Using externals IdPs on OpenStack: A security analysis of OpenID Connect, Facebook Connect, and OpenStack authentication

Glauber C. Batista, Maurício A. Pillon, Guilherme P. Koslovski, Charles C. Miers
Graduate Program in Applied Computing – Santa Catarina State University – Joinville – Brazil
glauber@colmeia.udesc.br, {mauricio.pillon, guilherme.koslovski, charles.miers}@udesc.br

Nelson Mimura Gonzalez
IBM Watson Research Center, Yorktown Heights, NY, USA
nmimura@us.ibm.com

Marcos A. Simplicio Jr.
University of São Paulo, São Paulo, SP, Brazil
mjunior@larc.usp.br

***Abstract*—The installation and configuration of cloud environments has increasingly become automated and therefore simple. For instance, solutions such as RedHat RDO and Mirantis Fuel facilitate the deployment of popular computational clouds like OpenStack. Despite the advances in usability, effort is still required to create and manage multiple users. This is of particular relevance when dealing with sensitive information, a somewhat common case for private clouds. To alleviate this burden, many clouds have adopted federated Single Sign-On (SSO) mechanisms for authenticating their users in a more transparent manner. In this work we analyze the practical security of an OpenStack IaaS cloud when combined with either OpenID Connect (using Google as IdP) or Facebook Connect (using Facebook as IdP). The criteria used in the analysis comprise the ability to provide data encryption, the risks involved in the use of an external IdP, and improper access control. We identify potential issues regarding these solutions and we propose approaches to fix them.**

***Index Terms*—OpenID Connect, Facebook Connect, OpenStack, Keystone**

I. Introduction

Cloud computing is a model that allows for ubiquitous, convenient, and on-demand access to a set of configurable resources that can be quickly provisioned and released with minimal effort [1]. OpenStack [2] is currently considered one of the most prominent open source cloud platforms [3], [4]. This prestige is reflected by the activeness of its community of developers and by its worldwide utilization for building large (including federated) cloud environments [2].

Since cloud environments can become quite complex, cloud solutions usually provide automated tools to facilitate their management. One of these tools, which is of particular interest in this work, refers to Single Sign-On (SSO) mechanisms [5], [6]. SSO solutions provide a unique identifier to each user, so they can authenticate themselves to any service using this identifier and a single set of credentials registered on an Identity Provider (IdP) [7]. This approach creates a user-centric authentication environment, so even when different services require strong authentication: (1) users do not need to create and manage multiple credentials for each service; and (2) the services themselves are not required to securely store and manage the users' credentials, offloading most of the authentication-related tasks to the SSO system.

With the growing interest of incorporating SSO into cloud environments, some organizations started to adapt their own existing identity systems to use them as SSO solutions in their own domains. However, federated authentication protocols such as OpenID Connect and Facebook Connect quickly became prefered approaches, since they provide further support for service integration. This led to the creation of cloud environments that support a more dynamic authentication and authorization process, easier to integrate with internal or external services. Nevertheless, the convenience brought by these tools is counterbalanced by potential security risks, which need to be well understood to prevent abuse by malicious entities accessing the cloud. To address the need of correctly integrating of SSO mechanisms into the cloud, this work analyzes the practical security of an OpenStack IaaS cloud when combined with popular SSO solutions, namely OpenID Connect (using Google as IdP) and Facebook Connect (using Facebook as IdP). More precisely, we use OpenStack's module responsible for identity management, named Keystone, which provides an OpenID Connect plugin and supports third party IdPs. We also develop a proxy for Facebook Connect on OpenStack's Horizon module, since this SSO mechanism is not natively supported. Building upon the analysis provided in [8], the selected criteria for evaluating both solutions comprise the ability to provide data encryption, the risks involved in the use of an external IdP, and improper access control.

The rest of this document is organized as follows. Sec. II briefly discusses the OpenStack architecture and components, identifying modules related to authentication (in particular, Keystone). OpenID Connect is then described in Sec. III, whereas Facebook Connect design is explained in Sec. IV. Sec. V discusses related works. After Sec. VI defines the criteria employed in our security evaluation and Sec. VII describes the corresponding testbed environment, the results obtained are analyzed in Sec. VIII.

II. OpenStack and Keystone

OpenStack is a cloud solution that controls several computational resources in a data center [2], [9]. OpenStack is composed by services, each one responsible for a specific set of roles, such as handling storage, networking, computing, databases, telemetry, and orchestration. Keystone is the module responsible for identity management as well as authentication and authorization of operations related to all OpenStack services. Keystone employs a Role-Based Access Control (RBAC) model for providing high-level authorization which transforms attributes into roles. The authorization itself occurs in a decentralized manner based on projects and roles of each user [6]. Keystone is extensible, allowing SSO authentication by means of the `OS-FEDERATION` extension. When running Keystone on an Apache HTTP Server, which allows the installation of plugins and modules, the `OS-FEDERATION` extension can be implemented in different manners. In particular, when using an OpenID Connect IdP, the `mod_auth_openidc` module handles the authentication process [10]. OpenID Connect uses claims between the communication parts. These claims require and provide attributes from registered users, and take the form of encrypted and signed JSON documents [6], [10]. Mapping rules convert remote user attributes into local attributes obtained through claims. More precisely, the mapping specifies which users can have access to a given service as well as which group and project they should be allocated to.

III. OpenID Connect

OpenID was created as an open source authentication mechanism for providing user-centric identity management in a decentralized manner, and is the basis for many SSO authentication solutions [9], [11], [12]. OpenID Connect is the 3rd generation of OpenID, operating on top of OAuth 2 authorization protocol and using REST/JSON direct messages protected by a Transport Layer Security (TLS) tunnel [13]. OAuth 2 is beneficial for this purpose as it provides a generic framework in which a user U can delegate to a service S_1 the right to access (part of) U's resources managed by a second service S_2. This process does not require U to share any credentials with S_1, so U has complete control over the data being shared and for how long [12], [14]. OpenID Connect creates an SSO system by providing an identity layer upon OAuth 2, therefore the delegation process involves accessing the user's identifiable attributes. This integration is more secure than the disjoint use of these protocols, making OpenID Connect less susceptible to phishing, Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF) attacks [11]. Besides usability, security is one of the main reasons why large organizations (e.g., Facebook and Google) use OAuth and related technologies to enable their users access to contracted services [14], [15].

An example scenario is useful to explain how users interact with OpenID Connect. A user U wants to access service `example.com` offered by a Service Provider (SP). The service is accessed via his/her User-Agent (UA) (e.g., a web browser). Instead of filling a signup form, U provides an identifier (e.g., a URL) that represents his/her identity. The SP then starts the discovery process to verify U's ownership of the identifier. First, the SP identifies that U can be authenticated via the IdP `openid.idp.com` with username `user`. Then, U is redirected to that IdP and provides the proper credentials (e.g., a username and corresponding password). U also authorizes the IdP to access (part of) his/her personal information. Finally, the IdP redirects U to the SP, which in turn provisions a new account to that user. To illustrate this process in the context of OpenStack, Figure 1 shows its execution when the SP is the Keystone module. At the end of the authentication process in OpenStack, the user receives a token with a defined scope, specifying the domain, groups, and projects to which he/she belongs. This token is used in every subsequent request sent by the user to OpenStack's services.

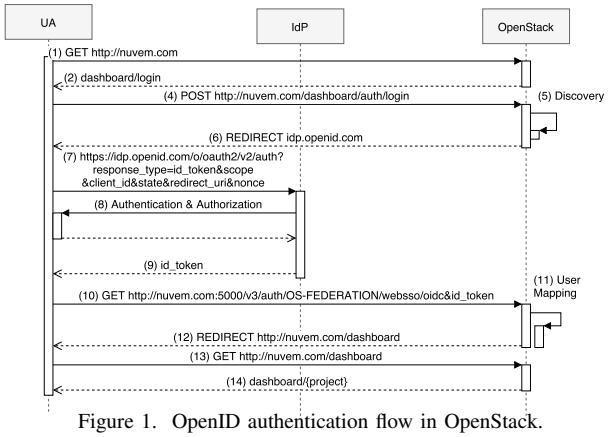

Figure 1. OpenID authentication flow in OpenStack.

IV. Facebook Connect

Facebook Connect is an authentication platform based on a RESTful Application Programming Interfaces (APIs) that allows applications to access Facebook services via GET or POST requests [16], [17]. The integration between applications and Facebook is done using Javascript SDK or direct requests, both implemented by the developer. One of the most important APIs is the Open Graph (http://ogp.me/), which allows applications to access user objects (e.g., photos, friends, groups), and the connections among them in the Facebook social network [15]. Similar to OpenID Connect, Facebook Connect is based upon OAuth 2, so users can control which data are accessible by applications. Although relatively easy to implement, Facebook Connect is not an open protocol and its detailed operation is not officially available. For this reason, in this study we developed a proxy, named Openstack-Facebook (OFL), that allows the integration of Facebook Connect with OpenStack. Since OFL acts as an authentication proxy for Horizon, Keystone is not directly involved in the authentication flow with Facebook IdP. Keystone is only necessary at the end of authentication process, when it creates or authenticates the user. After this, the user receives a scoped token which authorizes him/her to access the corresponding resources. The message flow among UA, OpenStack and Facebook is presented on Figure 2.

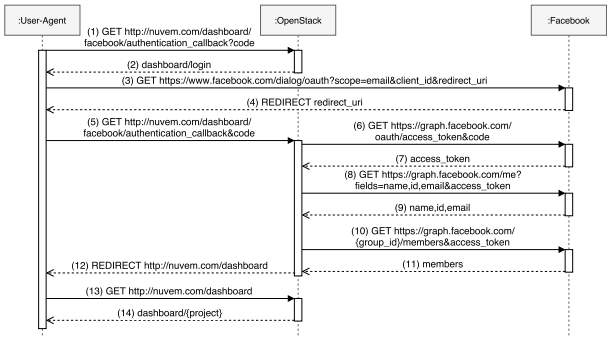

Figure 2. Facebook Connect authentication.

Figure 2 shows the authentication process launched when the user access the cloud's dashboard *(1)*. The server responds with an HTML page in which Facebook is presented as authentication option *(2)*. By selecting this option, the user is redirected to Facebook's authentication page *(3)*. Once authenticated, the user receives an authorization code and is redirected to OpenStack again *(4)*. After this code is provided to OpenStack by the UA *(5)*, OpenStack queries Facebook *(6)* to obtain an `access_token` from the `code` parameter. The `access_token` allows OpenStack to obtain *(8,9)* the user's unique identifier, which can be used in the creation of a new account (sign-up) or in the authentication of an existing account (sign-in). The authentication via OFL comprises an extra step to identify the user as a member of a specific group. For this purpose, OpenStack obtains the list of group members *(10,11)* and verifies the user's identifier against this list. If the user belongs to the group, the authentication process is successful and he/she is redirected to the corresponding project *(12,13,14)*.

V. Related Work

SSO authentication has been much debated in the last years, leading to the emergence of many solutions aiming to create a user-centric environment [18]. Among them, OpenID 2.0 and OAuth 2 became the most prevalent protocols for authentication and authorization, respectively [19]. This success is probably due to the fact that security and privacy concerns have always been important aspects in the evolution of these protocols. As an example, one recommendation highlighted in OAuth 2 specification [20] is that the authorization endpoint must be protected by TLS. Otherwise, authorization codes could be intercepted, making the endpoint vulnerable to attacks such as session swap: suppose an attacker builds a page that contains a request with intercepted tokens; when the victim accesses the page, its UA automatically sends a GET or a POST request to the SP, but the session is bound to the attacker's account. The system's security against attacks involving captured tokens also depends on the type of tokens employed. Namely, *bearer tokens* allow anyone who holds the token to access the protected resources. In comparison, *Message Authentication Code (MAC) tokens* require the user to demonstrate knowledge of a cryptographic key, so they are more secure against capture [21], [22].

The broad utilization of OAuth and OpenID motivated many studies that evaluate security aspects related to their design and implementation [8], [21], [23], [24]. Even though these studies are relevant to understand the security of SSO mechanisms based on these protocols, OpenID Connect actually addresses many of the vulnerabilities identified in OpenID 2.0 and OAuth 2 [11]. Hence, it is important to consider the security best practices identified in these studies and understand how they apply to the environment in which they are implemented. To the best of our knowledge, this is the first work that does so (for OpenID and Facebook Connect) in the specific environment of OpenStack-based clouds.

An interesting analysis focused specifically on OpenID and Facebook Connect, albeit not in the context of clouds, is presented in [7]. The main issue thereby identified resides in the single authentication option with Facebook, which allows the IdP to track all websites visited by the user. The authors propose a solution based on the possibility of choosing other IdPs, but recognize that the trend of Internet services is to use a limited set of IdPs (e.g., Facebook or Google). Unfortunately, however, an in-depth analysis could not be performed in [7], since: (1) there is not much publicly available information about the protocol's authentication flow; and (2) the captured data actually differs from the authentication flow specified by Facebook, which is likely due to internal, unreported changes in the protocol.

VI. Analysis Criteria

The adoption of well-defined good practices is crucial for the successful deployment of SSO mechanisms in computational clouds. One important aspect to note is whether the solution uses TLS for data channel encryption, as recommended in the specification of OAuth 2. This issue applies both to OpenStack's OpenID Connect plugin and to Facebook Connect's implementation. Another task refers to the privacy implications of using an external IdP in an OpenStack environment. After all, as pointed out by [7], [23], IdPs can store information about the user and, eventually, track their interactions with the cloud. Therefore, it is useful to review the privacy policy of Google and Facebook, used as IdPs in this work. It is also necessary to verify the possibility of user impersonation, since CSRF attacks and interception of packets can become an issue in this scenario [21], [24]. Finally, one potential issue particular to OpenID Connect is that data pertaining to an old user can be accessed by new users who assume the same OpenID identifier. This can be avoided if OpenID identifiers are not reused in the system, a recommendation followed by Google [25]. However, it is necessary to verify how the SP (in our case, OpenStack) handles the situation in which different users present the same OpenID identifier, i.e., whether or not the end up accessing the same account.

VII. Tests Environment

The testbed employed in the experiments comprises three servers: a controller node, a network node, and a computing node, running RDO OpenStack Liberty on GNU/Linux CentOS 7. The SSO environment authentication uses Google and Facebook IdPs to authenticate users. The cloud controller node has two network interfaces: one for external access and one for management. The network node has three interfaces:

one with Internet access for the virtual machines (VMs), one for management, and one for internal traffic of the VMs. The compute node has two network interfaces: one for management and one for internal traffic to VMs. The addresses for the management and internal traffic networks are `10.0.0.0/24` and `172.16.10.0/24`, respectively. The controller responds by the fictitious Fully Qualified Domain Name (FQDN) `www.cloud.com`. Keystone is located on the controller node and all services are authenticated by the `http://10.0.0.1:5000` address. The internal and administrative requests of the API are performed via the management network. Public API requests are performed via the external network, which is connected to one of the controller interfaces. To enable OpenID Connect in Keystone, the plugin `mod_auth_openidc` has to be installed and Keystone Apache file has to be configured to accept SSO authentication from the IdP. Since Google is used as IdP, it is necessary to create the application's OAuth 2 credentials in the Google Developers Console. It is also necessary to configure Keystone to enable OpenID Connect plugin, as well as configure Horizon to display the SSO authentication option on the home page. It is necessary to create a reference to the IdP, so it can be used in the authentication process, and configure the corresponding mapping rules. In the case of Facebook Connect, it is necessary to install the OFL tool and configure Horizon to enable authentication through the new proxy. Similar to the Google setup, the application's OAuth 2 credentials in the Facebook Developers panel must also be created. Finally, an auxiliary database is created to store additional user information provided by Facebook.

Traffic analysis was performed based on data collected by `tcpdump`, listening on the controller's external network interface (TCP/80, TCP/443, and TCP/5000). The traffic analysis experiments were repeated ten times to ensure that the flow of requests follows the same pattern, as well as to verify the nature of the data exchanged among the involved parties.

VIII. Security Analysis

In what follows, we discuss the results obtained when considering the criteria defined in Sec. VI.

A. Data Encryption

As previously mentioned, OpenID Connect and Facebook Connect are built upon the OAuth 2 protocol and use communication channels protected by TLS [7], [21], [26]. In addition, both protocols use short-term tokens when providing access to resources. This reduces the risk of data exposure and user impersonation in case those tokens are somehow intercepted by attackers (e.g., via a man-in-the-middle attack) [27]. To confirm the veracity of this protection, the testbed was configured so that the packets exchanged during the authentication process could be captured and analyzed.

1) Communication between UA and OpenStack: As mentioned in Sec. II, Keystone is the identity management component of OpenStack. By default, the Keystone module has an endpoint listening for requests on TCP/5000 port. For OpenID Connect, Keystone is responsible for exchanging messages with the IdP and the SP. In the case of Facebook Connect with OFL, Horizon is responsible for the initial authentication step, but it transfers the process to Keystone for the user account's creation or authentication. Hence, even though the message flow for each IdP is different, the packet capture can be performed on the same ports. Namely, `tcpdump` was configured to capture the packets exchanged by the UA and the SP on ports TCP/80 and TCP/5000, when TLS is disabled in OpenStack, and ports TCP/443 and TCP/5000, when TLS is enabled. The captured packets are then compared to verify whether or not TLS is protecting the connections.

a) OpenID Connect: Figure 3 exemplifies the communication sequence between UA and OpenStack during the authentication process. The captured messages show the data of each request in the diagram.

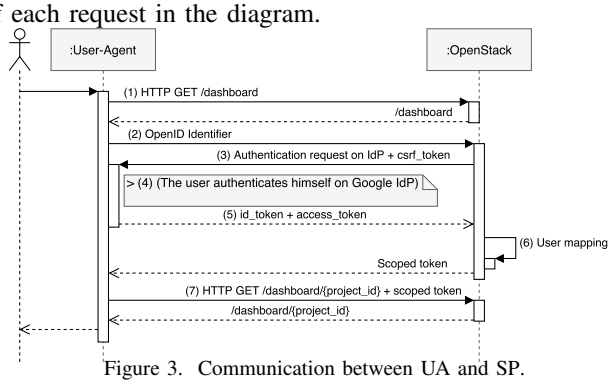

Figure 3. Communication between UA and SP.

(1) User, via UA, requests the dashboard to authenticate to OpenStack; *(2)* User selects OpenID Connect to authenticate; *(3)* OpenStack redirects the UA to Google's IdP; *(4)* User authenticates with Google, which returns `id_token` and `access_token`; *(5)* `id_token` and the `access_token` are redirected to OpenStack; *(6)* Through the `id_token`, which is a JSON Web Token (JWT), OpenStack obtains the necessary claims to authenticate the user, such as `iss`, `email` and `name`. The `access_token` is used if there is a need to get other user's claims through API calls (e.g., verify if the user belongs to a particular group); and *(7)* After authentication and user mapping, UA redirects the user to the dashboard of his/her project.

Collecting information without TLS confirms that the data is sent with no protection. Consequently, an eavesdropper can obtain the information required to impersonate users. Even without TLS, though, Keystone uses a token called `csrftoken`, which prevents CSRF attacks. Google also requires this token prior to the authentication request in order to protect the user's account against CSRF attacks. The experiment with TLS, on the other hand, revealed that the achieved security level is insufficient. Although Horizon uses TLS to protect its traffic, Keystone does not. As a consequence, all traffic going through port TCP/5000 is unprotected and can be captured by an attacker, including the `id_token`. Therefore, to ensure that the relevant endpoints communicate securely, TLS has to be explicitly enable on Keystone as well.

b) Facebook Connect: The message flow between UA and OpenStack during the authentication process is illustrated by Figure 4.

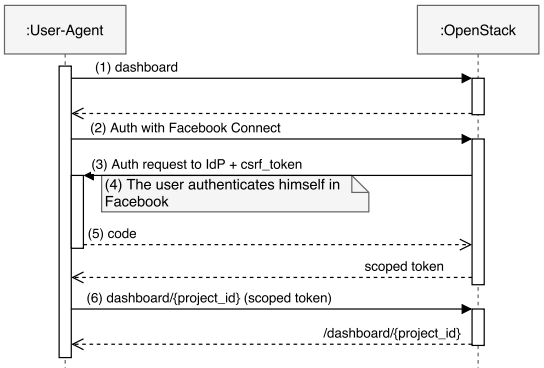

Figure 4. Data flow between UA and OpenStack.

(1) User, via UA, accesses OpenStack's dashboard; *(2)* User selects Facebook Connect to authenticate to the cloud; *(3)* OpenStack redirects the UA to Facebook's IdP; *(4)* User authenticates him/herself in Facebook, which returns a `code` token; *(5)* The `code` token is redirected to OpenStack; *(6)* OpenStack uses `code` to obtain the user's `facebook_id` and `access_token`. At this stage, OpenStack uses the `access_token` the user to request the user's e-mail to Facebook, necessary for the account creation; and *(7)* After the user is authenticated or have a new account created, the UA is redirected to the corresponding project.

In the experiment without TLS, the entities exchange unprotected data that can be intercepted for malicious purposes, especially the `code` parameter. However, according to Facebook(https://developers.facebook.com/docs/facebook-login/manually-build-a-login-flow), the `code` token is unique and generated for each authentication request. Therefore, even if intercepted, the `code` cannot be reused for other operations. The experiment also revealed the utilization of the CSRF token, as well as the protection of the communication channel via TLS. However, unlike OpenID Connect, the configuration of TLS in OpenStack Horizon is sufficient to prevent data capture from IdP. Thus, even if the attacker captures the user's `code`, it cannot be used in new authentication towards the service.

2) Communication between UA and IdP:

a) OpenID Connect: When using a particular service offered by an SP, the user account must be authenticated and the SP must be authorized to access a user's data. The authentication and authorization flow is composed by the steps presented in Figure 5.

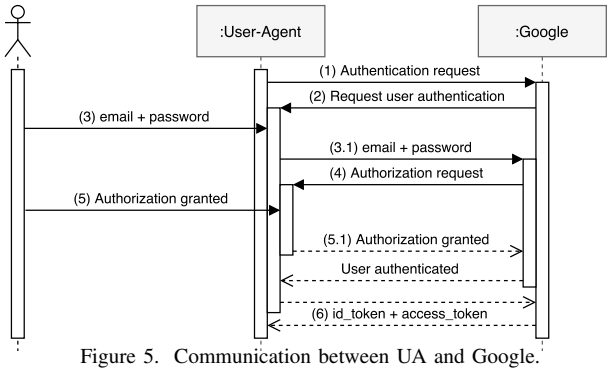

Figure 5. Communication between UA and Google.

OAuth 2 is responsible for requesting user permission before authorizing the application to access the requested information. This is typically accomplished by presenting a consent screen to the user, which lists the information the application wants to access. *(1)* OpenStack initiates the authentication process; *(2)* Google shows its usual login page; *(3)* User provides his/her credentials; *(4)* Google requests user to authorize the application via a consent screen, listing which data will be accessible by OpenStack; *(5)* User authorizes OpenStack to access his/her information; and *(6)* Google returns `id_token` and `access_token` to UA, which relays the tokens to OpenStack.

The user must agree to Google's terms of service and privacy during the authorization process. This allows Google to collect information about the accessed services, leading to privacy concerns. Another potential issue, identified in [24], is that CSRF attacks can be performed due to Google's "automatic granting" policy. Even though this vulnerability has already been addressed by Google itself, the implication is that OpenStack should verify every other IdP regarding this issue.

b) Facebook Connect: To authenticate to applications using Facebook, the user must authorize the application to access his/her Facebook information via OAuth 2. The authentication process may request additional information via specific APIs. This authentication process, which involves UA and IdP, is illustrated in Figure 6.

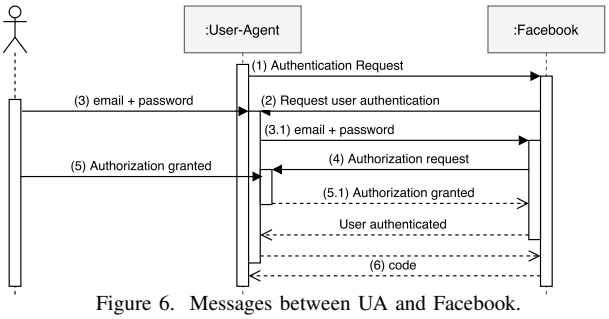

Figure 6. Messages between UA and Facebook.

(1) OpenStack initiates the authentication process; *(2)* Facebook requests the user to authenticate; *(3)* User provides the necessary credentials; *(4)* Facebook displays the application authorization page, informing the data being requested by the application; *(5)* User authorizes the application to access the requested data; and *(6)* Facebook returns `code` to UA, which is forwarded to OpenStack.

Similarly to Google's authentication and authorization process, the user must agree to Facebook's terms of service and privacy policy. This can also lead to privacy issues, allowing the IdP to track the user's activities. Moreover, Facebook is the only IDP in Facebook Connect, therefore it is not possible to a less invasive option for this particular authentication model.

3) Communication between OpenStack and IdP: In the communication channels between UA / SP and UA / IdP, the operation of OpenID Connect and Facebook Connect protocols is similar. However, the message flow is different in the communication between SP / IdP, particularly on how the users' data are obtained from the IdP.

a) OpenID Connect: The communication between OpenStack and Google IdP occurs by redirecting requests through the UA. Even though OpenID Connect's configuration requires a Uniform Resource Locator (URL) for the discovery process, OpenStack only accesses the document to obtain the information needed to authenticate the user in IdP (e.g., endpoints, and supported scopes). Thus, there is no direct traffic between OpenStack and Google's IdP during the authentication process. Data capture experiments revealed that no packet was exchanged directly between Keystone and the IdP, which reinforces this statement. Figure 7 lists the messages exchanged between OpenStack and Google IdP.

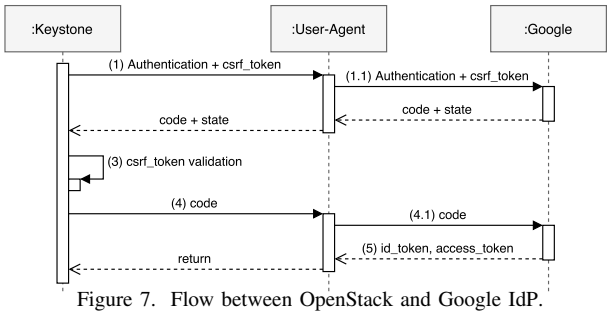

Figure 7. Flow between OpenStack and Google IdP.

For simplicity, UA is not shown in Figure 7. All messages exchanged between OpenStack and Google IdP go through UA according to the following steps: *(1)* OpenStack sends a token CSRF and the authentication request to Google's IdP; *(2)* The IdP responds the authentication request by passing the `state` parameter and the `code` parameter; *(3)* Token state is compared to the session's token and validated; *(4)* OpenStack uses the `code` and client information created in the Google Developers Console to request `id_token` and `access_token`; and *(5)* The IdP returns `id_token` and `access_token` to OpenStack.

It is possible to request additional user information via APIs, and use them for mapping. In this case, after OpenStack obtains the `access_token`, a message is sent to the IdP for accessing to the desired API and corresponding data. The `id_token` must be validated by the IdP to prevent replay attacks. Verification includes five steps that are done locally [25], so it is not possible to capture any data by eavesdropping the communication.

b) Facebook Connect: For OpenID Connect the scope of operations is defined within the API and under user authorization. For Facebook Connect, it is important to prevent applications from improperly accessing and using data. Applications that require access to specific APIs must be submitted for expert review, which verifies that the information requested is being used according to Facebook's terms of use. Hence, unlike OpenID Connect, Facebook Connect does not return basic user data by default – it is necessary to request this information via API calls. Also unlike Google/OpenID Connect, OpenStack communicates directly to Facebook. The authentication process is illustrated in Figure 8, which shows that not all messages are redirected through UA (similarly to OpenID Connect).

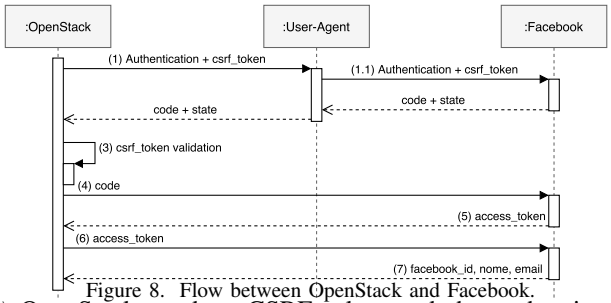

Figure 8. Flow between OpenStack and Facebook.

(1) OpenStack sends a CSRF token and the authentication request to Facebook via UA; *(2)* Facebook sends `code` and the `state` parameter; *(3)* The `state` is validated along with the session's CSRF token; *(4)* OpenStack uses `code` and the application credentials to get the user `access_token`; *(5)* Facebook returns the user's `access_token`; *(6)* OpenStack makes a new request, sending the `access_token` and requesting the user's name, e-mail and `facebook_id`; and *(7)* Facebook returns the requested information to OpenStack.

User information (*e.g.,* identifier) is not redirected through UA in Facebook Connect. Thus, Facebook provides an encrypted channel directly with the application, protecting the authentication packets.

4) Source Code Analysis: As an additional verification, we analyzed Keystone's source code, for OpenID Connect, and OFL's source code, for Facebook Connect.

a) OpenID Connect: Our analysis of the traffic between Keystone and Google's IdP revealed that no messages are exchanged to validate the `id_token` obtained. Therefore, this validation could only occur locally in Keystone. However, based on our analysis of the source code, Keystone does not perform any sort of validation of the token as specified by Google. More precisely, the existing functions only use the `token` or the `access_token` to obtain further user information via introspection. Hence, by intercepting the user's `id_token`, it is possible to generate an authentication request and consequently access the user's OpenStack projects. This vulnerability can be fixed in two ways: *(1)* Adoption of TLS by Keystone, thus protecting the data sent through the network; this requires the manual activation of TLS when configuring OpenStack, since this is not the default configuration. This approach does not fully solve past problems, though, since tokens that have been previously stolen can still be used by attackers. *(2)* Validate `id_token`, as specified in Google's documentation, which implies a modification in Keystone's source code in future releases.

Google specifies that the `id_token` is not always required if the service is able to ensure that the `id_token` comes from Google. This could be accomplished with some changes in the authentication flow, so the IdP's response takes the form of a `code` instead of an `id_token`. Precisely, in the current process, the `code` is received and used in exchange for the `id_token`, but this process goes through the UA, being susceptible to interception. With this modification, Keystone would query the IdP directly to get all required user information over a secure channel, instead of relying on the UA.

b) Facebook Connect: Unlike OpenID Connect, Facebook Connect obtains the user data directly, so there is no need to implement a separate process to validate `facebook_id`. However, the OFL tool stores some information from Facebook, such as `facebook_id` and the password generated during the Keystone account creation process. Storing `facebook_id` is necessary to identify whether the user's account already exists or needs to be created. The password, in turn, needs to be stored in an auxiliary table to allow users to authenticate to Keystone. However, storing the password without any protection may compromise users' information if the database is breached. Therefore, an authentication proxy implemented on Horizon is not sufficient for the correct authentication using Facebook Connect. A similar solution is adopted for user authentication in the OpenStack test cloud, TryStack (http://trystack.org/). Thus, the ideal option is to create a plugin for Keystone to manage the authentication without passwords, similar to the OpenID Connect plugin.

B. Using an External IdP

The adoption of an external IdP may be motivate by many reasons. For instance, securely implementing a local IdP may be a complex task, so it is reasonable to consider using existing solutions. The lower implementation cost may, however, result in an additional maintenance costs, as observed by [28] when using social networking authentication to allow users to access cloud resources. Nevertheless, an external IdP makes it easier to integrate the cloud with external services that rely on the same SSO mechanisms. In this study, Google and Facebook were selected because they currently are the most widely adopted IdPs [7], [25]. Using an IdP account for authentication implies following the stated privacy and security policies of that IdP. Hence, when using the Google account to authenticate to a third party service, the user agrees that Google can collect usage data such as [29]: *(1)* Information user transmits: Necessary information to create an account, such as first name, last name, and credit card (for purchases). *(2)* Information collected from service usage: This information includes application information used in account access, IP address, usage / advertisement information, location information, storage, cookies, applications, and more. When using a Facebook account to authenticate to a service, the user allows Facebook to access the following data: *(1)* Information of the user and his/her contacts; *(2)* Device information; *(3)* Shopping information inside Facebook; *(4)* Business partner information; and *(5)* Information from applications using Facebook services. It is possible to note on both privacy policies that Google and Facebook store information about the use of their services. However, using the authentication mechanism allows Google and Facebook to monitor user activities (e.g., partners, URI), which may impact the user's privacy [27]. Also, when using an external IdP, the user is susceptible to the vulnerabilities of the chosen IdP implementation. As noted in [24], the implementation of Google IdP has some features that can be exploited by attackers. Therefore, it is important to strictly follow the application's configuration guide, provided by the IdP, to avoid security and privacy issues.

C. Improper administrative access in OpenStack

Improper access to other projects may only occur with the use of OpenID Connect. In case of incorrectly configured mapping rules, the SSO solution may authorize user access to projects to which the user does not belong. On the other hand, OFL is implemented in a way that it does not use mapping rules. It operates in a way that is similar to users that authenticate via username and password. OpenID Connect authentication method should ensure that the user has access only to correct projects and that the user is not able to obtain administrative privileges in OpenStack. As mentioned in Sec. II, mapping rules are required to convert remote attributes into local ones and ensure user access to the designated project [10]. In order to ensure that only specific users have administrative access, it is necessary to specify criteria for the correct mapping. For instance, when Alice and Bob, with their respective e-mails `alice@example.com` and `bob@example.com`, are included in the administrators group, they acquire privileged access to OpenStack. Any attribute obtained through an OpenID claim can be used in the mapping filters, and regular expressions can also be used for mapping rules [10], [30]. To ensure users can only access their own projects, a remote attribute (e.g., e-mail) should be used to verify the existence of a project assigned to a given user or the creation of a new project. Therefore, user access is easily controlled via mapping rules.

D. Summary of the Analisys

Table I consolidates the results of the analysis hereby presented, listing the issues with each protocol and possible solutions for them. In this table, the risk level column is evaluated in a 1-5 scale, so it can be low, low-medium, medium, medium-high, or high. This indicates both the priority in which the listed solutions should be adopted and the severity of potential exploits when this is not done. Only the use of an external IdP was evaluated below medium, meaning that using OpenID Connect for SSO authentication in OpenStack is considered safe, but the selection of an IdP must be performed cautiously. Facebook Connect did not have as many vulnerabilities as OpenID Connect, only presenting problems with user privacy (the protocol implements, by default, several measures that minimize the compromise of user information). Also, OpenID Connect security integration with OpenStack is, overall, evaluated as medium-high. The main potential issue in this case is the use of an external IdP, which means that OpenStack's users become subject to an external privacy policy. Assuming that no correction is applied, Facebook Connect provides better security than OpenID connect in OpenStack. However, the proposed fixes are quite easy to implement, so they can be considered tied in terms of suitability.

IX. Considerations & Future Work

The analysis performed in this paper corroborates the perception that computational clouds inherit the security issues of technologies upon which they are built, but add new concerns when they are integrated and orchestrated in the cloud

Table I
SUMMARY OF SECURITY ANALYSIS OPENID AND FACEBOOK CONNECT ON OPENSTACK

Criterion	Vulnerability	Affected Protocols		Possible Solutions	Good Practices	Risk level
		OpenID Connect	Facebook Connect			
Data Encryption	*Man-in-the-middle*	Yes	No	Encrypted Channel Use `code` instead `id_token` as authentication response	`id_token` validation	Medium-High
Using external IdP	Privacy Violations	Yes	Yes	Choose a trustworthy IdP, compatible with internal policy rules		Medium-Low
User impersonation	CSRF	Yes	No	Use CSRF token	Verify CSRF token in all IdP sessions	Medium-High
Improper access to other projects	Recycling OpenID identifier	Yes	No	Use mapping rules which include user and group attributes, besides OpenID identifier	Generate unique OpenID identifiers (adopted by Google)	High

environment. Technologies such as TLS represent an essential building block that must be used to provide a basic level of security for services and their users, but they do not represent a definitive answer. Using external IdPs that outsource part of the security lifecycle, in particular the authentication and authorization components, also contributes to this complex scenario. In addition, privacy and compliance issues related to the (meta)data generated in the interactions with an external IdP (e.g., Google, Facebook) are also likely to be a concern. In the specific case of Google and Facebook, privacy issues may easily conflict with corporate security policies, so other external IdPs may be considered in real-world deployments.

As future work, we plan to build an authentication plugin for Keystone, instead of using a proxy in Horizon, and test the protocols in newer versions of OpenStack. Finally, we plan to expand the security analysis hereby presented, covering a broader range of aspects addressed in security standards and procedures [31].

Acknowledgments.This research was developed at LabP2D/UDESC.

REFERENCES

[1] P. Mell and T. Grance, "The NIST definition of cloud computing," 2011.

[2] Openstack, "Openstack – open source software for creating private and public clouds," www.openstack.org/, 2017.

[3] S. Bonner, C. Pulley, I. Kureshi, V. Holmes, J. Brennan, and Y. James, "Using OpenStack to improve student experience in an h.e. environment," in *SAI 2013*, 2013, pp. 888–893.

[4] M. Abid, I. Fihri, H. Mousannif, M. Bakhouya, C. El Amrani, M. Aissaoui, M. El Ouadghiri, A. Haqiq, A. Hayar, and M. Essaaidi, "MarUnivCloud: Towards a moroccan inter-university cloud," in *2nd World Conference on Complex Systems (WCCS)*, 2014, pp. 587–593, dOI:10.1109/ICoCS.2014.7060981.

[5] D. W. Chadwick, K. Siu, C. Lee, Y. Fouillat, and D. Germonville, "Adding federated identity management to OpenStack," *Journal of Grid Computing*, vol. 12, no. 1, 2013, dOI:10.1007/s10723-013-9283-2.

[6] I. Sette and C. Ferraz, "Integrating cloud platforms to identity federations," in *2014 SBRC*, 2014, pp. 310–318, dOI:10.1109/SBRC.2014.37.

[7] M. Urueña, A. Muñoz, and D. Larrabeiti, "Analysis of privacy vulnerabilities in single sign-on mechanisms for multimedia websites," *Multimedia Tools and Applications*, vol. 68, no. 1, pp. 159–176, jan 2014, dOI:10.1007/s11042-012-1155-4.

[8] G. C. Batista and C. C. Miers, "Security analysis of the OpenID Connect protocol integration with an OpenStack cloud using an external IdP," in *2016 XLII CLEI*, Oct 2016, pp. 1–12, dOI:10.1109/CLEI.2016.7833358.

[9] R. Khan, J. Ylitalo, and A. Ahmed, "OpenID authentication as a service in OpenStack," in *2011 7th IAS*, dec 2011, pp. 372–377, dOI:10.1109/ISIAS.2011.6122782.

[10] S. Martinelli, H. Nash, and B. Topol, *Identity, Authentication, and Access Management in OpenStack: Implementing and Deploying Keystone*, 1st ed. O'Reilly Media, Dec. 2015.

[11] M. Cordeiro Domenech, E. Comunello, and M. Silva Wangham, "Identity management in e-Health: A case study of web of things application using OpenID connect," in *2014 IEEE 16th Healthcom*, oct 2014, pp. 219–224, dOI:10.1109/HealthCom.2014.7001844.

[12] Z. Obrenović and B. d. Haak, "Integrating User Customization and Authentication: The Identity Crisis," *IEEE Security Privacy*, vol. 10, no. 5, pp. 82–85, Sep. 2012, dOI:10.1109/MSP.2012.119.

[13] L. Lynch, "Inside the Identity Management Game," *IEEE Internet Computing*, vol. 15, no. 5, pp. 78–82, sep 2011, dOI:10.1109/MIC.2011.119.

[14] J. Sendor, Y. Lehmann, G. Serme, and A. Santana de Oliveira, "Platform-level Support for Authorization in Cloud Services with OAuth 2," in *2014 IEEE IC2E*, mar 2014, pp. 458–465, dOI:10.1109/IC2E.2014.60.

[15] M. N. Ko, G. P. Cheek, M. Shehab, and R. Sandhu, "Social-networks connect services," *Computer*, vol. 43, no. 8, pp. 37–43, 2010.

[16] M. Miculan and C. Urban, "Formal analysis of Facebook Connect single sign-on authentication protocol," in *SOFSEM*, vol. 11, 2011, pp. 22–28.

[17] S. Egelman, "My Profile is My Password, Verify Me!: The Privacy/Convenience Tradeoff of Facebook Connect," in *Proc. of the SIGCHI*. ACM, 2013, pp. 2369–2378, dOI:10.1145/2470654.2481328.

[18] A. Sharma, "Social Login and Sharing Statistics for Q1 2015," apr 2015. [Online]. Available: http://blog.loginradius.com/2015/04/social-statistics-for-q1-2015/

[19] D. Nolan and D. T. Lang, "Authentication for Web Services via OAuth," in *XML and Web Technologies for Data Sciences with R*, ser. Use R! Springer New York, 2014, dOI: 10.1007/978-1-4614-7900-0_13.

[20] D. Hardt, "The OAuth 2.0 Authorization Framework," 2012.

[21] F. Yang and S. Manoharan, "A security analysis of the OAuth protocol," in *2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, aug 2013, pp. 271–276, dOI:10.1109/PACRIM.2013.6625487.

[22] D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage," 2012.

[23] B. v. Delft and M. Oostdijk, "A Security Analysis of OpenID," in *Policies and Research in Identity Management*, ser. IFIP Advances in Information and Communication Technology. Springer Berlin Heidelberg, nov 2010, no. 343, dOI: 10.1007/978-3-642-17303-5_6.

[24] W. Li and C. J. Mitchell, "Analysing the security of google's implementation of openid connect," *arXiv preprint arXiv:1508.01707*, 2015.

[25] Google, "OpenID Connect," 2017.

[26] OpenID, "OpenID Connect FAQ and Q&As," 2017.

[27] A. Bhargav-Spantzel, J. Camenisch, T. Gross, and D. Sommer, "User Centricity: A Taxonomy and Open Issues," in *Proc. of the 2nd ACM Workshop on Digital Identity Management*, ser. DIM '06. New York,NY, USA: ACM, 2006, pp. 1–10, dOI:10.1145/1179529.1179531.

[28] D. W. Chadwick, G. L. Inman, K. W. Siu, and M. S. Ferdous, "Leveraging social networks to gain access to organisational resources," in *Proc. of the 7th ACM workshop on Digital identity management*. ACM, 2011, pp. 43–52.

[29] Google, "Privacy police – Privacity & Terms – Google," 2017.

[30] Openstack.org, "Mapping Combinations for Federation," 2017.

[31] CSA, "Security Guidance for Critical Areas of Focus in Cloud Computing V4.0," Cloud Security Alliance, Tech. Rep., 2017.