Authentication portal in FreeIPA
bachelor thesis concept

Milan Kubík
xkubik17@stud.fit.vutbr.cz
mkubik@redhat.com

April 5, 2013

This paper contains an analysis of FreeIPA, OpenID and SAML technologies as well as some ideas and draft proposal for implementation of *Authentication portal in FreeIPA* project/bachelor thesis. The aim of this document is to summarize my progres through the technologies as well as present several proposals for the implementation in terms of my understanding of the project goals.

FreeIPA analysis

The most important part of FreeIPA system regarding authentication portal is 389 directory server, storing the identities of respective users as well as all necessary structures and entries required for the authentication portal functionality.

To be considered is possible use of internal certification authority for the needs of web interface of the system. Integral part of the system is kerberos authentication using internal realm for the user identities.

As for user interface part directly in portal in FreeIPA server, this will be just an interface for managing the entries, managing approved trusts for stored third party applications etc.

OpenID and SAML

The use case of OpenID is pretty straightforward as the standard covers one specific use case. There is a python library implementing OpenID protocol. SAML on the other hand is more of a toolkit for defining assertion protocols with more expressive power than OpenID. One of the "standardized" profiles useful for this usecase is Web single sign-on profile.

Pluggable system design

Currently the main idea is to create some intermediate layer between IPA server and IdP which basicly means accessing data stored in directory server and use it in application logic separated from IPA server itself.

Two approaches

- As a part of FreeIPA – has its downfalls. Most importantly would require direct access to the IPA server from the internet.
- Separate service – web service bound to IPA infrastructure, mostly utilizing already present functionality.

1 Component design

Component design proposal by those rules. The IdP application is intended to be a wsgi application basicly acting as a client of FreeIPA server with most of the user interface implemented directly in IPA.

1.1 Definitions

- *IdP* – identity provider (OpenID server, SAML compliant server)
- *End user* – user authenticating via IdP to 3rd party service (RP)
- *RP* – Relying Party as defined in OpenID specification[1]
- *SSO* – Single Sign-on

1.2 Target audience

- *by organizational size* – Small to Enterprise customers.
- *by role* – end users

1.3 Identity management

Identity of an user shall be managed by FreeIPA server/infrastructure itself. IdP related data should be stored in Directory server[2]. Either with user entry itself as an attribute added by introducing new auxiliary object class to the entry, or by a new subtree or branch in mapping tree bound to the plugin/subsystem.

1.4 Access control

In the best case scenario, logging in to the IdP[3] should be enabled only from internal IP address range. However at the moment I'm not sure if this is entirely possible with regard that OpenID requests can come as direct messages[4] from RP and as such being most probably from the Internet rather that from the internal network[5].

The access of IdP to data stored in IPA/DS should be done via system account with sufficient access rights in the way ipa client works. Either by using already present ipa client functionality or as a "plugin" (both server and client side) accessign IdP related data. In this way the actions of IdP (and the system user) will be subject to IPA policies regarding access control.

Access to web UI shall be done in the already present fashion. E. g. by group membership. UI[6] for end user and member of admin group will be different.

1.5 Configuration management

There should be two approaches to configuration of IdP. One derives from choosen technology. In case of OpenID this is the management of trusts.
Second type of management is "admin" approach managing the component itself. For instance enabling the entire subsystem, enabling and disabling the functinality of the subsystem for specific user account et cetera. Also setting the Identifier is vital part of the configuration. This choice should be made once and not changed after being used by an user to authenticate to *RP* as well as not changing related DNS configuration not to break the discovery procedure. Both of those types of configuration management should be accessible from IPA server web UI (and perhaps in some fasion from command line utility). The only exception from this being the process of saving the trust for a new RP. This would also require the IdP to be able to write the change on behalf of the End user into the backend database (directory server).
User's view[7] should also serve as entry point to the SSO functionality, initiating the authentication against RP.

[1] `http://openid.net/specs/openid-authentication-2_0.html#terminology`
[2] E. g. trusted realms in case of OpenID
[3] Assuming Identity provider runs separately from IPA server itself.
[4] As a matter of fact direct messages are used only while establishing association or verifying assertions.
[5] To some extent it could be assumed that using Kerberos in internal network is sufficiently safe, however this is an uninformed guess of mine at the moment.
[6] UI itself complying web ui guidelines
[7] A tab in FreeIPA server.

1.6 Policy

The system should put emphasis on the policies regulating the authentication against the IdP subsystem as this is potentially dangerous point to internal infrastructure that should not be disclosed to outside world. This proposal considers using Kerberos subsystem as the choice for authentication of users deployed and managed on the side of http server (apache).