Break ice or don’t login twice: FreeIPA and OAuth 2.0

Alexander Bokovoy // Francisco Triviño García
Red Hat
• FreeIPA core developers
• Engineers at Red Hat
FreeIPA

• Identity management solution:
  • provides centralized infrastructure to manage POSIX identities across a fleet of Linux machines
  • combines 389-ds LDAP server, MIT Kerberos, BIND DNS server, SSSD, Samba, and Python-based management tools
  • often seen as ‘Active Directory for Linux’ but this is not exactly correct comparison
  • Depends on a lot of OS components working together, can be used as a canary to detect breakage in many packages
  • Used as a core of Fedora Accounts system
FreeIPA at an operating system level

- Identity and access information
  - user and group POSIX information for Linux environments through SSSD
  - user authorization through SSSD host-based access controls
- Authentication
  - Centralized Kerberos authentication with different authentication methods
  - Single sign-on to system services
  - Centralized management of SSH public keys
FreeIPA integration to non-operating system services
FreeIPA as a backend provider

• Identity provider integration:
  • Direct identity backend to a web service with LDAP ‘driver’
  • SSSD as an identity backend to a web service
    • Ipsilon and Keycloak
    • Apache module mod_lookup_identity
    • NGINX module nginx_http_lookup_identity_module

• Authentication integration:
  • LDAP BIND
  • SPNEGO/Kerberos
    • Apache module mod_auth_gssapi
  • PAM authentication via SSSD PAM module
    • Apache module mod_authnz_pam
    • NGINX module nginx_http_authnz_pam_module
Disadvantages

• Applications authors haven’t really mastered LDAP and Kerberos
  • some frameworks do allow for extensibility but documentation isn’t great
• Typical integration approaches struggle to scale
  • a single LDAP server in a configuration
  • lack of support for more than ‘username+password’ methods
  • Java-based frameworks have outdated Kerberos support, aren’t aware about features added since 2010
  • Java-based frameworks struggle to integrate with UNIX domain sockets
  • Micro-services often cannot be assumed to use system-wide domain enrollment details
• Web services moved on to OAuth 2.0 authorization framework
  • OAuth 2.0 methods rely on browser redirects
• Web services moved on to OAuth 2.0 authorization framework
  • Identity Provider (IdP) handles authentication and authorization, one place to focus on instead of every single app
  • Applications rely on IdP-issued grant to operate
• Web services map OAuth 2.0 subjects, not system-level ‘users’, it gives a bit of flexibility to map ‘POSIX’ users
FreeIPA as a consumer of external resources

Consume external identities

• Trust to Active Directory

Consume external authentication

• FreeIPA already allows to authenticate against an external source with RADIUS protocol
  • exposed through a Kerberos pre-authentication method
  • user details stored in FreeIPA, authentication handled by external source

• RADIUS support has some limitations:
  • single RADIUS server end-point per each user
  • only supports ‘PIN + token’ opaque scheme
  • cannot support conversation protocols
OAuth 2.0 moves authentication step to IdP

- Authentication is not visible to OAuth 2.0 clients, they ask IdP for a grant to access resources instead
  - IdP authenticates the user, if needed, and asks the user to authorize the request
  - All this implies use of a browser and HTTP-based redirects
  - Hard to integrate without browsers being available
  - OAuth 2.0 has few authorization flows to address different use cases; they all still need the browser to be present somewhere

We want to use OAuth 2.0 framework flows to log in over SSH

- How can we avoid running a browser on the server side?
Wait, this is a familiar issue, right?

Captive portals at public Wi-Fi access points
- how to login as a network-bound user if a Wi-Fi access point wants to show a browser window to ‘click the checkbox’ before the login?

Passwordless authentication
- how can we help to improve Linux login experience?

How to run untrusted code prior to login?
Demo 1: log in over SSH
Using OAuth 2.0 device authorization grant flow

1. **End User at Browser**
2. **Device Client**
3. **Authorization Server**

   - (A) **Client Identifier**
   - (B) **Device Code, User Code & Verification URI**
   - (C) **User Code & Verification URI**
   - (D) **End user reviews authorization request**
   - (E) **Polling with Device Code and Client Identifier**
   - (F) **Access Token (& Optional Refresh Token)**

End User at Browser      | Device Client      | Authorization Server
Detailed guides

- FreeIPA workshop: chapter 12: Authentication against external Identity Providers
- FreeIPA design documents: general design and IPA API design for IdP
Actual flow for FreeIPA

FreeIPA client code is scoped in the MIT Kerberos pre-authentication module
• provided by SSSD project as sssd-idp subpackage
• tells KDC “I support OAuth 2.0 method, consider it”
• shows KDC response as “Authenticate at https://… and press ENTER.”

FreeIPA server side is reusing RADIUS helper `ipa-otpd`
• KDC side of the MIT Kerberos pre-authentication module triggers IdP support
• KDC asks RADIUS helper `ipa-otpd` to handle it
• `ipa-otpd` calls SSSD-provided `oidc_child` helper to talk OAuth 2.0 to user-specific IdP
• on successful authorization response from IdP, KDC issues a Kerberos ticket

User runs browser elsewhere

Kerberos ticket is issued for the user
Possible integration

Authentication is done by an external IdP

Authorization grant is turned into a Kerberos ticket by FreeIPA KDC
  • Kerberos authentication indicator “idp” is assigned to the ticket

Kerberos ticket is consumed by an IPA-enrolled application
  • application can check the authentication indicator and deny non-IdP access
    • pam_sss_gss PAM module can be used to limit sudo access
    • mod_auth_gssapi Apache module can be used to limit authentication to web sites
OAuth 2.0 device authorization grant flow

Tested against multiple public IdPs
  • Keycloak / Red Hat Single Sign-On
  • Google
  • Github
  • Microsoft Azure
  • Okta

Does not work against IdPs which do not implement OAuth 2.0 device authorization grant
  • Ipsilon (Fedora Accounts)
  • Gitlab
Demo 2: Keycloak and Nextcloud
A generic backend dream

System for cross-domain identity management (SCIM v2.0)

- RFCs 7642 / 7643 / 7644
- Automation of the user identity information exchange between identity domains
- Supported by many proprietary identity providers
- Exposes user and group information and access methods over REST API (over HTTPS)
ipa-tuura - a proof of concept SCIMv2 bridge between FreeIPA and IdPs
  • tuura – Finnish word for an ice chisel, a tool for breaking ice

Current scope
  • Supports FreeIPA, LDAP, and Active Directory as read sources
  • Supports FreeIPA, LDAP, and Active Directory as writable targets
  • Rudimentary password authentication support
PoC code:

• Django application combining IPA API and a SCIMv2 Python module:
  freeipa/ipa-tuura

• Keycloak user store plugin to connect over SCIMv2:
  justin-stephenson/scim-keycloak-user-storage-spi
Future plans
IPA-enrolled applications to benefit from OAuth 2.0 client support
  • e.g. Cockpit UI on each server to accept OAuth 2.0 authentication of IPA users
  • FreeIPA Web UI integration

Secure transition from OAuth 2.0 grant to Kerberos on behalf of a user
WebAuthn / FIDO2 support

OAuth 2.0 IdPs already have support for WebAuth tokens
  • FreeIPA 4.9.10+ can authenticate users with WebAuthn tokens through external IdP integration

SSSD plans to support FIDO2 tokens natively
  • Locally, with libfido2 first, to replace pam_u2f
  • Over Kerberos for integration with FreeIPA

Integrate with GNOME login

Enable passwordless FreeIPA deployments
SCIMv2 support

Turn PoC project freeipa/ipa-tuura into a production code

- Automate integration with known SCIMv2 providers
- Support more IdPs
- Add more authentication methods
- ...

22
Thanks!