FreeIPA and SSSD

Free software identity management

Red Hat Developers' Conference

Jakub Hrozek
Martin Nagy

September 14, 2009

Section 1

Introduction

Identity Management Problem

- Problem:
  - How do we manage policies and user identities across multiple computers in a large network?
- Solution:
  - Put user identity information and the policies that govern users into a central location.
- Technologies:
  - Kerberos (might require DNS and NTP)
  - LDAP

Kerberos

- Authentication protocol.
- Single sign-on, user acquires a *ticket* from *Key Distribution Center* and uses it for authentication.
- Machines are grouped into kerberos *domains*.
- Identity is represented by a kerberos *principal*.
  Example: `admin@EXAMPLE.COM`
- Clients and KDC have to have synchronized time.
- LDAP server can serve as a database back-end.
- Clients can discover Kerberos servers via DNS.

LDAP

- Lightweight Directory Access Protocol
- Protocol for querying and manipulation of directories.
- Directory is a set of objects with attributes organized in a tree-like structure (DIT).
- Good access control granularity.
- Optimized for read operations.
- Supports multi-master replication.
- Binding to an LDAP server can serve as authentication as well.

Problems

- Administrator needs to use and understand more than one non-trivial technology.
- Users that are not tech savvy can't manage their own information themselves.
- Bad level of abstraction. Administrator doesn't want to add an entry into the database, but e.g. create a new user.

Section 2
FreeIPA

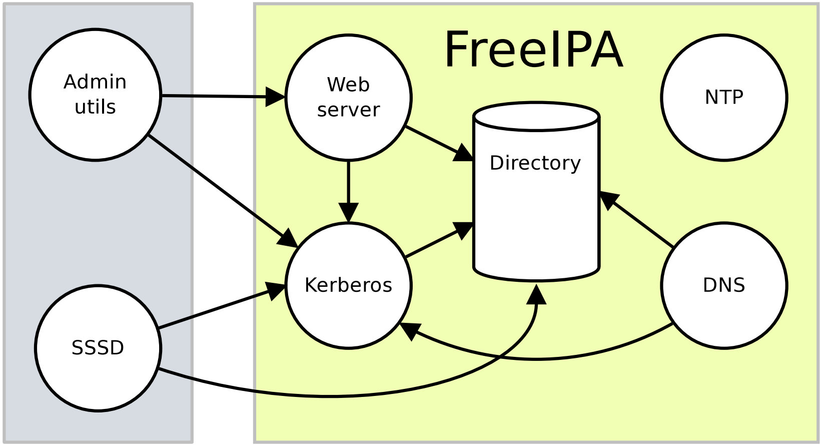
FreeIPA

- Ease off setting up the necessary infrastructure.
- Make the management of identities easier.
- Provide single sign-on.
- Centralizes identities in a database with multi-master replication.
- Hide LDAP and Kerberos implementation details.

FreeIPA Components

- 389 Directory Server
    - Formerly called "Fedora Directory Server"
    - Supports multi-master replication.
- MIT Kerberos
- Apache HTTP server
- BIND DNS server
- NTP server

FreeIPA Architecture



redhat.

Using FreeIPA (v1)

Installation

```
# ipa-server-install
```

Administration

```
# kinit admin
Password for admin@EXAMPLE.COM:
# ipa-useradd -f John -l Doe jdoe
```

Replica installation

```
server# ipa-replica-prepare replica.example.com
Packaging replica information into
/var/lib/ipa/replica-info-replica.example.com
server# scp replica-info-replica.example.com.gpg \
        root@replica:~/
replica# ipa-replica-install \
         replica-info-replica.example.com.gpg
```

Using FreeIPA (v1)

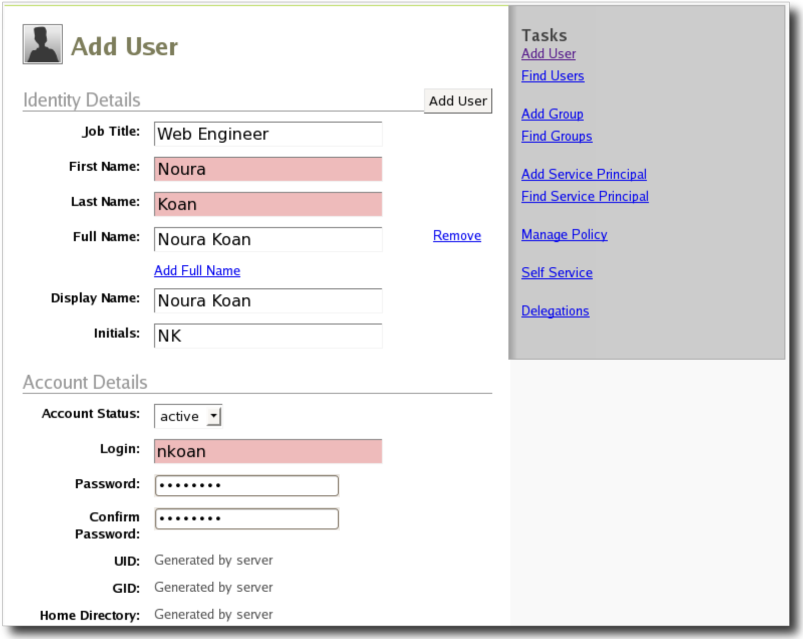
User

- Installing the IPA client:

```
# ipa-client-install
```

- Changing the user's shell:

```
$ kinit jdoe
Password for jdoe@EXAMPLE.COM:
$ ipa-usermod -s /bin/tcsh
```



Add User

Identity Details

Add User

Job Title:	Web Engineer
First Name:	Noura
Last Name:	Koan
Full Name:	Noura Koan Remove
	Add Full Name
Display Name:	Noura Koan
Initials:	NK

Account Details

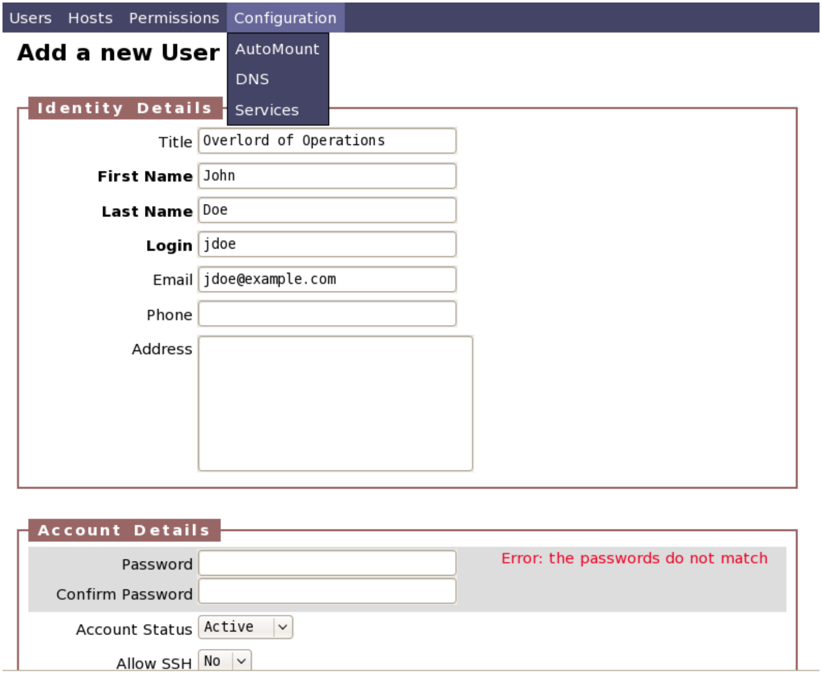
Account Status:	active
Login:	nkoan
Password:	••••••••
Confirm Password:	••••••••
UID:	Generated by server
GID:	Generated by server
Home Directory:	Generated by server

Where are we now?

- FreeIPA version 1
  - Only user identity.
  - Getting the 389 DS to cooperate with MIT Kerberos.
  - Command line utilities and a Web UI.
- FreeIPA version 2
  - Currently being actively developed.
  - Easily extensible plug-in framework.
  - Machine identity (DNS integration).
  - Host based access control.
  - Certificate Authority integration.

Where are we now?

- FreeIPA version 3
  - Design underway.
  - Cooperation with Active Directory.



Add a new User

Identity Details

Title	Overlord of Operations
First Name	John
Last Name	Doe
Login	jdoe
Email	jdoe@example.com
Phone	
Address	

Account Details

Password	
Confirm Password	
Account Status	Active
Allow SSH	No

Error: the passwords do not match

Section 3

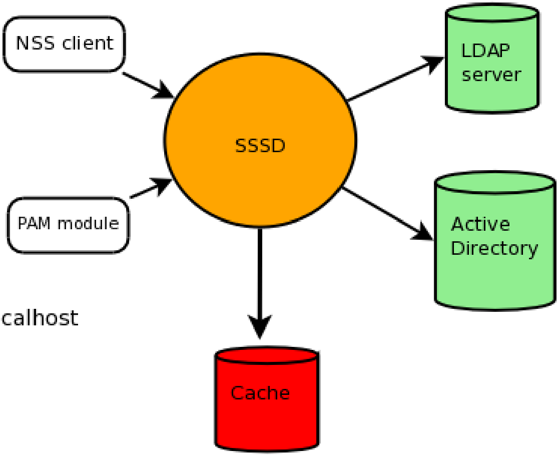
SSSD

SSSD - more than a FreeIPA client

- `http://fedorahosted.org/sssd`
- system daemon
- provides access to identity and authentication remote resources
- better database to store local users as well as extended user data.
- interfaces with the system via NSS module and a PAM module
- under development since September 2008

SSSD architecture illustrated

getent passwd foo



ssh foo@localhost

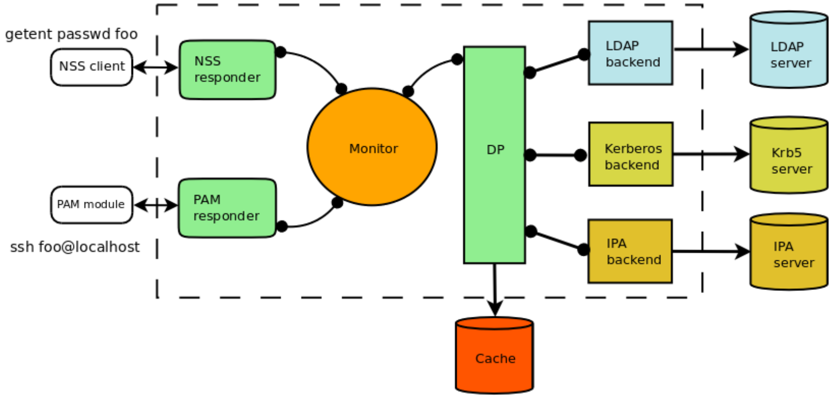
Current development state

- current released version is 0.5.0
  - LDAP, Kerberos backend
  - caching of identity and credentials
  - local database with management tools
- development of 0.6.0 ongoing (F12, RHEL6)
  - will provide IPA backend with HBAC
  - aliases for complex domain settings (`domain-type=`)
  - server failover
  - python bindings for managing local users
- Pre-built binaries available for Fedora, Ubuntu, builds on Suse

SSSD architecture

- all SSSD processes are single-threaded and use an event loop for pseudo-concurrence
- monitor - a process that watches over other services, starts or restarts them as needed
- specialized SSSD services
    - Data provider populates cache from backends, reaches out to backend if necessary
    - NSS responder answers NSS requests from the `nss_sss` module
    - PAM responder manages a PAM conversation through the `pam_sss` PAM module
- the specialized services communicate with monitor via DBus

SSSD architecture illustrated



The local database

- indented to complement or replace /etc/passwd, /etc/shadow
- the format of the database is LDAP-like
  - sopihisticated search operations
  - extensible - user avatar, locale, preferred DE
  - LDB, `http://ldb.samba.org`
- SSSD comes with a set of tools to manage the local domain
  - `sss_useradd`,`sss_userdel`,...
- groups can be nested

Remote databases

- LDAP, Kerberos, IPA, AD, . . .
- provides caching
  - no need to contact remote servers for every request
- offline authentication
  - offline authentication and identity for laptop users
- can provide backend-specific services
  - Host Based Access Control for FreeIPA
  - auto-discovery of servers
  - location based discovery

Example: configuring an LDAP/Krb client

Domains configuration example

```
[domains]
domains = ldap.example.com,krb.example.com

[domains/ldap.example.com]
domain-type = ldap
server = ldap.example.com
ldap-use-tls = ssl
ldap-usersearchbase = ou=users,dc=example,dc=com

[domains/krb.example.com]
auth-module = krb5
krb5KDCIP = 192.168.1.1
krb5REALM = EXAMPLE.COM
```

Example 2: configuring an IPA/AD client

Domains configuration example

```
domains = local,ipa.example.com,ad.example.com

[domains/local]
domain-type=local

[domains/ad.example.com]
domain-type=ad
server=ad.example.com

[domains/ipa.example.com]
domain-type=ipa
server=ipa.example.com ipa2.example.com
```

Get involved

- home page - `www.freeipa.org`
  - read docs, get tarballs, learn more about FreeIPA
- `http://fedorahosted.org/sssd`
  - HOWTOs, bugtracker
  - manpages, annotated sssd.conf
- talk to us
  - IRC - FreeNode, `#freeipa`
  - mailing lists freeipa-devel, sssd-devel
- hack on FreeIPA
  - `http://freeipa.org/page/Contribute`

That's it

- Questions?

talloc

- hierarchical, reference counted memory pool system with destructors

Code example

```
struct foo *X = talloc(mem_ctx, struct foo);
X->name = talloc_strdup(X, "foo");
```

- hierarchical, reference counted memory pool system with destructors
- `talloc_free(X->name) != talloc_free(X) != talloc_free(mem_ctx)`
- n-ary tree where you can free any part of the tree with `talloc_free`
- provides destructors
- provides means to ”steal” pointers from one context to another

The end.

Thanks for listening.