solution to this requirement as followed (Hughes et al., 2005):

1. User send a HTTP request to service provider
2. Service provider resolve the identity provider
3. Service provider send a <AuthnRequest> message to identity provider
4. Identity provider identify the user (Authenticate the user or use existing session)
5. Identity provider send the <Response> to service provider
6. Based on the <Response> service provider deny or grant the access to user

H. Revoke application access

OAuth has the advantage over the basic authorization method based on the fact of having tokens instead of password.

In the basic authorization, which uses the password for accessing the APIs and resources of backend, the problem is by giving password, the only way for user to revoke access to her resources on the backend is to change the password.
And if user has multiple applications using one service endpoint, then after changing password, to revoke one of the application's accesses to her resources then she should provide new password for all the other applications that she wants to still have access to her resources. This is obviously spoiling the user experience; they want to have an easy way to cut the access of an application to a resources or services.

In OAuth, because the access is made through tokens and not passwords, when a user wants to revoke an application's access, the only thing that system need to do is to discard that specific token which grants access to that application (Boyd, 2012).

I. Discussion Summary

The overall picture of what these protocols can satisfy in term of requirements of section IV is presented in Table 1. The first two requirements (limited access right and shared mobile device between employees) are the ones that none of the protocols can handle. But for the rest of requirements we have at least one protocol to satisfy them. Therefore in order to satisfy the maximum functionality, a combination of these protocols could be the solution. There are some projects that are combining OAuth to OpenID or SAML; for example OpenID Connect is a combination of OpenID authentication method with OAuth capabilities (AB/Connect Work Group, 2012). At another project, Developerforce is combining SAML and OAuth (Developerforce, 2012). Since those projects are in the initial stage of development and because of the low reliability we are not going to investigate them. But they could be considered for the future solutions.

OAuth is not a full security solution for enterprise but it can handle *delegate authorization* in a secure way. It is suitable for enterprise companies that have numerous mobile applications or mobile users. That's why; OAuth is mainly used by social websites like Google and Facebook. There is still a gap between implementation and standardization - at the time of composing this paper; OAuth 2.0 still is a RFC draft (Recordon et al., 2012). Yet, some companies are using the implementation based on the draft version currently (OAuth, 2011).

Furthermore, OpenID is web friendly protocol that provides a solution for SSO; OpenID can be implemented and integrated to mobile web enabled applications. SAML also have solution for SSO. However, SAML is more enterprise friendly, provides more security, and is more compatible with other security methods, on the other hand OpenID bring more simplicity and scalability (Maler & Reed, 2008). So, these two protocols can be implemented and utilized for securing mobile devices according to the specific use case.

VI. Conclusion and Future Works

Providing backend service to new mobile user is bring up some issues to enterprises; most important issue is how to secure these services to mobile devices over the insecure network like internet. This problems and issues are mainly related to authentication and authorization aspects of security. How can a user get authenticated outside the trusted domain and how services and resources should be accessible

TABLE 1. COMPARISION BETWEEN OAUTH, OPENID AND SAML AGAINST THE REQUIREMETNS

	Req 1	Req 2	Req 3	Req 4	Req 5	Req 6	Req 7	Req 8
OAuth	✗	✗	✔	✔	✗	✔	✗	✔
OpenID	✗	✗	✔	✗	✗	✗	✔	✗
SAML	✗	✗	✗	✗	✔	✗	✔	✗

by user and in what level? These are the questions that should be taken into consideration in order to securing mobile devices integration to enterprise backend services.

To answer the research question, we investigated three protocols that bring security to the process of integrating mobile devices, to enterprise backend services. OAuth, SAML, and OpenID are protocols that have defined set of rules for authentication and authorization. We performed a case study at Volvo in order to find out to what extent these protocols can satisfy the requirements of Volvo regarding to providing their services for mobile devices. We found eight requirements, which are categorized into user experience and security.

None of the security methods can meet all of the requirements. However, there are possibilities of combining these security methods together to meet more requirements. These combinations need more research and study as a future research. For requirement one, Role Based Access Control system is the solution for limited access control (Windley, 2005). To meet this requirement with others, we suggest to investigation of what protocol can support Role Based Access Control and if they can be combined with SAML, OAuth, and OpenID. Another area to research is requirement 2, how mobile devices that are shared between employees of an organization can be authenticated and authorized. One area for future investigation could be biometric authentication for meeting this requirement (Windley, 2005; Tuyls et al., 2005). Finally for finding suitable solution for Volvo we suggest to perform another investigation on financial and implementation barriers that we have not considered in this article.

REFERENCES

AB/Connect Work Group | OpenID. (n.d.). Retrieved May 2, 2012, from http://openid.net/wg/connect/

Benbasat, I., Goldstein, D. K., & Mead, M. (1987). The Case Research Strategy in Studies of Information Systems. MIS Quarterly, 11(3), 369–386.

Boyd, R. (2012). Getting Started with OAuth 2.0 (2012th-02 ed.). O'Reilly Media.

Developerforce.com. (n.d.). Single Sign-On for Desktop and Mobile Applications using SAML and OAuth - developer.force.com. Retrieved May 20, 2012, from http://wiki.developerforce.com/page/Single_Sign-On_for_Desktop_and_Mobile_Applications_using_SAML_and_OAuth

Fitzgerald, J. (2009). Managing mobile devices. Computer Fraud & Security, 2009(4), 18–19.

Hughes, J., Cantor, S., Hodges, J., Hirsch, F., Mishra, P., Philpott, R., & Maler, E. (2005, March). Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0. Retrieved from http://saml.xml.org/saml-specifications

LeBlanc, J. (2011). Programming Social Applications: Building Viral Experiences with OpenSocial, OAuth, OpenID, and Distributed Web Frameworks (1st ed.). O'Reilly Media.

Lewis, K. D., & Lewis, J. E. (2009). Web Single Sign-On Authentication Using SAML. *International Journal of Computer Science Issues (IJCSI)*, *2*, 41–48.

Maler, E., & Reed, D. (2008). The Venn of Identity: Options and Issues in Federated Identity Management. *Security Privacy, IEEE*, *6*(2), 16 –23.

Myers, M. D., & Newman, M. (2007). The qualitative interview in IS research: Examining the craft. Information and Organization, 17(1), 2–26. doi:10.1016/j.infoandorg.2006.11.001

OAuth / OAuth 2. (2011, July). Retrieved May 26, 2012, from http://wiki.oauth.net/w/page/25236487/OAuth%202

Recordon, D., Hardt, D., & Hammer-Lahav, E. (2012, May). The OAuth 2.0 Authorization Framework. Retrieved May 18, 2012, from http://tools.ietf.org/html/draft-ietf-oauth-v2-26

Rehman, R. U. (2008). Get Ready for Openid (1st ed.). Conformix Technologies Inc.

Sandhu, R. S., & Samarati, P. (1994). Access control: principle and practice. *Communications Magazine, IEEE*, *32*(9), 40 –48.

Teng, C.-C., & Helps, R. (2010). Mobile Application Development: Essential New Directions for IT. Information Technology: New Generations (ITNG), 2010 Seventh International Conference on (pp. 471 – 475).

Todorov, D. (2007). *Mechanics of user identification and authentication*. Auerbach Publications.

Tuyls, P., Akkermans, A., Kevenaar, T., Schrijen, G.-J., Bazen, A., & Veldhuis, R. (2005). Practical Biometric Authentication with Template Protection. In T. Kanade, A. Jain, & N. Ratha (Eds.), Audio- and Video-Based Biometric Person Authentication, Lecture Notes in Computer Science (Vol. 3546, pp. 1–53). Springer Berlin / Heidelberg.

Unhelkar, B., & Murugesan, S. (2010). The Enterprise Mobile Applications Development Framework. IT Professional, 12(3), 33 –39.

Windley, P. J. (2005). *Digital Identity* (1st ed.). O'Reilly Media.