External and Federated Identities on the Web

Jan Pazdziora
Sr. Principal Software Engineer
Identity Management Special Projects, Red Hat

ApacheCon:
CORE
Europe
1st October 2015
Applications get deployed in large organizations and across them.

What are the recommendations for application developers concerning authentication and access control methods and protocols that they should support?
Typical small application approach

- Own database schema for users, passwords, and access policies.

Diagram:
- User's browser
- Web application
- Apache HTTP Server
- Database (possibly on separate host)
- Web server host

Pass control flow:
- User's browser → Web application → Apache HTTP Server → Database

Authentication process:
- auth
Application-level authentication

Pros:
- Integrated user management, without external dependencies.
- Closely linked to custom data, including access control.

Cons:
- In large organizations, users already exist in central identity management systems.
  - LDAP, IdM/FreeIPA, Active Directory (AD), ...
  - With policies for access control.
  - No one will sync the users manually.
Application-level LDAP auth

Developers told user identities are in LDAP

- Typical solution: add LDAP support to the application (or framework).

```
user's browser  →  Apache HTTP Server  →  Web server host

Web application

LDAP to auth

⇑

pass control

identity management server
```
Application-level LDAP auth

Pros:
- Organization's primary identity source is used.

Cons:
- Getting all aspects of LDAP operations right is not easy.
  - Authentication to the LDAP server.
  - Failover, DNS discovery.
  - Multiple domains and forests (AD).
- Every new framework / project starts from scratch.
- Usually only login + password supported.
- Kerberos / GSSAPI, smartcard, or two-factor authentication usually done via different means.
Front-end (Apache) authentication

Front-end accepted for some setups

- Application supports some basic authentication methods.
- For complex ones, authentication front end (Apache HTTP Server) is used.

```
user's browser → Web application
                 ↑
                 pass REMOTE_USER

Apache HTTP Server
mod_authnz_ldap

Web server host

LDAP to auth

identity management server
```
Front-end (Apache) authentication

Pros:
- Single solution (Apache modules) possible for various deployments.
- Failover, caching.
- Support for Active Directory Global Catalog.

Cons:
- No support for multiple forests (AD).
- No support for Group Policy Objects (GPO) or centralized host-based access control.
- While authorization can be done on Apache level, fine-grained access control in the application not easy.
- Fewer possibilities for nice user management from the application.
OS-level tools for Web services

System Security Services Daemon (SSSD)

- Used for logon to system (ssh), can be used for Web services as well.

User's browser → Web application

- pass REMOTE_USER
- + REMOTE_USER_EMAIL, REMOTE_USER_GROUP_N, REMOTE_USER_GROUP_1, ...

Web server host → Apache HTTP Server

- mod_authnz_pam
- mod_lookup_identity

SSSD

⇒ identity, authn, authz sources

⇒ auth + get info
OS-level tools for Web services

Pros:
- Single solution (Apache modules) for various applications.
- Failover, caching, DNS discovery, cross-forest trust support (Windows users can log in to Web services on Linux).
- Host-based access control (with IdM/FreeIPA) and GPO support (with AD, via PAM) for central access management.
- Application can get additional information, not just REMOTE_USER.
  - Better user experience.
  - Fine-grained access control in apps based on group membership.

Cons:
- Fewer possibilities for nice user management from the application.
## Setups within organizations

### Apache modules typical in large organizations

<table>
<thead>
<tr>
<th></th>
<th>Authentication</th>
<th>Access Check</th>
<th>Extra User Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kerberos / GSSAPI</td>
<td>mod_auth_kerb, mod_auth_gssapi</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Certificate</td>
<td>mod_ssl</td>
<td>mod_authnz_pam</td>
<td>mod_lookup_identity</td>
</tr>
<tr>
<td></td>
<td>mod_nss</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2FA</td>
<td>mod_auth_form (provider PAM)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>mod_lookup_identity</td>
<td></td>
<td>mod_lookup_identity (uses mod_authnz_pam internally)</td>
</tr>
</tbody>
</table>
Authentication

- Identity is established and verified.
- The identifier (login name) can be further tweaked.

SSLVerifyClient require # authenticate with mod_ssl
SSLUserName SSL_CLIENT_CERT # r->user = whole certificate data
LookupUserByCertificate On # find the user in IPA via SSSD
# and set r->user = user login

- With mod_auth_gssapi and GSS-Proxy, privilege separation possible. Apache process does not need access to keytab.

[service/HTTP]
mechs = krb5
cred_store = keytab:/etc/gssproxy/http.keytab
cred_store = ccache:/var/lib/gssproxy/clients/krb5cc_%U
euid = 48

- 2FA (password + one-time code) possible if backend supports it.
Authorization / access control

- Not every authenticated user should be let in.

- In Windows domain, everyone has Kerberos ticket granting ticket.

- Avoid require valid-user.

- Editing Apache .conf files is not very flexible.

- Delegation to centralized policy definition (IdM, AD) is preferred.

```
# require group crm-admins # do not define policy locally
require pam-account crm-production # delegate via PAM

# /etc/pam.d/crm-production
auth required pam_sss.so # pam_sss.so for SSSD
account required pam_sss.so # or other PAM module
```

- Applications can run fine-grained access mechanisms based on user group memberships, obtained from external identity sources.
For better user experience, applications need additional attributes.

For application-level roles and permissions, applications need group information.

Since we have just authenticated/authorized the user in Apache, why not get their attributes as well?

- LookupUserAttr mail REMOTE_USER_EMAIL " "
- LookupUserAttr givenname REMOTE_USER_FIRSTNAME
- LookupUserAttr sn REMOTE_USER_LASTNAME
- LookupUserGroupsIter REMOTE_USER_GROUP

We map SSSD's attributes to environment variables.

Application does not need to reach out for the information.

And hit another round of "where to get it from? how to authenticate to that source?" issues.
Recommendations

What are the recommendations for application developers concerning authentication and access control methods and protocols that they should support, for deployments within large organizations?

- Accept authentication/authorization result from front-end server.
  - REMOTE_USER or other mechanism used.

- Accept additional user attributes and group membership.
  - REMOTE_USER_EMAIL
  - REMOTE_USER_GROUP_N
  - REMOTE_USER_GROUP_1
  - REMOTE_USER_GROUP_2
  - or REMOTE_USERGROUPS as colon-separated list
Users across organizations

Application hosted by a provider

- Users are managed by different organization (the customer).
- Likely no Kerberos as no HTTP/ service keytab from customer's KDC.
- No way to reach into customer's internal network, to IdM/FreeIPA or AD.
- Is our recent recommendation faulty?

- Actually, it gets even more valid across organizations.
- With federation, all information about the user comes from the initial authentication exchange.
Security Assertion Markup Language

- Getting identity of authenticated user, their attributes, and authorization information from Identity Provider (provided by customer).
- The single sign-on on the Web.
- Client side (Service Provider, that hosted solution) implemented by Apache module: mod_auth_mellon.
- Previous agreement/setup with metadata and public key exchange needed.
  - Good or bad thing depending on point of view.
- No communication between the Service Provider and Identity Provider needed.
  - Browser handles the redirects of signed data.
SAML workflow

Across organizations' boundaries

1. User's browser

2. HTTP redirects

3. Ipsilon or other IdP (possibly with SSSD)

4. Apache mod_auth_mellon

5. Web server host in provider's network

6. IdM, AD, LDAP

7. Identity Provider host

8. Identity source
Ipsilon

Three-command SAML Identity Provider

- Install packages, configure, restart Apache.

```bash
yum install -y ipsilon ipsilon-saml2 ipsilon-authform \  
   ipsilon-authgssapi ipsilon-infosssd ipsilon-tools-ipa
ipsilon-server-install --ipa yes --saml2 yes \  
   --form yes --gssapi yes \  
   --gssapi-httpd-keytab /etc/httpd.keytab \  
   --info-sssd yes --info-sssd-domain example.test
service httpd restart
```

- Written in Python, with protocols and providers as plugins.
- Integrates with FreeIPA/SSSD.
- SAML, OpenID, Mozilla Persona available.
- OpenID Connect actively developed.
- Available in Fedora, coming to RHEL/CentOS soon.
Service Provider

mod_auth_mellon configuration

- Against Ipsilon server:
  ```bash
  yum install -y epsilon-client
  epsilon-client-install --saml-idp-url https://idp.example.com/idp \
  --saml-sp-name application --saml-auth /application/login
  ```

- Against generic SAML server:
  ```bash
  yum install -y epsilon-client
  epsilon-client-install \ 
  --saml-idp-metadata https://idp.example.com/saml/metadata \ 
  --saml-auth /application/login
  ```

Generated SP metadata needs to be transferred to IdP manually.
Service Provider

mod_auth_mellon configuration

- Mapping SAML response attributes to environment variables:
  
  ```
  MellonSetEnvNoPrefix REMOTE_USER_EMAIL email
  MellonSetEnvNoPrefix REMOTE_USERGROUP groups
  ```

- The same multivalued variables as mod_lookup_identity with
  
  ```
  MellonEnvVarsIndexStart 1
  MellonEnvVarsSetCount On
  # MellonMergeEnvVars On "::"
  ```

  Version 0.11.0 needed for these directives.

- These are currently not setup by ipsilon-client-install automatically.
Which part of the setup should we show first?

Web app on IPA-enrolled server

IdM / IPA server ⇔ cross-forest trust

Web app on externally hosted SP server

IdP on IPA-enrolled server

mod_auth_mellon

mod_authnz_pam with SSSD

AD
Conclusion

What are the recommendations for application developers concerning authentication and access control methods and protocols that they should support, for deployments across organizations?

- The same as for setups within organizations.
- Teach applications to accept REMOTE_USER and REMOTE_USER_*
- The actual protocol/setup is deployment specific, using Apache HTTP Server modules.

By merely changing Apache configuration, we can switch the application from intra-organizational to federated setup.

- Additional application-level changes are not needed for single IdP setups when no selection is required.

Currently looking at mapping of claims in mod_auth_openidc to REMOTE_USER_* for OpenID Connect federation.
References

- www.adelton.com/apache/mod_authnz_pam/
- www.adelton.com/apache/mod_lookup_identity/
- github.com/UNINETT/mod_auth_mellon
- fedorahosted.org/ipsilon
- github.com/pingidentity/mod_auth_openidc
- www.freeipa.org/page/
  Environment_VARSIables#Proposed_Additional_VARSIables
- Jan Pazdziora <jpazdziora@redhat.com>