Secure Command Line Solution for Token-based Authentication

Dave Dykstra, dwd@fnal.gov

vCHEP 2021

20 May 2021

Fermilab

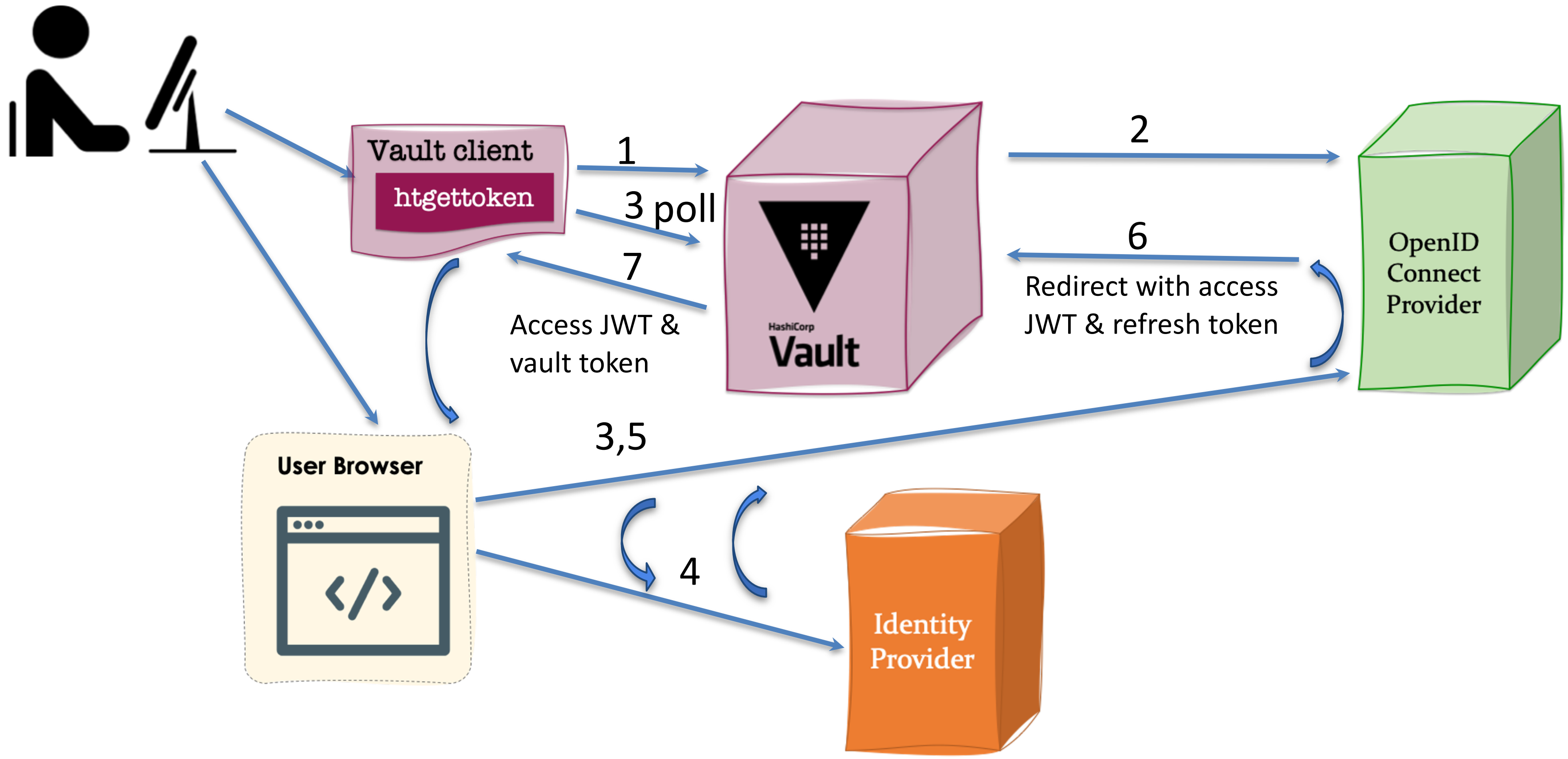
Background

- The X.509 user credentials the grid depends on never came into common use in industry
- We're now moving to the new industry standard Oauth2/Open ID Connect (OIDC) and JSON Web Tokens (JWTs), which are good but introduce challenges:
  - OIDC assumes a web browser world, and our tools are mostly command line
  - Need a new way to renew tokens for grid jobs, that is, a new secure storage for refresh tokens to replace MyProxy
- The WLCG authorization working group considered oidc-agent as a command line client, but it wasn't a good match
  - Didn't solve the MyProxy replacement problem
  - Not as user friendly as it could be, needing to encrypt refresh tokens with a user-typed passphrase whenever restarting

Solution: Vault with htgettoken

- Hashicorp Vault
  - Popular open source general purpose secure secret store
  - Very flexible plugin architecture and client/server API, and secrets are stored like in a filesystem
  - Has existing OIDC and Kerberos plugins
    - Needed some extensions, submitted as pull requests
  - Integrates well with both Indigo IAM and CILogon OIDC Providers
  - Manages access with its own tokens
  - We use it to store long-lived refresh tokens for many users
- htgettoken (ht from High Throughput Computing)
  - Relatively simple custom python command line Vault client to automate the flows
  - Initially authenticates via OIDC & a web browser
  - Long life (~1 month, renewable) refresh token stays in Vault, limited life (~1 week) Vault tokens and even shorter life (~1 hour) access JWTs stored unencrypted in local files
  - Follows WLCG bearer token discovery standard for local filename
  - Uses Vault token to get more access tokens, or renews Vault access with Kerberos

htgettoken with Vault initial OIDC flow



htvault-config configuration package

- Package for configuring Vault for use with htgettoken
  - Automates all the installation and setup of Vault
  - Configuration done through simple, flexible yaml files
  - Includes a modified Hashicorp plugin and an added puppetlabs plugin
  - Supports an option of using 3 servers for high availability using a builtin Vault capability
  - Now available in OSG yum distribution along with vault and htgettoken

Capability sets, issuers, and roles

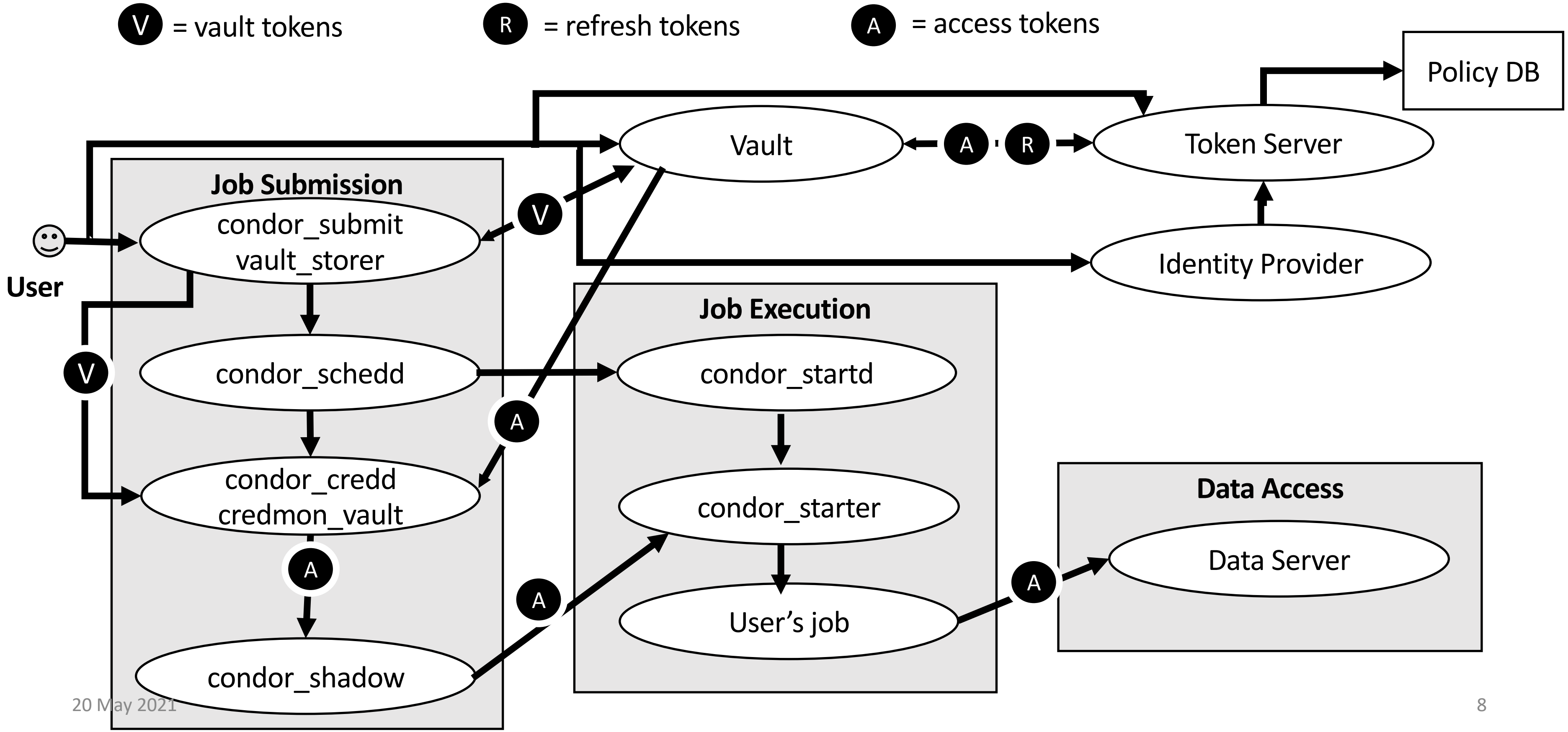
- JSON Web Tokens can be tailored to minimum privilege by use of “capability” scopes with access limits (and also specific audiences)
- The knowledge of what scopes are allowed per user is maintained by the OIDC Provider, aka the token issuer
  - Not known by clients
- We configure Vault to request scope wlcg.capabilityset:/group which the token issuer translates into a set of capability scopes
  - Groups correspond to VOs and roles within those VOs
  - Vault configuration is done per issuer, with one VO per issuer, and each role maps to a wlcg.capabilityset, for example:

```
   htgettoken -a htvault.cern.ch –i cms –r production
=> https://cms-auth.web.cern.ch, wlcg.capabilityset:/cms/pro
```

HTCondor integration

- htgettoken and Vault have been integrated into HTCondor
  - condor_submit can be configured to automatically invoke htgettoken as needed and store a vault token internally that is used to get new short-lived access tokens pushed to jobs
  - Submit file specifies issuer, optional role, and optionally can choose reduced audience and/or scopes
    - May obtain more than one token for a job
  - Available in OSG build of htcondor-9.0.x
    - HTCondor team's own distro includes the source code but is waiting for 9.1 series to build it by default

Token flow with HTCondor and Vault



Support for “robot” (unattended) operation

- Important for tasks such as production job submission
- Vault administrator can create indefinitely renewable vault tokens
  - Could be automated by a web service
- htgettoken & htvault-config also support use of robot kerberos credentials to get new vault tokens
  - Robot kerberos credentials are long lived, stored unencrypted
  - Principals are in the form “user/purpose/machine.name”
    - “user” can also be a group login, for example “dunepro”
  - User (or authorized user for a group) does OIDC authentication once but specifies htgettoken `--credkey` option matching Kerberos principal to store refresh token in subpath under the user’s Vault secrets path
    - The same htgettoken command can be used with robot Kerberos credentials

Conclusions

- Getting credentials almost as hidden as they can be
  - Users with Kerberos only need to approve on web browser once
  - Should be able to extend Vault to support ssh-agent in addition to Kerberos for when Kerberos is not available
- Configuration is managed by server operators, nothing for users
- All protocols are in common industry use
- JWTs are better supported and more secure than X.509 proxies
  - Can be much more purpose-specific
- Tools all open source, generally available

Links

- WLCG Authorization Working Group client tools investigation report
  - https://github.com/WLCG-AuthZ-WG/client-tools
- Bearer token discovery:
  - https://github.com/WLCG-AuthZ-WG/bearer-token-discovery
- WLCG JWT profile
  - https://github.com/WLCG-AuthZ-WG/common-jwt-profile
- Vault & plugins
  - https://www.vaultproject.io/
  - https://github.com/hashicorp/vault-plugin-auth-jwt
  - https://github.com/puppetlabs/vault-plugin-secrets-oauthapp
- htvault-config: https://github.com/fermitools/htvault-config
- htgettoken: https://github.com/fermitools/htgettoken
- htcondor with vault docs: https://htcondor-vault.readthedocs.io
- oidc-agent: https://indigo-dc.gitbook.io/oidc-agent/