Workload Identity Practices

Benedikt Hofmann

Hannes Tschofenig

Edoardo Giordano

Yaroslav Rosomakho

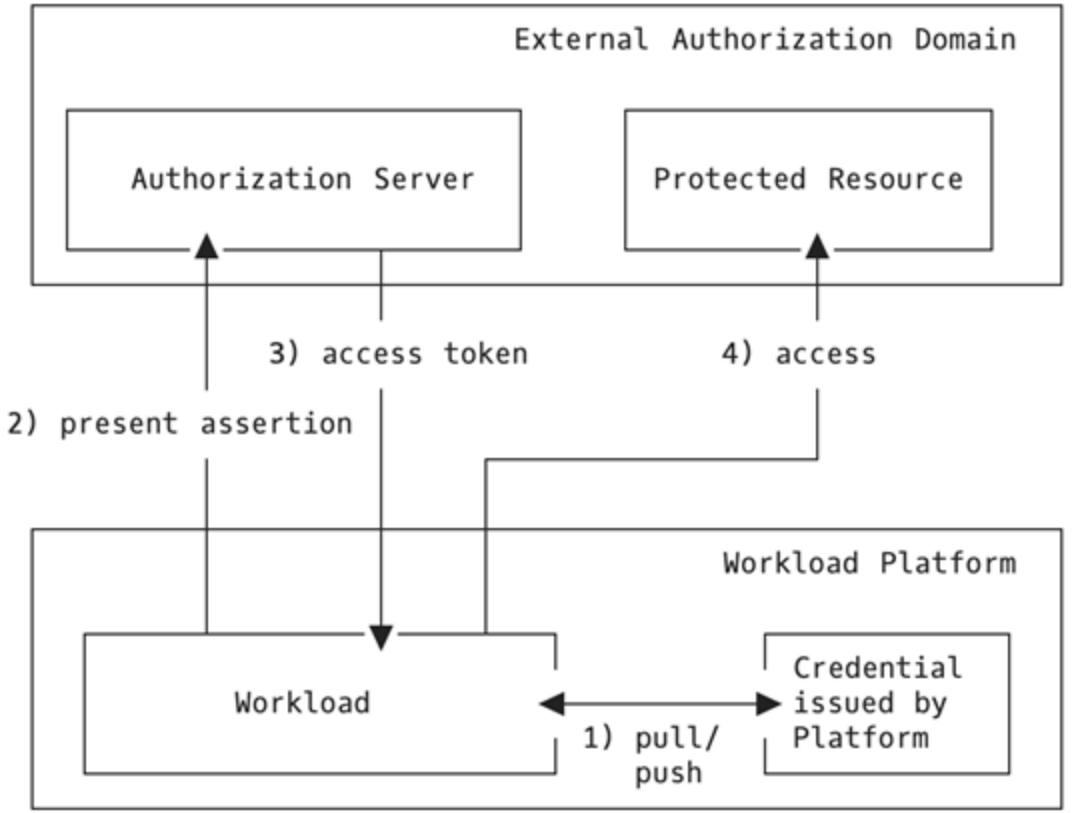
Arndt Schwenkschuster

Since IETF 121

- Rename to “Workload Identity Practices” - No “current” or “best”. Thank you Justin for your help with Datatracker.
- **Credential issuance**
  - **Environment variables**
  - **Files**
  - **Local APIs**
- Clarifications:
  - Token typing
  - Audience
  - ID tokens and general, relation to OpenID Connect

Common pattern

1) Workload receives a credential issued by the platform. (Most of the time a JWT token).
2) Workload presents this credential as an assertion to a AS in an external domain.
3) AS returns an access token.
4) Workload access the protected resource.



Common pattern -> applied across platforms

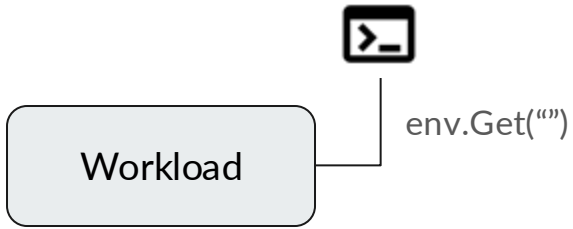
Credential issuance - Environment variables

Injecting the credentials into the environmental variables

Considerations:

- Static in nature
- Access control not trivial
- Observed by many components

Common in **build systems**, CI environments (“Github ID Token”, “Gitlab Token”, etc.).



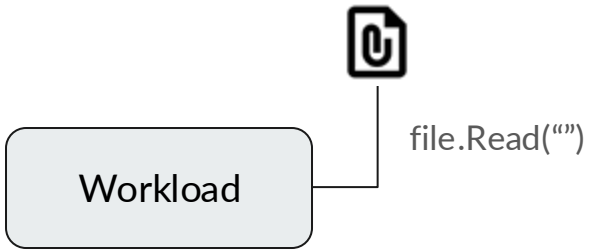
Credential issuance - Files

Files allow to inject credentials into the workload environment through the file-system and access them through primitives, e.g., Open, Close, etc.

Considerations:

- Access control incl. potential mount isolation.
- Rotation requires solution from both sides. File update + re-read.

Common in **Kubernetes** environments (Token Volume Projection), also in **local environments**. Sometimes combined with an environment variable that points to a file.



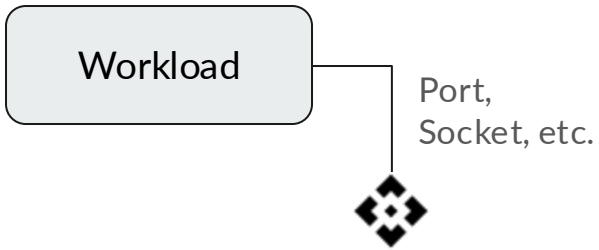
Credential issuance - APIs

Local APIs to communicate between the Host and the Workload. E.g., Unix Domain Sockets, Loopback or Magic (Link-Local) Addresses.

Considerations:

- Only issued on request.
- More structured delivery but less portability.
- Potential additional latency, operational overhead.

Common in **cloud environments** (Instance Metadata “pattern”), **SPIFFE**, also in **Kubernetes** (TokenRequest).



Guidance from the working group wanted

Should other “Workload Identity Practices” be described in the document? So far:

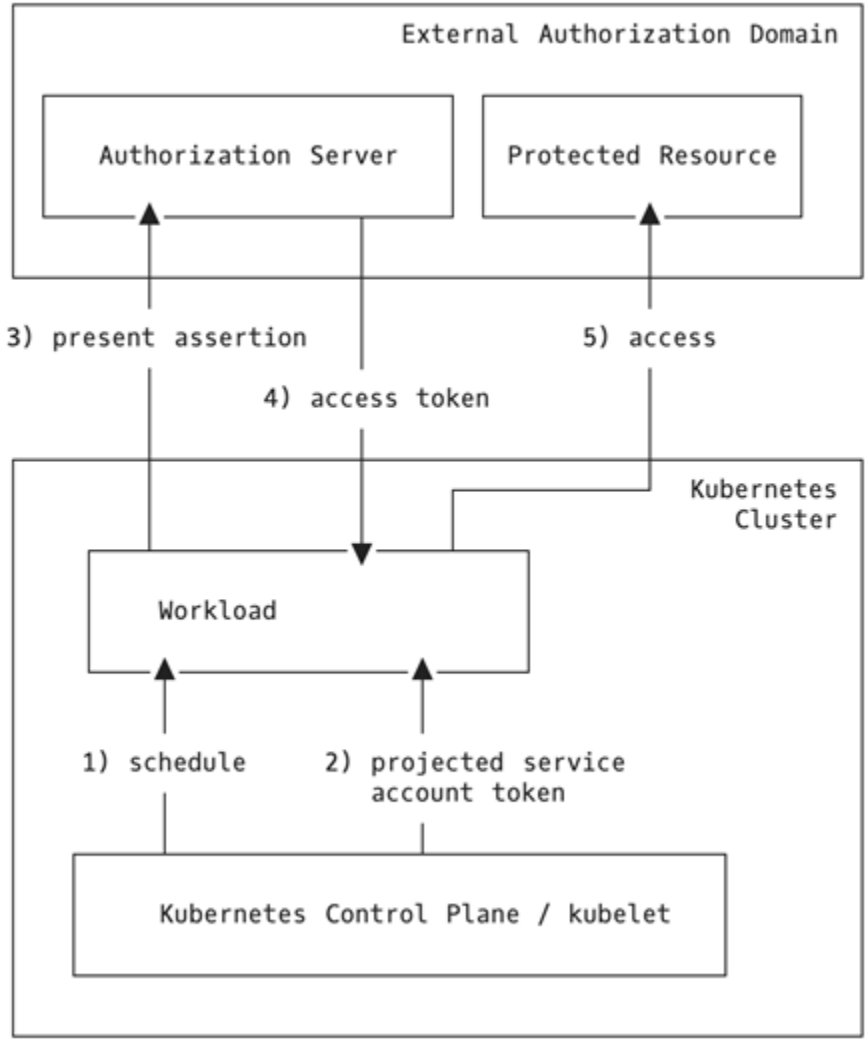
- Delivery patterns
- Use as assertions in OAuth environments
- Claims
- (Security) Considerations

Next steps? Should we finalize this work?

Appendix

Kubernetes

1) Workload (aka “Pod”) gets scheduled by the Kubernetes Control Plane and kubelet
2) Kubelet requests service account token (PSAT) on behalf of the workload from the control plane and projects it into the workloads filesystem.
3) Workload presents PSAT as an assertion to an AS in an external domain.
4) AS returns an access token.
5) Workload access the protected resource.



Cloud providers

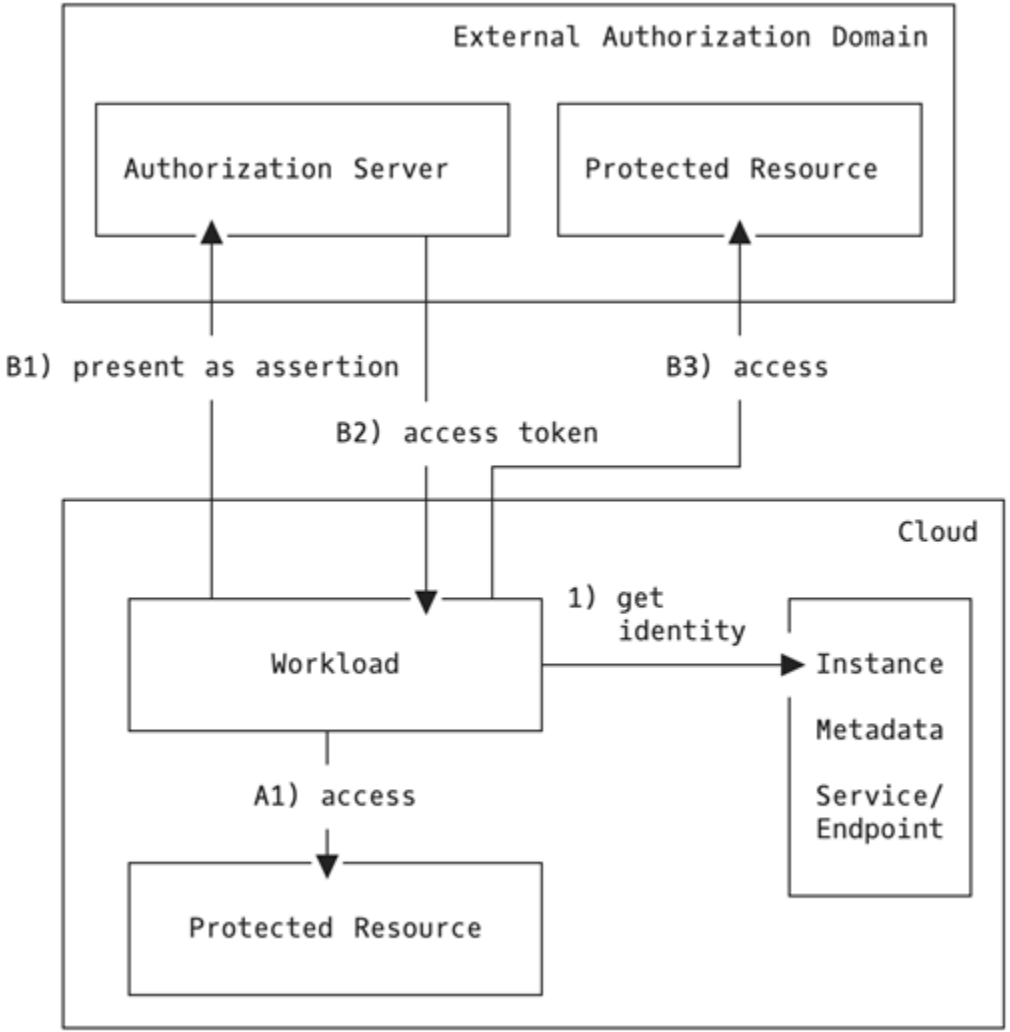
1) Workload that runs as VM, Serverless, low/no-code, etc. requests a credential from the “Instance Metadata Service”.

—- SAME DOMAIN —

A1) Workload uses the token directly to access resources in the same domain.

— OTHER DOMAIN —

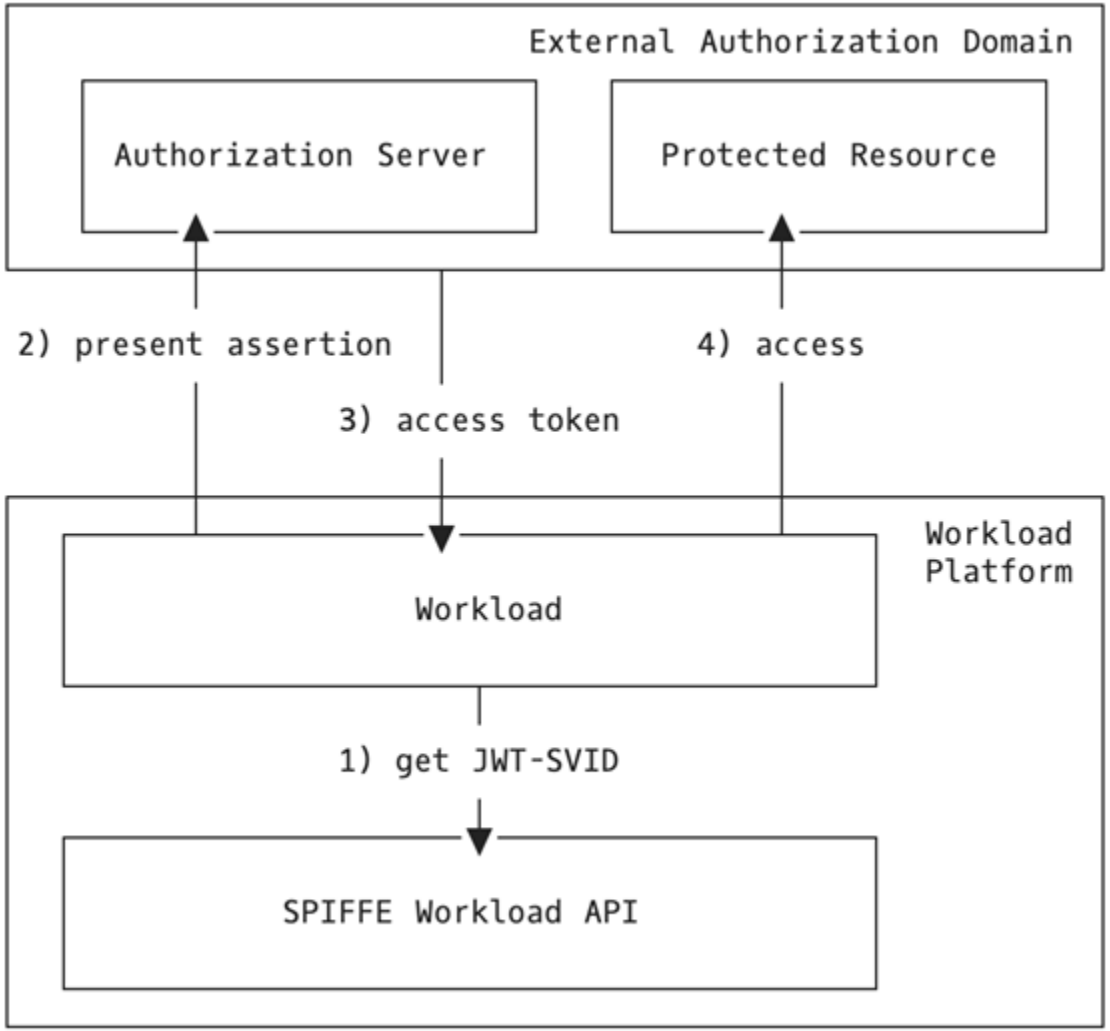
B1) Workload presents token as an assertion to external AS
B2&3) Workload gets access token and uses it.



SPIFFE

1) Workload uses the SPIFFE Workload API to get a JWT-SVID.
2) The Workload uses the JWT-SVID as an assertion to the external AS
3) External AS returns access token
4) Workload accesses protected resource.

Workload platform here can be anything that SPIFFE supports, from VM to Kubernetes, to serverless.



CI/CD

1) CI/CD platform schedules a task/job/workflow
2) Workload retrieves identity from the platform (or got provisioning with it)
3) Workload uses identity credential as an assertion towards an external AS
4) AS returns access token
5) Workload access protected resource.