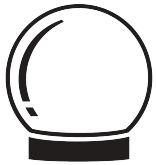
Using OS-level identity, authentication, and access control for Web applications

Jan Pazdziora
Principal Software Engineer
Identity Management Engineering, Red Hat
jpazdziora@redhat.com



DEVCONF.cz
6th February 2015

Identity Management

- Users; user groups. Hosts; host groups; services; ...
- Policies, host-based access control (HBAC) rules.
- FreeIPA (IPA) server holds and manages the identities (what in the old days was in `/etc/passwd`, `/etc/group`, ...) and policy definitions.
  - Multiple protocols and technologies under common interfaces (WebUI, CLI, helper tools).
  - Replicas for fault-tolerance and performance.
- sssd is a client-side component for authentication, identity operations, rule enforcement.
  - Caching (speed, offline use), failover, multiple domains.
- In latest versions, cross-realm trust with Active Directory (AD), and seamless handling of AD group memberships and user attributes.

Setting up FreeIPA server

```
[root@ipa ~]# ipa-server-install [ some helpful parameters ]
[...]
This program will set up the FreeIPA Server.

This includes:
  * Configure a stand-alone CA (dogtag) for certificate management
  * Configure the Network Time Daemon (ntpd)
  * Create and configure an instance of Directory Server
  * Create and configure a Kerberos Key Distribution Center (KDC)
  * Configure Apache (httpd)
  * Configure DNS (bind)
[... a minute or so later ...]
[root@ipa ~]# kinit admin
Password for admin@EXAMPLE.COM:
[root@ipa ~]# ipa host-find ipa
--------------
1 host matched
--------------
  Host name: ipa.example.com
  Principal name: host/ipa.example.com@EXAMPLE.COM
  Password: False
  Keytab: True
```

Command line interface

```
[admin@ipa ~]$ ipa user-add --random --first Thomas --last Thomasson tom
------------------
Added user "tom"
------------------
  User login: tom
  First name: Thomas
  Last name: Thomasson
  Full name: Thomas Thomasson
  Display name: Thomas Thomasson
  Initials: TT
  Home directory: /home/tom
  GECOS: Thomas Thomasson
  Login shell: /bin/sh
  Kerberos principal: tom@EXAMPLE.COM
  Email address: tom@example.com
  Random password: H9eFnMskdskk
  UID: 554000008
  GID: 554000008
  Password: True
  Member of groups: ipausers
  Kerberos keys available: True
```

IPA-enrollment of client machines

```
[root@wiki ~]# ipa-client-install
Discovery was successful!
Hostname: wiki.example.com
Realm: EXAMPLE.COM
DNS Domain: example.com
IPA Server: ipa.example.com
BaseDN: dc=example,dc=com

Continue to configure the system with these values? [no]: yes
Synchronizing time with KDC...
User authorized to enroll computers: admin
Password for admin@EXAMPLE.COM:
[...]
Configured sudoers in /etc/nsswitch.conf
Configured /etc/sssd/sssd.conf
[...]
Hostname (wiki.example.com) not found in DNS
DNS server record set to: wiki.example.com -> 192.168.100.220
Adding SSH public key from /etc/ssh/ssh_host_rsa_key.pub
[...]
Client configuration complete.
```

IPA-enrollment with one time password

```
[admin@ipa ~]$ ipa host-add wiki.example.com --random
-----------------------------
Added host "wiki.example.com"
-----------------------------
  Host name: wiki.example.com
  Random password: E0d-JEC4-Iwp
  Password: True
  Keytab: False
  Managed by: wiki.example.com
```

- Use `--force` to create the host record when it cannot be found in DNS. The host can update its own DNS record upon IPA-enrollment.

```
[root@wiki ~]# ipa-client-install --password E0d-JEC4-Iwp --unattended
[...]
Client configuration complete.
```

- Admin's password is not needed on the host being IPA-enrolled, just host's OTP.

Example: HBAC with ssh

```
[admin@ipa ~]$ ipa hbacrule-find allow_ssh
--------------------
1 HBAC rule matched
--------------------
  Rule name: allow_ssh
  Enabled: TRUE
  Users: tom
  Host Groups: linux-servers
  Services: sshd
----------------------------
Number of entries returned 1
----------------------------
```

```
[tom@client ~]$ ssh tom@server.example.com id
tom@server.example.com's password:
uid=554000008(tom) gid=554000008(tom) groups=554000008(tom) context=unconfined_u:
```

- Host `server.example.com` must be in host group `linux-servers`.
  - Quiz question: how to figure out host's group membership?
- Do not forget to disable `allow_all` rule for HBAC to work properly.

Example: ssh with Kerberos

```
[tom@client ~]$ kinit tom@EXAMPLE.COM
Password for tom@EXAMPLE.COM:
[tom@client ~]$ ssh -o 'GSSAPIAuthentication yes' tom@server.example.com id
uid=554000008(tom) gid=554000008(tom) groups=554000008(tom) context=unconfined_u:
```

Cross-realm trust

- Active Directory users can access Linux machines and services run in IPA realm.
- Enable trust support in IPA

```
[root@ipa ~]# ipa-adtrust-install --netbios-name=EXAMPLE -a password
```

- Set up DNS forwarding in IPA for the AD domain

```
[root@ipa ~]# ipa dnsforwardzone-add addomain.com \
        --forwarder=10.1.2.3 --forward-policy=only
```

- Set up DNS forwarding in AD to the IPA domain

```
C:\> dnscmd 127.0.0.1 /ZoneAdd EXAMPLE.COM /Forwarder 192.168.100.133
```

- Establish two-way trust

```
[root@ipa ~]# ipa trust-add --type=ad ADDOMAIN.COM --admin Administrator .
```

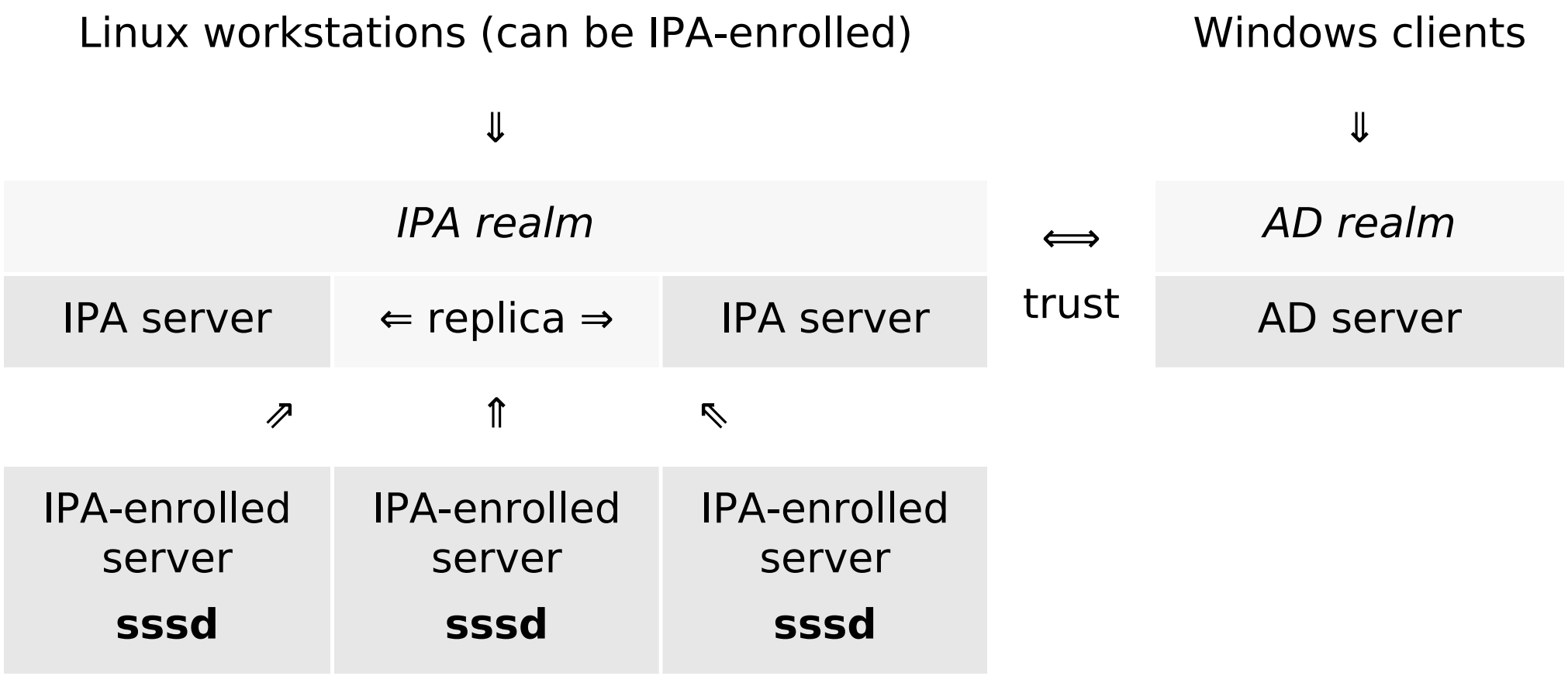
HBAC for cross-realm trust

- Create external group in IPA with AD group as member.
- Make the external group a member of a POSIX group.
- Use the POSIX group in HBAC rule.

```
[admin@ipa ~]$ ipa group-add-member ad-admins-external \
             --external 'linux-admin@ADDOMAIN.COM'
[member user]:
[member group]:
  Group name: ad-admins-external
  External member: S-1-5-21-2441374837-362968615-2867366494-1114
  Member of groups: ad-admins
  Indirect Member of HBAC rule: allow_ssh
-------------------------
Number of members added 1
-------------------------
```

- If bob is AD user in AD group `linux-user`, he can ssh to Linux hosts that are (members of host groups) listed for HBAC rule `allow_ssh`, without providing password.

The architecture



- The arrows show the direction of enrollment / trust.
- IPA-enrolled servers do not need to know anything (be configured to know) about the AD realm to serve AD users.

The goal

- Use the tools that work for OS-level authentication for Web applications as well.
- Easier deployment of Web applications within organization.
- Kerberos single sign-on (SSO), cross-realm trusts, HBAC, OTP ... for free.

IPA-enrolled Web server:

Web application
Apache HTTP server Modules
sssd

- Let authentication, identity operations, and access control be handled by Apache modules, and consumed by Web applications.

Needed pieces

- Account validation / access check for Kerberos-based authentication.
- If application has logon form for internal authentication, make it possible to plug in PAM easily, while not changing the user experience.
- Retrieve needed user attributes like email address or full name and group membership of authenticated users and deliver the information to applications.
- Applications will (passively) consume the results, just like they do with `REMOTE_USER` for Basic Authentication.
- No implementation of active authentication or identity operations needed in applications.

PAM for Web applications

	HTTP request processed by Apache server
	↓
Authentication module	mod_auth_kerb, mod_auth_gssapi, any other module
Authorization provider module	~~require valid-user~~ **mod_authnz_pam** `require pam-account <PAM-service-name>`

- Configure `/etc/pam.d/<PAM-service-name>`.
  - Use any PAM service name you want: httpd, wiki, foreman, ...
  - Use matching HBAC service name for HBAC check via sssd to work.
- Especially useful for SSO that should not reach applications.

PAM for applications' logon forms

<table>
<tr><td></td><td colspan="2">User submits application's standard logon form</td></tr>
<tr><td></td><td colspan="2">↓</td></tr>
<tr><td rowspan="4">Module</td><td colspan="2">Module mod_intercept_form_submit
intercepts the POST HTTP request</td></tr>
<tr><td colspan="2">PAM auth is run with [login, password] pair
(when found)</td></tr>
<tr><td>Authentication passes</td><td>Authentication fails</td></tr>
<tr><td><code>REMOTE_USER</code>
is set to login</td><td><code>EXTERNAL_AUTH_ERROR</code>
is set to PAM message</td></tr>
<tr><td>Application</td><td>Consumes <code>REMOTE_USER</code></td><td>Gets chance to
authenticate internally</td></tr>
</table>

PAM for apps' logon forms (cont'd)

- The same look of the logon screen.
- Authenticating against central identity provider.
- And access control check.
- No 401 status ever.
- It uses `mod_authnz_pam` internally.

Additional user information

- Web applications need more than just login name.
- Especially when applications autocreate user records in their internal databases based on access of externally authenticated users.
- Additional attributes for nice user experience.
  - Email address, full name, phone number, ...
- Group membership for application-level authorization and roles.
- Module **mod_lookup_identity** uses D-Bus interface of SSSD to retrieve additional data about authenticated users.
- New environment variables beyond `REMOTE_USER`:
  - `REMOTE_USER_EMAIL`, `REMOTE_USER_FULLNAME`, ...
  - `REMOTE_USER_GROUPS`; `REMOTE_USER_GROUP_N`, `REMOTE_USER_GROUP_1`, `REMOTE_USER_GROUP_2`, ...

Module overview

<table>
<tr><th rowspan="2">Authn Method</th><th colspan="3">Apache Modules</th></tr>
<tr><th>Authentication</th><th>Access Check</th><th>Extra User Info</th></tr>
<tr><td>Application</td><td colspan="3">None</td></tr>
<tr><td rowspan="2">GSSAPI</td><td>mod_auth_kerb</td><td rowspan="5">mod_authnz_pam</td><td rowspan="6">mod_lookup_identity</td></tr>
<tr><td>mod_auth_gssapi</td></tr>
<tr><td>SAML</td><td>mod_auth_mellon</td></tr>
<tr><td rowspan="2">Certificate</td><td>mod_nss</td></tr>
<tr><td>mod_ssl</td></tr>
<tr><td>Form</td><td colspan="2">mod_intercept_form_submit</td></tr>
</table>

How can applications use the new capabilities

- Many applications already support REMOTE_USER authentication, from HTTP Basic Authentication days.
- Authentication should ideally happen on isolated location, with internal sessions initiated.
- Allow/expect REMOTE_USER to be consumed when processing HTTP POST submission of logon form.
- When user is externally authenticated, process other REMOTE_USER_* environment variables.
- Add support for external groups and external group membership, map to internal application groups and/or roles.
- Amend Apache configuration, configuration scripts, ...

Benefits for Web applications

- Applications become accessible by all users in the organization.
  - Including Windows users.
  - With centralized access control.
- No more manually managing users in applications' databases needed.
- User records get auto-provisioned and kept in sync.
- Single sign-on with HBAC
  - Password-based authentication also available, including OTP.
- Application admins still locally manage mapping of groups to roles or authorization permissions.
  - Use user group membership from the central identity provider.

Conclusion and references

- Spacewalk, Foreman, and ManageIQ already take advantage of the new authentication options.
- Django proof of concept finished.
- Your favorite application not supporting Kerberos or IPA's HBAC?
  - We might not be able to enhance it ourselves but we will be happy to help people who would like to add the features.

- www.freeipa.org/page/Web_App_Authentication
- www.freeipa.org/page/Environment_Variables#Proposed_Additional_Variables
- www.freeipa.org
- fedorahosted.org/sssd/
- www.adelton.com/docs/idm/