SODALITE: Solid as a Rock

Improvements across the full stack for a more secure solution

Source: https://www.freepik.com/free-vector/abstract-secure-technology-background_5939475.htm#page=5&query=cyber+security&position=29

With the growing importance of DevOps roles in technological companies, bringing closer together development teams and operation teams, deployments are more frequent. As a result, possible security holes appear more often. The need to improve the "security at speed" in the context of these workflows created the need of DevSecOps, i.e., best practices to empower the security of the code. SODALITE is all about DevOps, managing remote resources with key users making data privacy a priority. The management of remote resources incorporates security, using well established and robust technology. Let us tell you how we do it!

The SODALITE stack provides tools and methods to authenticate and authorize actions on API endpoints using open-source Identity management and Secure Secret handling tools. Why is authorization required? A single SODALITE endpoint can manage multiple infrastructures belonging to different domains. In addition to proper authentication and authorization of user actions, safe secret management across the whole deployment pipeline is also required and ensured by SODALITE.

From the user's perspective, any orchestration or automation tool that uses any kind of credentials, access tokens or keys to access and configure resources is considered a risk. The risk of exposing these credentials can be an overwhelming factor for not using a specific orchestration or automation tool. SODALITE addresses the complex issues of creating a secure workflow in an orchestrated automation context by using industry standard open source tools to handle the following three security aspects of complex orchestration: user authentication, user authorization and secret management. The first two points are usually covered by Identity and Access Management (IAM) tools that connect the user identity with system access credentials (authentication) and user rights regarding the usage of specific system resources (authorization). To enable a seamless integration the OAuth 2.0[1] authorization framework was chosen, which is the de-facto industry standard for authorization.

SODALITE uses *Keycloak*[2] as the IAM provider, a popular and widely used open source tool which simplifies the creation of secure services with minimal coding overhead for authentication and authorization. Keycloak allows a wide customization of options regarding authentication schemas such as two-factor authentication and seamless integration with third-party identity providers like Google or GitHub, making it an ideal tool for extending the reach of the project providing different levels of access to different SODALITE components and different users.

A JSON Web Token (JWT), once issued by Keycloak, is then used across the whole SODALITE workflow. Keycloak allows the creation of various clients for different components, e.g., one set of rules can be applied to users logins from the SODALITE IDE and others for automation components like SODALITE's Deployment Refactorer. Each client has a client secret assigned that is provided to Keycloak upon token creation to validate Token Endpoint API calls. Once a JWT Token is issued, it can be validated by Keycloak using two standard mechanisms: the introspection endpoint and JSON Web Key Sets. Introspection mechanism provides a more secure way for validating tokens as the token can be revoked before expiration and should be used as default.

In SODALITE, multiple users may cooperate on the same application deployment project for modelling specific Resource Models (RMs) and Abstract Application Deployment Models (AADMs) in the IDE. These models can be private or public, and are usually referred to as different Project Domains, meaning that only a certain group of end users should be granted read or write access to these resources. To ensure this access granularity, client roles and groups are used in Keycloak. Each role provides read or write access to a certain project domain. One user can belong to many Project Domains and thus have any number of client roles assigned, for an effective user and group management without any other separate component needed.

In addition to properly authorising user actions, the third security aspect, Secret Management has been introduced in SODALITE for proper handling of infrastructure secrets such as RSA keys,

[1] https://oauth.net/2/ IETF industry-standard protocol for authorization. OAuth 2.0 focuses on client developer simplicity while providing specific authorization flows for web applications, desktop applications, mobile phones, and living room devices.

[2] https://www.keycloak.org/ Open Source Identity and Access Management for Modern Applications and Services

tokens, logins and passwords. Secret Management has in itself two aspects of handling secret data: security of data in use and security of data at rest. First one is addressed by properly handling the secrets across the whole pipeline: not storing unencrypted information, no logging for security critical parts, and proper user management on virtual containers that host SODALITE components. While SODALITE allows not storing any secrets at all and providing them in inputs, storing secrets in a vault allows to automate workflow and additionally ensure its safety. For that purpose, *Hashicorp Vault*[3] has been chosen, which is a widely used open source tool for secret management. This approach allows SODALITE operators to integrate with their own Vault installations that might not be a part of SODALITE component stack.

[3] https://www.vaultproject.io/ Secure, store and tightly control access to tokens, passwords, certificates, encryption keys for protecting secrets and other sensitive data using a UI, CLI, or HTTP API.

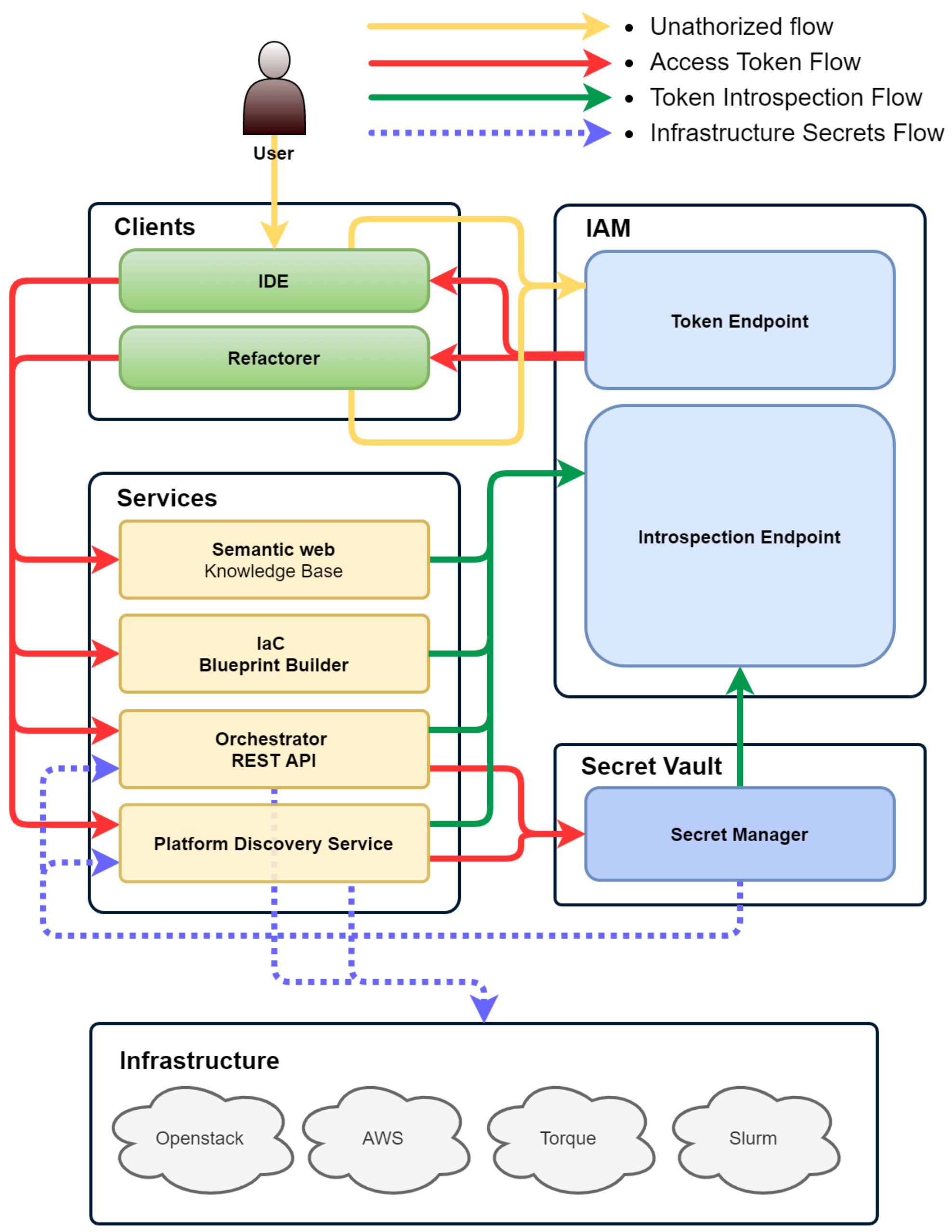

Figure. Authorization and and Secret management flow in SODALITE

Hashicorp Vault configuration ensures secret management for Project Domains by setting separate secret storages. When accessing a secret, client components must provide a client role and a valid access token issued by Keycloak. This access token is validated by Vault and a short lived token is returned by Vault. Using this token and a specific secret address (namespace) enables the retrieval of a secret. The figure above shows the flow for managing Authorization and Secret management in the SODALITE stack.

By default both Keycloak and Hashicorp Vault are deployed as part of the SODALITE stack. Configuration of the components is done on the fly making these two components ready to use after deployment of the SODALITE blueprint. Admin credentials, roles, groups, clients, policies are created automatically, while the additional configuration can be done via API calls or component Web UIs.

In the future several improvements will be addressed regarding the overall usage of secrets and security of the credentials during the execution of the blueprint and application deployment process. Additionally several options are being considered regarding the possibility of using encrypted artifacts at rest before deploying them through the TOSCA blueprint as well as have a more secure environment during the run of Ansible playbooks.

Abstract Application Deployment Models (AADMs) - a connected graph of application components that declares the relationships among components and the requirements to the infrastructure resources they need.

Identity and Access Management (IAM) - Framework of technologies and policies to ensure the appropriate access to resources by the appropriate people.

JSON Web Key Set (JWKS) - a set of keys that contain the public keys used to verify any JWT issued by the authorization server and signed by the RS256 algorithm.

JSON Web Token (JWT) - an Internet standard to produce data with optional signature and encryption with payload holding JSON asserting claims.

SODALITE IDE - SODALITE component that incorporates the usage of a dedicated DSL, provides a DSL editor and consults a semantic knowledge base for validation/recommendations

SODALITE's Deployment Refactorer - SODALITE component using the static models and dynamic monitoring data to assess the current state of infrastructure and application.