Standardized Approach to Handling Secrets at Runtime

This is the approach that has been in use for several years for New Solutions (multi-tenant, kube deployed).

However, we are now proposing that this be the ideal/standard, wherever possible. For new applications where Ellucian has control, it would be required. For heritage or 3rd party it should be done unless technical feasibility prevents it. For containerized workloads, it can almost always be done by introducing an entrypoint that translates between secret lookup and internal configuration (files).

Philosophy

The guiding philosophy is that deployment pipelines should not have access to the secrets of an application being deployed (e.g., database password). Applications should instead have an IAM role with a policy that allows them to call Hashicorp Vault to retrieve secrets. This solution

- limits blast radius - only the application that uses the secrets needs a role having a policy to access those secrets
- ideally exposes the secrets only in memory or in config files of the running app, without passing them through deployment code where we'd have to ensure they can't be logged, intercepted, etc

Deployment pipelines will pass the key(s) of secrets the application needs to the app, ideally via environment variables (12 factor apps).

Tooling

We are stating that we should converge on using Vault for secrets storage, moving away from SSM parameter store and AWS Secrets Manager.

Implementations

Net New apps

Net new apps should use SDKs/APIs directly to retrieve secrets from Vault as needed. The location of Vault as well as key(s) to secrets should be passed as environment variables (12 factor apps)

Existing Ellucian Apps

Ideally, should be retrofitted to use SDKs/APIs. However, for containerized apps, it is permissible to use an entrypoint script, at least temporarily, vs updating the application to access Vault directly. For example, a containerized application that runs on tomcat and reads a configuration file that contains a db password, can, instead of changing the app, utilize a script as the entrypoint. The script would intercept the env vars for the Vault key that holds the db password, retrieve the secret, and update the (in-container) config file prior to starting tomcat. The same technique can be used on EC2 based workloads as well.

Third party apps

Third party apps should attempt to use the entrypoint scripting technique to retrieve secrets and write them into local config files/env variables prior to startup wherever possible, particularly for containerized third party apps.