Notes on account auto-provisioning and federated remote access

Account auto-provisioning is not a trivial matter, not just from the technical point of view, but also for:

- Every institution has implemented its own techniques, policies and procedures to deal with account management.
- The concept of ‘account’ is not so clear, and less in a federated environment.

For example, what services needs or has permission to access and external, federated, user? Do our directory system design or our access control mechanisms provide enough granularity to cope with these cases?

Our initial interest in account auto-provisioning arose in the context of the federation of the Andalusian Universities (CONFIA), whose first goal is to provide the Andalusian Virtual Campus with the benefits of identity federation technologies: sign on in the home institution when accessing remote VLEs, automated interchange of curricula information between institutions, etc. Besides access to the VLE of a remote institution where the student is to get a course, he probably needs access to its computing environment, where is installed some software that supports the course.

In our University, we have a home made web-based application for account management: it creates, deletes, changes the password, etc. for the two computing environments we provide to our academic activities: Linux and Windows. For that, it directly manipulates our LDAP and Active Directory servers.

In this context it is almost trivial for us to create a local account for external students that access our federated VLE: just place a link on it called ‘Create local account’ or something similar that triggers an internal connection to a special URL of the account management application together with the necessary parameters to create the account and redirect the user to the page where he can select its password. The user can only access this link if he has successfully authenticated to ‘the federation’. At this moment, the account management app itself is not federated, so the access to this link and the ‘secret’ account creation URL are protected by not very secure mechanisms, but this will change as our federation comes into production.

Our computing rooms environment is based on a heavily customized version of Thinstation, converted from thin client to a fully functional GNU/Linux system and Citrix for access to our Windows servers farm. The PCs in these rooms do not use a local hard drive (except for caching). For remote access from home, we also have some servers that run the same GNU/Linux environment as the PCs, and accept remote access via NX. So, both our students and remote, federated ones, have local and remote access to exactly the same environment, once the have an account, of course.

Our initial proposal on naming is that the new account name is the same as their home username with ‘-home-domain’ appended. So the user from uma.es whose login there is ‘victoriano’ will be assigned the account ‘victoriano-uma-es’. This guarantees unicity (as long as you don't use the '-' character in local accounts) and makes it easy to keep track of these accounts.

But… one of the benefits of identity federations is to eliminate the burden of having several passwords ¿isn't it? So the next step is to try to provide access to desktop environments like those of our computer rooms without the need of remembering a password (anyway, maybe we have no choice if the external user has to get access to local resources and services that are not federated).

Our remote access method for the GNU/Linux environment is NX, and it allows you to declare an username and a password in the connection configuration file, so that the authentication dialog is obviated. In any SP (such as the VLE) we can provide a link with a text such as 'GNU/Linux desktop' that when clicked, invokes a server script that:

1. Checks in the LDAP server if account named as stated before already exists.
2. If it doesn´t exist, it optionally can create it (the script is protected as an SP). This is an alternative to the previous creation of the account.
3. Generates a temporal password and stores it in the userPassword attribute on the LDAP server
4. Generates a NX connection file (.nxs) that is sent to the browser and includes the username and password. This will make the browser launch the NX client. Another option is to send a page that includes the NX applet and references a connection file on an URL that makes it be generated that way.

When the NX client starts up, the NX server accepts the username and password, as it binds successfully to the LDAP server thanks to the step 3.

The temporary password should allow only this one login. There are several ways to achieve this, some of them are cron jobs that resets passwords for these accounts, the use of ppolicy on the LDAP server to make the password usable just once, etc.

For the sake of security, we can configure the LDAP server to allow this script to change the password only for federated users with rules such as the following:

```
access to attrs=userPassword \
filter=(&(objectclass=schacEmployeeInfo)(schacPersonalPosition=confia))
         by dn="cn=fedadmin,dc=uco,dc=es" write
         …
```

where 'cn=fedadmin,dc=uco,dc=es' is the DN that the script uses for modifying the userPassword attribute, and only for federated accounts, those with the attribute schacPersonalPosition indicates a federated user.

We have not yet explored remote access to the Windows servers farm, but surely it is not so affordable ☹

Of course, there are a lot of details to consider, differences in local policies, etc. But the model allows for many alternatives.