HTCondor Integration with Hashicorp Vault for Oauth Credentials

Dave Dykstra, dwd@fnal.gov

HTCondor European Workshop
22 September 2021

Fermilab

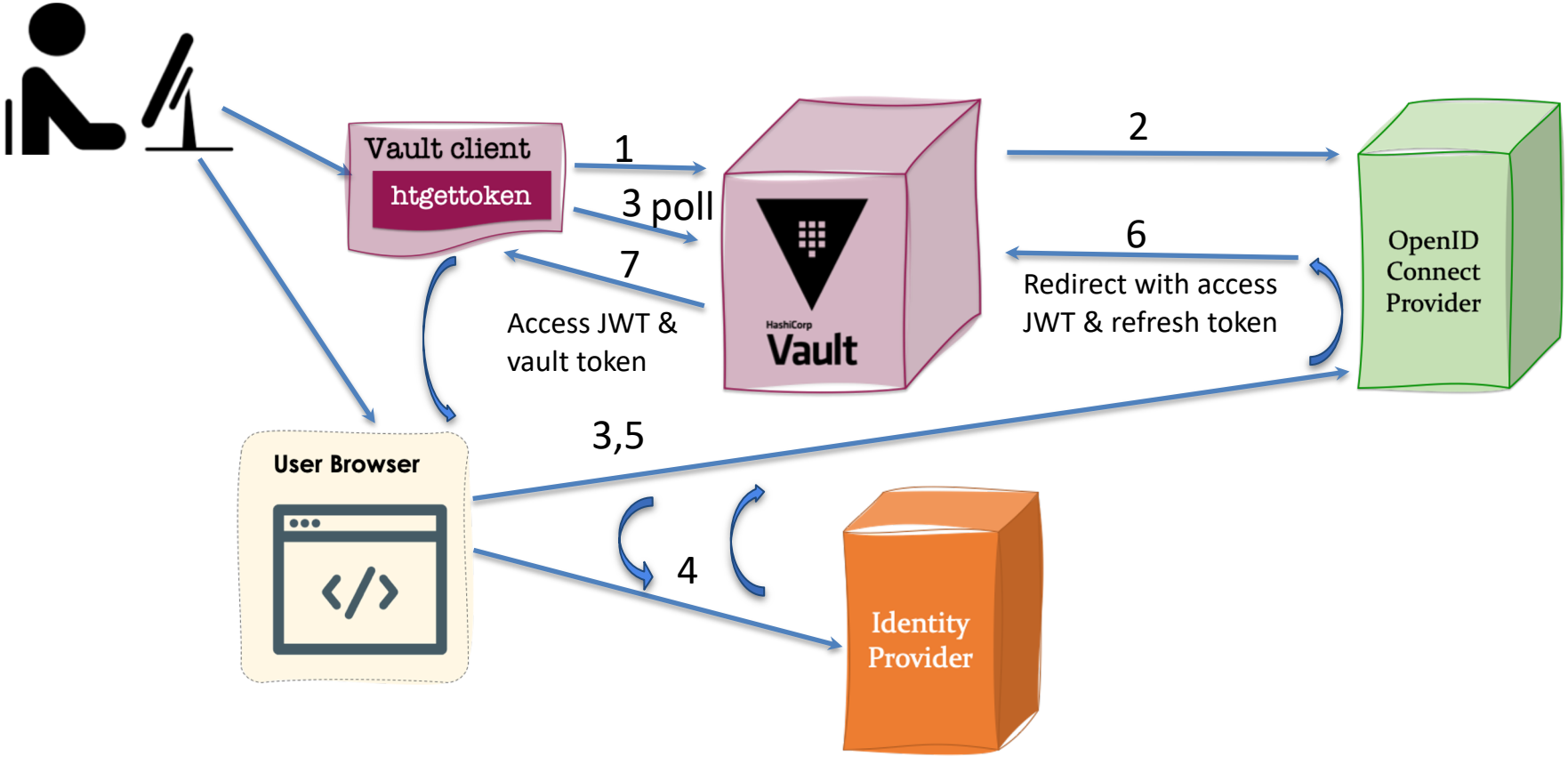
Background

- High Throughput Computing has been traditionally based on X.509 for user authentication
  - Unlike host side, this was never embraced by industry
- We're now moving to the newer industry standard Oauth2/Open ID Connect (OIDC) and JSON Web Tokens (JWTs)
- HTCondor had a couple of solutions of its own, but they each have limitations
  - The "local token issuer" solution, where HTCondor issues it own tokens without Oauth2, doesn't scale to many submission points, and only supports fixed set of scopes
  - The "Oauth2 credentials" solution, where HTCondor is an oauth2 client, requires web browser authentication before most job submissions, and doesn't help with non-condor use cases

Vault with htgettoken (independent of HTCondor)

- Hashicorp Vault
  - Popular open source general purpose secure secret store server
  - Very flexible plugin architecture and client/server REST/JSON API, and secrets are stored like in a filesystem
  - Has existing OIDC and Kerberos plugins
    - Needed some extensions, submitted as pull requests
  - Behaves as an Oauth2/OIDC client
    - Integrates well with both Indigo IAM and CILogon OIDC Providers, at least
  - Manages access with its own tokens (“vault tokens”)
  - We use it to store long-lived refresh tokens for many users
- htgettoken
  - 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 token and even shorter life (~1 hour) access JWT both 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
  - Available in OSG yum distribution along with vault and htgettoken

Capability sets, issuers, and roles

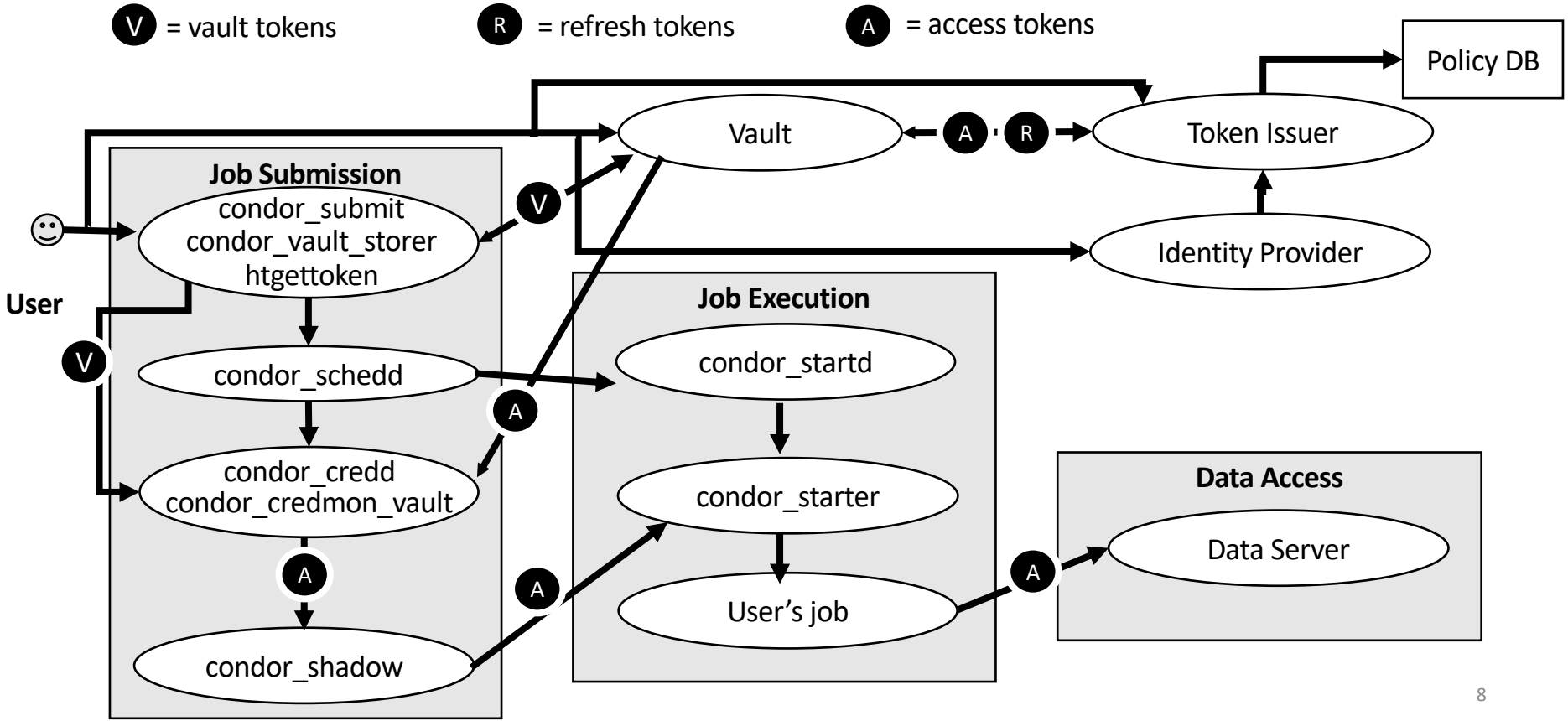
- JSON Web Tokens can be tailored to minimum privilege by use of “capability” scopes with access limits (and also specific audiences)
- The knowledge of what scopes are allowed per user is maintained by the OIDC Provider, aka the token issuer
  - Not known by clients
- We configure Vault to request scope 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 dwdvault.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 in credd
    - Vault token used by condor_credmon_vault 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
    - Based on previous implementation of Oauth2 credential support
  - Will be in HTCondor's distro in 9.0.6, 9.1.5, now in release candidate
    - Also available in all OSG builds of htcondor-9.0+

Token flow with HTCondor and Vault



HTCondor configuration

- System admin:
  - Install condor-credmon-vault rpm and set for example:
    ```
    SEC_CREDENTIAL_GETTOKEN_OPTS = -a dwdvault.cern.ch
    ```
- User submit file for example:
  ```
  use_oauth_services = cms
  cms_oauth_permissions = storage.read:/ #optional
  cms_oauth_resource = https://eos.cern.ch #optional
  ```
- Service names may include role, such as `cms_production`
- Handles may appended to store multiple variations for each service:
  ```
  cms_oauth_permissions_readonly = storage.read:/
  cms_oauth_permissions_write = storage.write:/
  ```
- All tokens end up in `$_CONDOR_CREDS`

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, very little necessary for users unless they want to “down-scope” their tokens
- 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

- 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