htgettoken+HashiCorp Vault as a Service for Managing Grid Tokens

Dave Dykstra, dwd@fnal.gov

WLCG Operations Coordination meeting
May 2, 2024

Fermilab

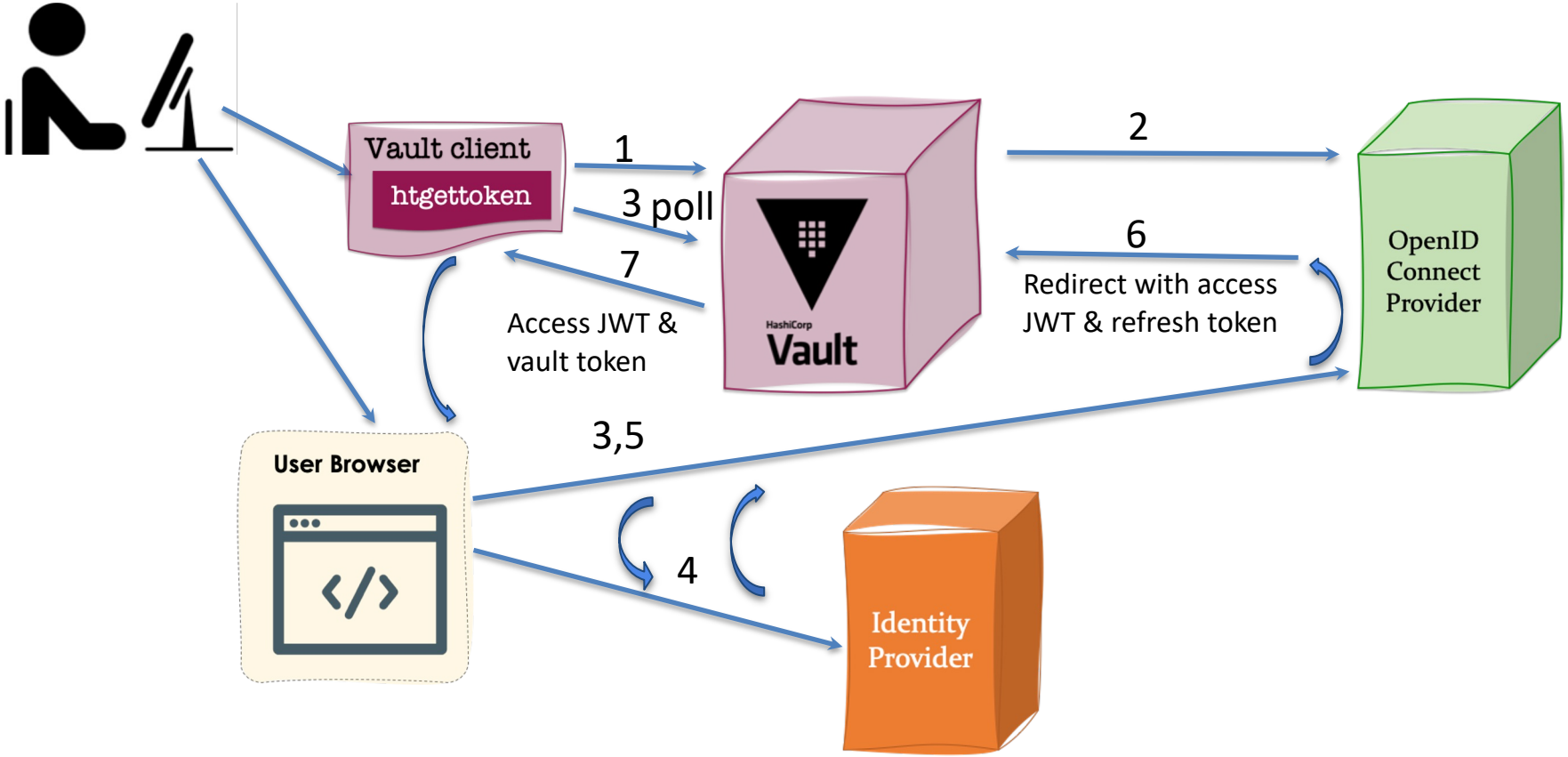
Background

- When Fermilab was first investigating transition from X.509 user proxies to tokens, the only OIDC client recommended was oidc-agent, which was insufficient for at least the analysis grid job use case
  - Stored high-value renewable refresh tokens on client machines, encrypted by passphrase
  - Required separate Oauth client credentials for every user
  - No good way to renew credentials sent with jobs
- Instead, we developed a server-based solution based on HashiCorp Vault
  - Designed for many-user grid job submission
  - Has production operational experience with ~25 Fermilab-hosted experiments and LIGO/IGWN
  - CMS is working on integrating it
- Since then, the Mytoken client & server were written
  - Similar architecture, has some nice features, but does not yet have Kerberos-based renewal or integration with job submission infrastructure for renewal of tokens in jobs

Vault with htgettoken

- HashiCorp Vault
  - Popular open general purpose secure secret store
  - Flexible plugin architecture and API; accesses mapped like a filesystem
  - Had existing OIDC, Kerberos, and ssh-agent auth plugins and Oauth secrets plugin
    - Needed some extensions, submitted as pull requests
  - Integrates well with both IAM and CILogon OIDC-based token providers
  - Manages access with its own tokens
  - We use it to store long-lived refresh tokens for many users and experiments
- htgettoken (ht from High Throughput Computing)
  - Relatively simple python command line Vault client to automate the flows
  - Initially authenticates via OIDC & a web browser, then renews Vault access via Kerberos or ssh-agent
  - Long life (~1 month, renewable) refresh token stays in Vault, limited life (~1 week) Vault tokens and even shorter life (~3 hour) access JWTs stored unencrypted in local files
  - Follows WLCG bearer token discovery standard for local filename
  - New access tokens retrieved from OIDC token issuer via Vault using Vault token
  - Includes httokensh for automated client-side renewal

htgettoken with Vault initial OIDC flow



htvault-config configuration package

- Package for configuring Vault for use with htgettoken
  - Includes a modified Hashicorp plugin, an added puppetlabs plugin, and an added ssh-agent plugin
  - Automates all the installation and setup of Vault
  - Configures OIDC issuers and roles via yaml files
  - Supports an option of using 3 servers for high availability using a builtin Vault capability
  - Available in OSG yum/dnf distribution along with vault and htgettoken
- We use a Fermilab-specific configuration generator for the yaml files so configuration is completely automated
  - Info comes from the same configuration system where we define experiments, membership, roles, and token scopes that are pushed to the token issuer

Token issuers, capability sets, and roles

- JSON Web Tokens can be tailored to minimum privilege by use of specific scopes with limited access (and also specific audiences)
- The knowledge of what scopes are allowed per user is maintained by the OIDC Provider, aka the token issuer; the user doesn't need to know
- 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:

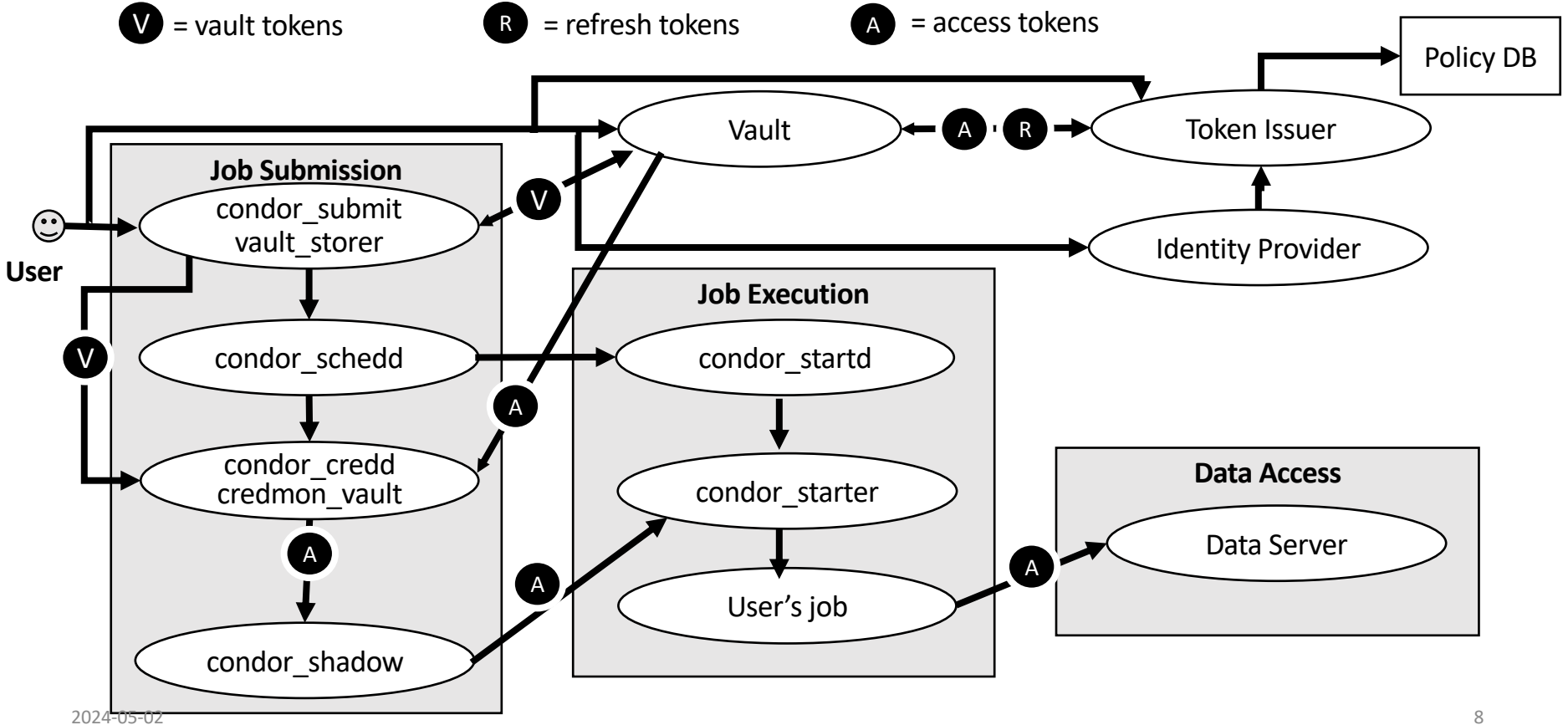
```
   htgettoken -a htvault.cern.ch –i cms –r production
=> https://cms-auth.web.cern.ch, wlcg.capabilityset:/cms/pro
```

- Since the vault configuration is generated, we could directly generate lists of scopes there instead, but capability sets enable changing the list of scopes in the token issuer without requiring new refresh tokens

HTCondor integration

- htgettoken with vault has been integrated into HTCondor
  - condor_submit can be configured to automatically invoke condor_vault_storer which invokes htgettoken as needed and stores a vault token into the condor credd
  - Submit file specifies issuer, optional role, and optionally can choose reduced audience and/or scopes
    - May obtain more than one token for a job
  - condor_credmon_vault periodically gets new access tokens which get pushed to jobs
  - Available in all current versions of HTCondor

Token flow with HTCondor and Vault



Support for “robot” (unattended) operation

- Important for tasks such as production job submission
- Vault administrator could 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 the robot’s Kerberos credentials
  - Not simplest design, but it’s a minor extension on top of interactive case

Managed Tokens Service

- Concentrates the knowledge and security for robot token operations into a single place
- Kerberos keytab credentials for all the robots stored on one machine
- Initial OIDC authentication done by administrator(s) of the service
  - The administrators need to be listed as authorized users of all the roles
- Pushes new vault tokens to credd from cron jobs by invoking condor_vault_storer
- Copies vault tokens to robot accounts on job submission machines with scp
- Configurable, written in go, and available as open source

Summary

- The transition to tokens is complicated and touches many aspects
- Using standard protocols enables reusing some existing software, even if we need to adapt it for our unusual cases
- htgettoken + htvault-config + Vault is useful as a part of the solution while making maximum use of industry-standard software
- Tools all open source*, generally available

* The HashiCorp license was updated last year to include a non-compete clause and so is no longer considered fully open source, but it is close enough for our purposes

Links

- Vault & plugins
  - https://www.vaultproject.io/
  - https://github.com/hashicorp/vault-plugin-auth-jwt
  - https://github.com/puppetlabs/vault-plugin-secrets-oauthapp
  - https://github.com/42wim/vault-plugin-auth-ssh
- htvault-config: https://github.com/fermitools/htvault-config
- htgettoken: https://github.com/fermitools/htgettoken
- managed-tokens: https://github.com/fermitools/managed-tokens
- HTCondor integration: https://htcondor.readthedocs.io/en/latest/search.html?q=vault
- WLCG JWT profile
  - https://github.com/WLCG-AuthZ-WG/common-jwt-profile
- Bearer token discovery:
  - https://github.com/WLCG-AuthZ-WG/bearer-token-discovery