Apache managed SSO for web-applications

with or without Kerberos Ticket

SambaXP 2026

Kees van Vloten

About me

Kees van Vloten

- Freelancer in infrastructure automation
- Using Samba in infrastructure since the 90’s
- Started in 2020 with integrations on Samba-AD

Agenda

- Apache authentication intro
- Kerberos authentication
- Authentication without ticket
- Alternative authentication method
- Additional components
- Setup in Apache
- Questions

Authentication - Apache

- Vhost authentication

```
<VirtualHost>
    <Location /<protected-uri>>
        AuthType <authentication-module-reference>
        <authentication-config>

        <other-options>
    </Location>

    Alias /<protected-uri> /<filesystem-path>
    <Directory /<filesystem-path>>
        <some-as-above>
    </Directory>
</VirtualHost>
```

- Some authentication modules

```
auth_basic.load   authn_core.load     authnz_external.load  authz_dbd.load        authz_user.load
auth_digest.load  authn_dbd.load      authnz_fcgi.load      authz_dbm.load
auth_form.load    authn_dbm.load      authnz_ldap.load      authz_groupfile.load
auth_gssapi.load  authn_file.load     auth_openidc.load     authz_host.load
authn_anon.load   authn_socache.load  authz_core.load       authz_owner.load
```

Authentication - applications

- Apache: authentication and authorization
- Application: fetches user
  - Environment: `REMOTE_USER`
  - Header:

```
<Location /<protected-uri>>
    <authentication-options>
    RequestHeader set "X-Forwarded-User" "%{X_REMOTE_USER}e"
    RewriteCond %{LA-U:REMOTE_USER} (.+)
    RewriteRule .* - [E=X_REMOTE_USER:%1,NS]
    <other-options>
</Location>
```

  - Module / application specific

Authentication - applications

- Exmple `mod_wsgi` for Python wsgi
  - vhost.conf

```
WSGIDaemonProcess <my_app_group> python-home=<my_app_path>
WSGIScriptAlias /<protected_uri> <wsi_file>.wsgi
process-group=<my_app_group>


<Location /<protected-uri>>
    <authentication-options>
    WSGIPassAuthorization On
    WSGIProcessGroup <my_app_group>
    WSGIApplicationGroup %{GLOBAL}
    <other-options>
</Location>
```

  - my_app.py

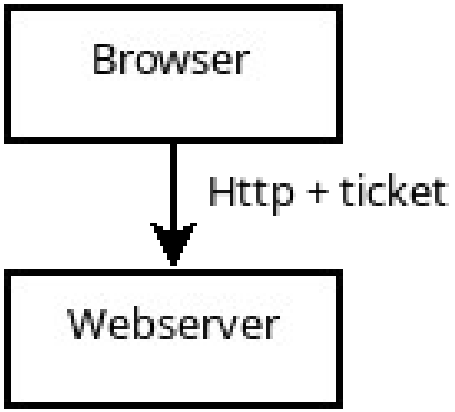
```
@route('/<protected_uri>')
def web_endpoint():
    user = request.auth[0]
```

Kerberos auth - server

- Kerberos module: `mod_gssapi`
  https://github.com/gssapi/mod_auth_gssapi
- Prevent re-auth on each request: `mod_session`
- Debian package: `krb5-kdc`
- Keytab file:

```
samba-tool domain exportkeytab -d 8 -principal=<principal>
# principal = http/<vhost-name><domain>[:<port>]
```

- Authentication flow:



Kerberos auth - server

- Protected resource configuration:

```
<Location /<protected-uri>>
    AuthType GSSAPI
    AuthName "My Organization Login"
    GssapiSSLonly On
    GssapiUseSessions On
    GssapiSessionKey key:<domain-wide-gssapi-session-key>
    GssapiCredStore keytab:<path_to_keytab>
    GssapiCredStore ccache:DIR:/run/apache2/krb5_ccache
    GssapiBasicAuth On
    GssapiAllowedMech krb5
    GssapiBasicAuthMech krb5

    Session On
    SessionCryptoPassphrase <domain-wide-gssapi-crypto-passphrase>
    SessionCryptoCipher aes256
    SessionCookieName gssapi_session_<domain> path=/;domain=<domain>;<attrs>
    SessionMaxAge <domain-wide-max-age>

    Require valid-user

    <other-options>
</Location>
```

Kerberos auth - browser

- Browser configurion:
  - Firefox:

```
open "about:config", set:
network.negotiate-auth.trusted-uris: <domain>,<domain2>
```

  - Chrome:

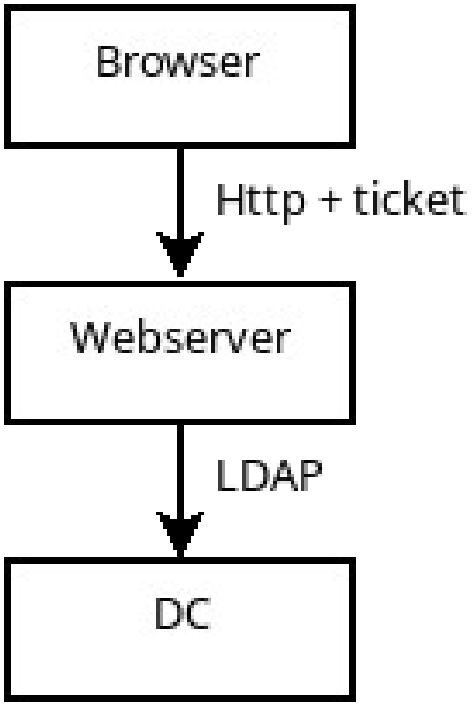
```
cat << EOF > /etc/opt/chrome/policies/managed/kerberos.json
{
  "AuthServerAllowlist": "*.<domain>,*<domain2>",
  "AuthNegotiateDelegateAllowlist": "*.<domain>,*<domain2>"
}
```

Kerberos auth - session

- Browser sends request:
  `GET <webserver>/<protected-uri>`
- Web-server:
  `401 Unauthorized`
  `WWW-Authenticate: Negotiate`
- Browser:
  `Authorization:Negotiate <ticket>`
- Web-server:
  `200 OK`
- Browser displays page

Kerberos auth – server authnz

- `mod_gssapi`: authentication, no authorization
- `mod_ldap` can do both but has no SSO
- Combine `mod_gssapi` + `mod_ldap`
- Authentication flow:



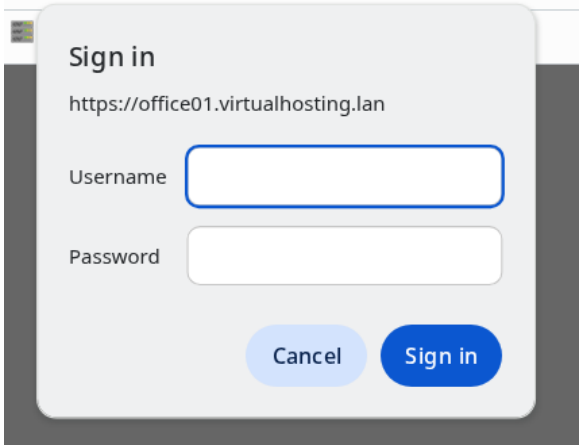
Kerberos auth – server authnz

- Protected resource configuration:

```
<Location /<protected-uri>>
    AuthType GSSAPI
    <gssapi-config>

    AuthLDAPURL "ldap://<dc1> <dc2>/<base-dn>?sAMAccountName?sub?
(&(objectCategory=person)(!(userAccountControl:1.2.840.113556.1.4.803:=2)))"

    AuthLDAPBindDN "<account-dn>"
    AuthLDAPBindPassword "<account-password>"
    AuthLDAPRemoteUserAttribute sAMAccountName

    <RequireAll>
        Require valid-user
        Require ldap-filter memberof:1.2.840.113556.1.4.1941:=CN=<group-dn>
    </RequireAll>

    <other-options>
</Location>

# <group-dn> = <permission-group>,<container-dn>
# <permission-group> = acl-app_<application-name>
```

Auth without ticket - fallback

- `Mod_gssapi` falls back to basic-auth
- Curl session:
  `curl -u <user>:<password> https://<webserver>/<protected-uri>`
  - Curl:
    `GET <webserver>/<protected-uri>`
    `Authorization:Basic <encoded user+password>`
  - Web-server:
    `200 OK`
- Browser shows basic-auth popup:



Auth without ticket – mod_form

- `mod_form` shows a custom html form for login
- Combine: `mod_gssapi` + `mod_ldap` + `mod_form`
- Authentication flow:

Source	Action
Browser	Get *protected-url*
Webserver	`mod_gssapi` if ticket ok: Response: content of *protected-url* – **flow ends** else: Response: redirect to `redirect.html`
Browser	Get `redirect.html`
Webserver	Response: set cookie `FORM_AUTH` + redirect to *protected-url*
Browser	Get page + passes cookie `FORM_AUTH`
Webserver	Response: `login.html`
Browser	Post login data to *protected-url*
Webserver	`mod_ldap` if credentials ok: Response: content of *protected-url* – **flow ends** else: Response: `login.html` (and go back to previous step)

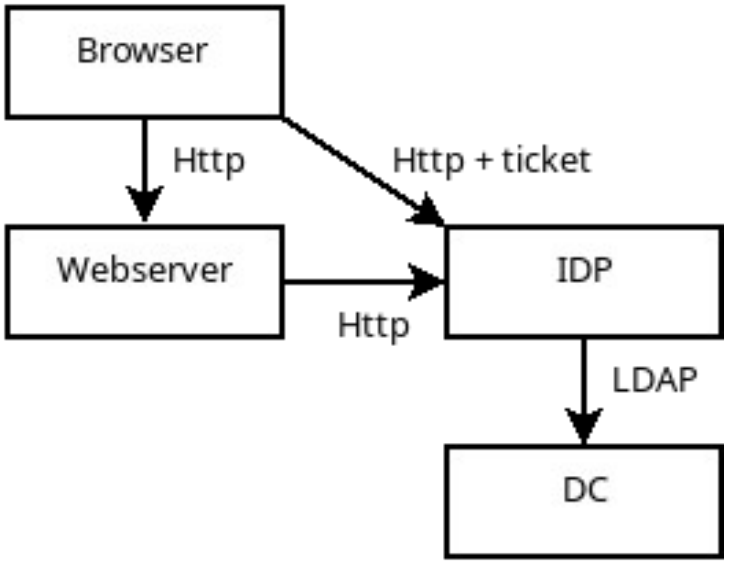
Auth without ticket – mod_form

- Protected resource configuration:

```
<Location /<protected-uri>>
    <If "%{HTTP_COOKIE} =~ /FORM_AUTH=true/">
        AuthType FORM
        AuthName "My Organization Login"
        ErrorDocument 401 /login.html

        AuthLDAPURL "ldap://<dc1> <dc2>/<base-dn>?sAMAccountName?sub?
(&(objectCategory=person)(!(userAccountControl:1.2.840.113556.1.4.803:=2)))"

        AuthLDAPBindDN "<account-dn>"
        AuthLDAPBindPassword "<account-password>"
        AuthLDAPRemoteUserAttribute sAMAccountName

        <session-cookie-config>

        <RequireAll>
            Require valid-user
            Require ldap-filter memberof:1.2.840.113556.1.4.1941:=CN=<group-dn>
        </RequireAll>
    </If>
    <Else>
        ErrorDocument 401 "<html><meta http-equiv=\"refresh\"
content=\"0;url=/redirect.html?path=%{REQUEST_URI}\"></html>"

        <gssapi+ldap-config>
    </Else>

    <other-options>
</Location>
```

Alternative auth method

- Looking for something better...
- No modules handle the authnz flow in webserver
- Let identity provider (IDP) handle authentication
- Options: Openid-connect (OIDC) or SAML
- Use OIDC
- Simplified flow:



Auth method oidc

- Authentication flow:

Source	Destination	Action
Browser	Webserver	Get *protected-url*
Webserver	IDP	Get IDP info
Webserver	Browser	Response: redirect to IDP
Browser	IDP	Get login page
Browser	IDP	Post login data
IDP	Browser	Response: redirect to webserver with unique-code
Browser	Webserver	Get *protected-url* + passes unique-code
Webserver	IDP	Get token for unique-code + get userinfo
Webserver	Browser	Response: Set cookie + redirect to *protected-url*
Browser	Webserver	Get *protected-url* + passes cookie
Webserver	Browser	Response: content of *protected-url*

Infrastructure

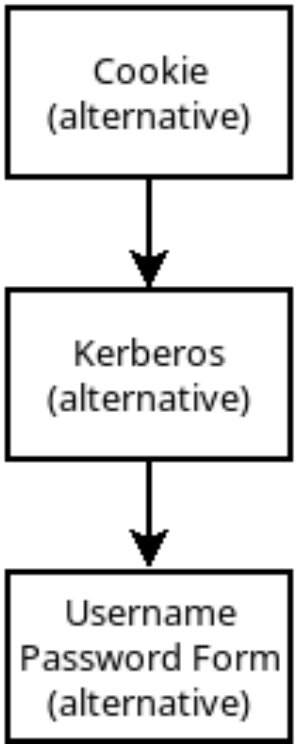
- Keycloak as identity provider
  https://www.keycloak.org/
  - Frontend: OIDC
  - Backend: AD via Kerberos and LDAP
- Apache: `mod_oidc`
  https://github.com/OpenIDC/mod_auth_openidc
  - Sends browser to Keycloak for authentication
  - Fetches userinfo and token
  - Token contains claims used for authorization
- AD permission-groups follow a naming convention:
  - `acl-app_<application-name>`

Keycloak - setup

- Extra provider: Regex filter group mapper https://github.com/mrDFX/keycloak-regex-filter-group-mapper
- User federation: Kerberos and LDAP
- Authentication flows: Browser flow: `auth_browser`
- Client scopes: `domain_permissions`
- Clients: `cli_webserver`

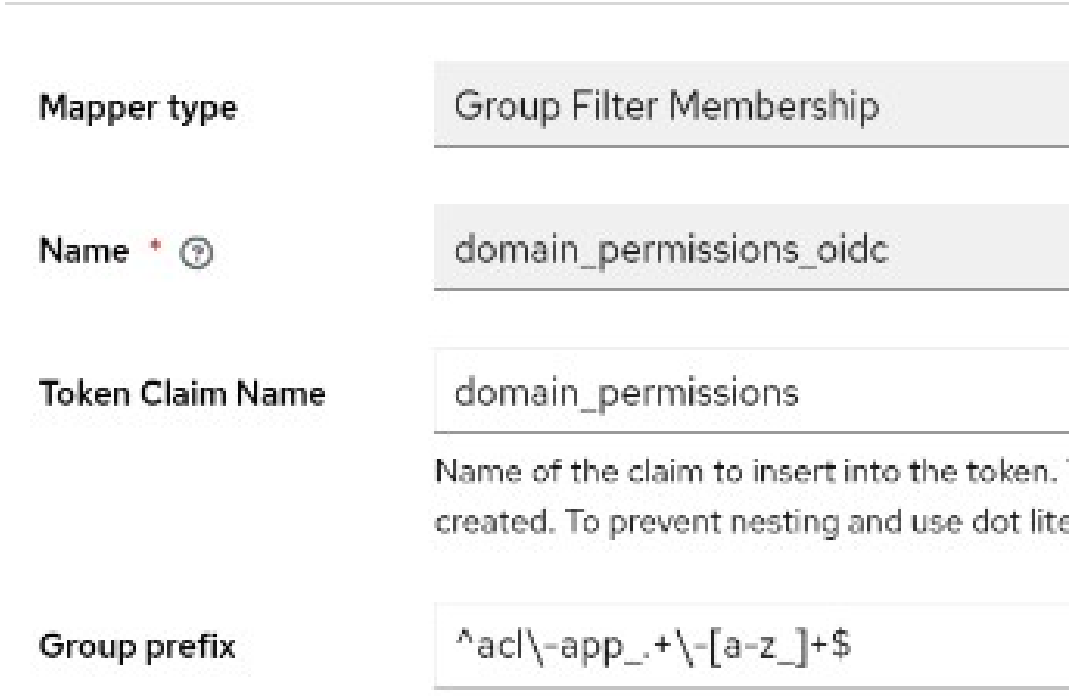
Keycloak – setup

- Browser flow `auth_browser`
  - Cookie?
  - Kerberos?
  - Login form



Keycloak - setup

- Client scope `domain_permissions`
  - Uses provider: Regex filter group mapper
  - Creates claim: `domain_permissions`
  - Claim contains all permission groups for applications



Keycloak - setup

- Client `cli_webserver`
  - From Keycloak perspective the webserver is a client
  - Settings:
    - Root URL: `https://<webserver>`
    - Valid redirect URIs:
      `/login/oidc-login`
      `/protected-uri/*`
    - Credentials: set a `client-secret`
    - Client scopes:
      `domain_permissions`
      `profile`

      client scope `openid` is included by default

Setup in Apache - vhost

- vhost configuration:

```
OIDCCryptoPassphrase <domain-wide-oidc-crypto-passphrase>
OIDCProviderMetadataURL
https://<keycloak-host>/auth/realms/<domain>/.well-known/op
enid-configuration

OIDCClientID cli_webserver
OIDCClientSecret <client-secret>
OIDCRedirectURI https://<webserver>/login/oidc-login
OIDCScope "openid profile domain_permissions"
OIDCRemoteUserClaim preferred_username
OIDCCookieDomain <domain>
OIDCCookie oidc_session_<domain>

# <keycloak-host> must be reachable by browser
# OIDCRedirectURI must be absolute,
#   otherwise OIDCCookieDomain cannot be set
```

Setup in Apache - resource

- Protected resource configuration:

```
<Location /<protected-uri>>
    AuthType openid-connect
    AuthName "My Organization Login"

    <RequireAll>
       Require valid-user
       Require claim domain_permissions:<permission_group>
    </RequireAll>

    <other-options>
</Location>

# <permission-group> = acl-app_<application-name>
```

Thank you

- Questions?

- Reach out via:
- Github: https://github.com/kvvloten
- Linked-in: https://linkedin.com/in/keesvanvloten/
- Email: keesvanvloten@gmail.com