

Tequila – A distributed Web authentication and access control tool

Claude Lecommandeur

Ecole Polytechnique Fédérale de Lausanne, Switzerland

claudio.lecommandeur@epfl.ch

1. Abstract

EPFL is one of the two federal institutes of technology in Switzerland, located in Lausanne, with more than 6000 students and 3500 staff members. For many years we have been using a centralized Web authentication tool called Gaspar. Two years ago, the need arose for a more sophisticated tool that can do cross institution authentication and access control.

Tequila was designed to meet this goal. The main features of Tequila are:

- Authentication AND access control.
- Automatic multi-institutions cross authentication.
- Protection of dynamic (e.g. scripts) and static (e.g. files) content.
- Single sign-on.
- Focus on data protection and users privacy.
- Easy customization.
- Easy administration.
- Portability.
- GPL license.

2. Local authentication

To start with an example: a user wants to authenticate to an application inside his home organization. Tequila does Web based authentication messaging; there are not that many ways to do this. When the client application (service provider) wants to authenticate a user, it asks the service of the local Tequila server by redirecting the user's browser to this server, the user then authenticates himself, the server checks the authentication, checks if the user meets the constraints imposed by the application and, in case of success redirects the user to the calling application.

Actually, Tequila has three authentication operation processing modes. In this paper I'll describe in detail only the main mode since it is by far the most secure and the one that is the most simple for client applications.

There are seven major steps in the process:

1. Upon a user request, the client application (after verifying this user has not already an opened session), calls an URL on the server with all the characteristics of the authentication needed :
 - What kind of information about the user is needed (name, status, etc...)
 - What kind of constraints must the user verifies.
 - What is the service name.
 - Where to redirect the user after successful authentication.
 - Etc...
2. The server stores the request, and sends back a unique key to the calling application.
3. The application then redirects the user to the Tequila server with the unique key appended in the server's URL (and optionally deposits a cookie in the user's browser containing the unique key for later use).
4. The Tequila server presents the user with a login screen. Upon successful authentication and constraint verification, the server creates a short life session with all the needed information. This session is tagged with the unique key of the user.
5. The Tequila server redirects the user agent to the calling application, using the access URL inside the request, appending the unique key.
6. When the application sees the user coming back with the key, it opens a new URL to the Tequila server that asks it if the key is valid, and what are the attributes of this user. The server responds with the values of all these attributes.
7. The application creates a new session for this user.

In this simple case, the login screen looks like this:

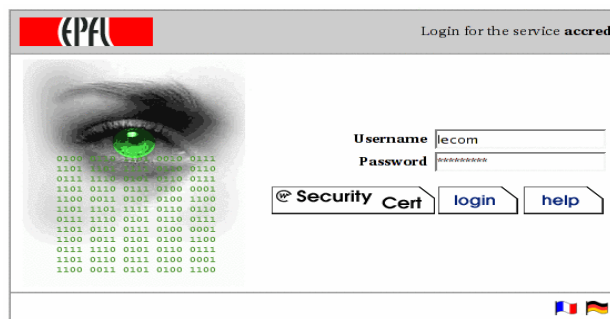


Illustration 1 : Local only authentication

3. Distributed authentication

In the distributed authentication case, the user wants to authenticate to an application that is not inside his home organization, and he may actually be completely unknown to the identity managing service of the application's organization.

But the application's organization belongs to a set of organizations that have a level of trust among them, and have configured their Tequila servers to trust each other as well. The application does not need to be aware of this. It acts as usual, accessing its own Tequila Server.

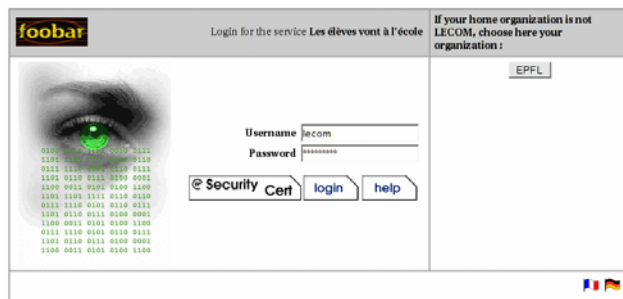


Illustration 2 : General authentication

In this case the login screen presented to the user is not the same. You can see the difference on the right. The user can choose to authenticate to another organization. Here, there is only one choice, but there could be many.

What happens when the user authenticates himself by clicking on his organization?

The application's Tequila server receives the request, sees it is not for itself, checks the asked organization to see if it knows it and trusts it. If yes, it acts as a client and asks for user authentication to the Tequila server of this organization.

The user is redirected to his home organization Tequila server, authenticates, and is redirected to the local Tequila server, that itself redirects the user to the local application.

With this scheme, the user has disclosed his user name and password only to his home organization server.

4. Attributes

Tequila manages users and attributes. Attributes are just name/values pairs. Names and values are strings, attributes can be multivalued. Clients can request the value of attributes for the authenticating user, or they can impose boolean criteria that these attributes must meet. This is all the purpose of server side access control.

5. Access control

Tequila does more than authentication. It can do access control. The client application can ask for the server to perform tests on user attributes and immediately reject users that do not fit these tests.

An example, suppose an application wants to give access to its services only to people belonging to a certain group, say 'Mygroup'. This is very simple, the application has just to tell the server the constraint 'require=group=Mygroup'.

Constraints can be composed to obtain more complex behavior, such as: 'require=unit=Myunit&group=Mygroup'. That means you want authenticate only users that belong to unit Myunit and group MyGroup. You can use any Boolean parenthesized expression to specify constraints.

The advantage is double, it relieves the client from doing the check itself, and more importantly, the client doesn't even need to know who is in the group or the unit.

Right now, the server always delivers the user's username to the client, but is very easy to change this behavior and have completely anonymous access control, the server only discloses the information the connected user verifies some criteria, without disclosing the identity of the user.

6. Dynamic content protection

The Tequila services can be accessed directly and easily from scripts. Scripts can of course implement directly the raw URL redirection necessary to use Tequila server services. But this is tedious and error prone. The standard Tequila distribution includes modules for Perl, PHP and Java.

These modules perform all the URLs plumbing and redirections, manage local sessions. They are written to do the thing you think they will do in their simplest form, but also leave space for more advance usage.

The simplest usage can have the form (all examples are in Perl):

```
use Tequila::Client;
my $tequila = new Tequila::Client ();
$tequila->authenticate ();
my $user = $tequila->{user};
```

The first line tells the Perl compiler we are using the Perl Tequila client module. In the second line, we are creating a new Tequila::Client object. The third line asks the user for authentication, doing all the session management, URL manipulation and redirecting stuff. The first time *authenticate* is called, it never returns because the user is redirected to the server. The second call returns (in case of successful authentication and authorization) and the Tequila::Client object is filled with default user attributes.

It couldn't be simpler.

Another example with attributes and constraints:

```
use Tequila::Client;
my $tequila = new Tequila::Client ();
$tequila->require ("role=admin&group=aasl");
$tequila->request ("name", "firstname");
$tequila->authenticate ();
my $user = $tequila->{user};
my $name = $tequila->{attrs}->{name};
my $firstname = $tequila->{attrs}->{firstname};
```

Not that difficult either. The call to *require* means that to authenticate correctly, the user must verify the assertion "role=admin&group=aasl", you can imagine that with a suitable setting of your identity management system, this

means that user has role 'admin' and belongs to group 'aasl'. The *request* call means that we want to know the name and firstname of the user. After the successful return from *authenticate* we can be sure that the required criteria is met, and the name and firstname fields are filled with the correct values.

7. Static content protection

Protecting scripts is one thing, protecting static documents (html files, directory) is another thing.

Tequila provides an Apache module cleverly named `mod_tequila`. This module encapsulates part of the interaction with a Tequila server, in an easy to understand access control tool. Let's first examine an example of a simple *.htaccess* file:

```
TequilaAllowIf group=aasl&org=EPFL
```

It is as simple as that. You just tell the module the criteria used to give access to the directory and it is done, the whole directory and all subdirectories are protected, only members of 'aasl' group and members of EPFL organization will be given access to the directory where the *.htaccess* is.

Of course, you can accomplish more complex things. Setting several constraints, adjust log file, log level, redirect to others directories, etc. But this is another story told in the documentation. Just to tease you :

```
LoadModule tequila module
/usr/lib/httpd/modules/mod_tequila.so

<IfModule mod_tequila.c>
  TequilaLogLevel      9
  TequilaLog           /etc/httpd/logs/tequila.log
  TequilaServer        tequila.epfl.ch
  TequilaSessionDir    /var/www/Tequila/Sessions
  TequilaSessionMax    3600

  <Location /restricted/>
    TequilaRewrite      /var/www/html/
    TequilaAllowIf      unit=DIT-DEV
    TequilaAllowIf      username=lecom@slpc1.epfl.ch
    TequilaAllows       categorie=epfl-guests
  </Location>
</IfModule>
```

8. Single sign-on

There are many misunderstandings of what single sign-on is, it is often confused with single login. Single login is when all applications share a common set of users/passwords base. In this case every application must authenticate itself its users and hence be able to verify that passwords are correct. This is often done by using a LDAP back-end server that has the charge of checking the correctness of user/password pairs.

The advantage is obvious; people have to memorize only one username and one password. But drawbacks are numerous, each application must have a way to check passwords, users must disclose their passwords to every application they log into, hence trusting it, and users have to retype username and password several times a day.

Single sign-on is different, authentication is performed by one server only, and applications delegate this task to this server. This alleviates all the drawbacks of the single login usage, users have only to trust this central server, this server can

cleverly recognize users so that they don't have to re-type their username/password.

Of course Tequila is of the latter breed. The first time a user authenticates, the server deposits a specially crafted session cookie into the user's browser. On subsequent authentications, this cookie will be used to safely recognize the user and not ask for password again and again. Single logout is possible by calling a specific URL on the server itself that destroys the cookie.

9. Users privacy protection

Tequila servers can manage potentially sensitive data about users, data that are not supposed to be handed out to any application that asks for it.

For this purpose, Tequila has the notion of 'sensitive' attributes. Sensitive attributes are defined in the server configuration. When a Tequila client asks for the value of such attribute, the login screen shows more information to the user. For example, if I define the attribute 'firstname' as being sensitive, the following things will happen:



Illustration 3 : Sensitive attributes warning.

10. Connectors

Tequila servers don't hold any information about users all by themselves. They must rely on external identity data servers. The communication between Tequila servers and the local identity management system is done via 'connectors'. There are two kinds of connectors, authentication connectors and data connectors.

10.1 Authentication connector

The authentication connector (only one by server) is in charge of authenticating users (surprising!). It implements only three methods :

- `configure (config_file)` : Connector initialization.
- `auth (username, password)` : Does what you think and returns true or false.
- `validuser (username)` : Is this username valid?

Some more methods can be implemented in order to take advantage of the automated configuration tool (Margarita) :

- `init ()`
- `setattributes ()`
- `adminpage ()`
- `changeconfig`
- `writeconfig`
- `etc...`

10.2 Data connectors

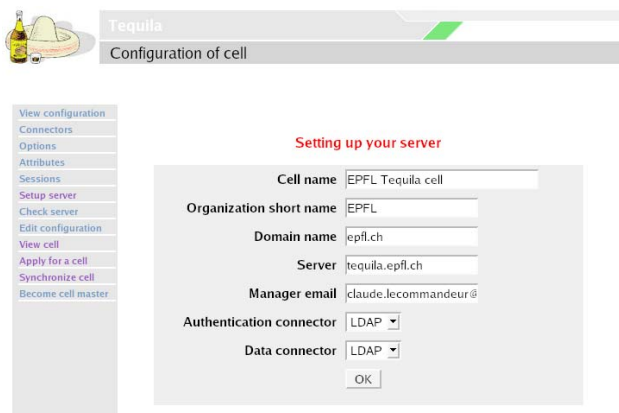
There may be any number of data connectors. They are responsible for providing user attributes values. They must implement the following methods:

- `configure ()` : Connector initialization.
- `supports ()` : Returns the list of attributes supported by this connector.
- `getattrs (username, @attrslst)` : returns a hash with the values of all attributes in *attrslst* filled.

Other methods are also necessary if you want the connector to be configurable with Margarita.

11. Administration

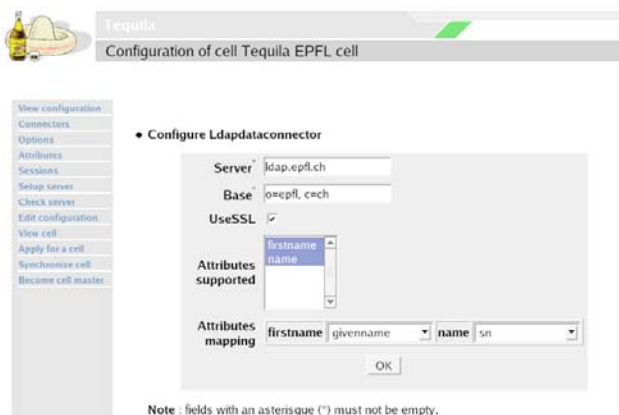
A Web tool called Margarita is provided to easily configure the Tequila server. Margarita can manage the server itself and all the connectors that export the configuration interface (see previous paragraph). Example screens :



The screenshot shows the Margarita web interface. At the top, there's a header with a Tequila logo and the text 'Configuration of cell'. Below this is a sidebar menu with options like 'View configuration', 'Connectors', 'Options', 'Attributes', 'Sessions', 'Setup server', 'Check server', 'Edit configuration', 'View cell', 'Apply for a cell', 'Synchronize cell', and 'Become cell master'. The main content area is titled 'Setting up your server' and contains a form with the following fields:

- Cell name: EPFL Tequila cell
- Organization short name: EPFL
- Domain name: epfl.ch
- Server: tequila.epfl.ch
- Manager email: claudel.lecomandeur@
- Authentication connector: LDAP (dropdown)
- Data connector: LDAP (dropdown)
- OK button

Illustration 4 : Margarita main screen



This screenshot shows the 'Configure Ldapdataconnector' form within the Margarita interface. The sidebar menu is the same as in the previous screenshot. The main content area is titled 'Configure Ldapdataconnector' and contains the following fields:

- Server: ldap.epfl.ch
- Base: o=epfl, c=ch
- UseSSL: checked
- Attributes supported: A dropdown menu showing 'firstname' and 'name'.
- Attributes mapping: A table with two columns. The first column has 'firstname' and 'givenname' in the first row, and 'name' and 'sn' in the second row.
- OK button

Below the form, there is a note: 'Note : fields with an asterisk (*) must not be empty.'

Illustration 54 : Margarita, configuring LDAP connector

12. Implementation

The Tequila server is fully written in Perl, so are the default connectors. Other connectors must be written in Perl too, but a thin Perl interface to the actual code can be used. The Apache modules are written in C.

13. Conclusion

Tequila can play a major role either in single and multiple organizations trust scheme. It is not a finished tool, there is plenty of room for improvement. Objects manipulated by Tequila are conceptually information and assertions about users, it is very close to what SAML (Security Assertion Markup Language) is all about, I will probably soon add SAML support into Tequila, enabling more generic SAML clients to dialog with Tequila servers.

14. License

Tequila is GPL, you are encouraged to download, use, and improve it. I will offer support, but Tequila is a work in progress, and any help is welcome.

15. References

- [1] Tequila home page : <http://tequila.epfl.ch/>
- [2] Source forge : <http://sourceforge.net/projects/tequila-auth/>
- [3] Download : <http://slpc1.epfl.ch/public/software/tequila/>
- [4] EPFL : <http://www.epfl.ch/>