SiteMinder's Support for Where Developers…….Develop

According to a superficial Google search, the word DevOps was first coined in 2009. Since then its practice has evolved well beyond the simple merger of terminology. Being able to practice DevOps at a high level of maturity is one path that enables an organization to efficiently take advantage of market opportunities (or to service citizens in the case of a public service organization) through software. Which organization wouldn't want that??

Public sources of information tracking and documenting the advancement of DevOps are plentiful. One source I like to reference is the annual State of DevOps Report published by Puppet (www.puppet.com). These reports have been produced over the last decade with each one highlighting aspects of advancement in the state of the art notable for the specific year. They also emphasize longer duration themes covered in multiple reports as the topic evolves. For a number of issues the reports have touched on the recurring theme that security is critical for the successful practice of DevOps at any level of maturity.

Prior to the concept of a well-integrated DevOps process, IAM Security was squarely situated in the operations side of the IT house. Application development teams would follow the sequential pattern of plan, code, build, test, release, deploy, operate and monitor. Some of these stages were managed together, but some were executed in separate silos. IAM Security typically fell into the tail-end deploy and operate stages. By today's standards, that is clearly suboptimal. IAM Security should be part of each stage. How can an organization design an application if there isn't a plan for what types of authentication unlock which security-sensitive features within the application or what sort of user attributes are needed to make appropriate authorization decisions? Effectively planning those application characteristics require knowing how to leverage these authentication and authorization functions from your existing IAM Security solution rather than building them independently, from the ground up, directly into the application. That leads to the two most recent reports (2020, 2021) from Puppet. The reports point out that organizations with higher levels of DevOps maturity have a higher probability of leveraging a common "platform" by more of their developer community. The concept of platform encompasses ***all*** elements, not unique to the application, which are necessary to the building and delivery of the application. That includes the IAM security component providing the critical path service ensuring only the right entities (user, thing, and software) access the right information under the right conditions. The key conclusion is there is little value in having application development teams spend time researching and choosing IAM security if there is a "platform" they can easily take advantage of. "Easily take advantage of" being the important part of that statement. If it is not easy, developers will avoid it.

SiteMinder has a long, successful history of supporting developers. Release 12.7 continued that history by introducing a Swagger-based REST interface for managing policies in the SiteMinder platform. That

was a good step in a modern DevOps direction. The REST interface supports creating new access policies or new authentication policies, and can do so for standard HTTP GET/POST patterns of access as well as SAML and OIDC. The newest enhancement added in the 12.8.05 release improves this capability for developers. In that release SiteMinder added support for Swagger Codegen (https://swagger.io/tools/swagger-codegen), further enabling developers in the language they prefer to develop in.

Codegen accelerates and simplifies the process of using SiteMinder's REST interface by

- Creating server stubs (set and get) and client APIs that are defined with the Swagger specification.

- Creating communication logic to invoke REST APIs endpoints

Codegen supports almost any programming language your developers may use.

Which languages you ask?

The list as of today noted in the Git repository (https://github.com/swagger-api/swagger-Codegen) includes:

- **API clients**: **ActionScript**, **Ada**, **Apex**, **Bash**, **C#** (.net 2.0, 3.5 or later), **C++** (cpprest, Qt5, Tizen), **Clojure**, **Dart**, **Elixir**, **Elm**, **Eiffel**, **Erlang**, **Go**, **Groovy**, **Haskell** (http-client, Servant), **Java** (Jersey1.x, Jersey2.x, OkHttp, Retrofit1.x, Retrofit2.x, Feign, RestTemplate, RESTEasy, Vertx, Google API Client Library for Java, Rest-assured), **Kotlin**, **Lua**, **Node.js** (ES5, ES6, AngularJS with Google Closure Compiler annotations) **Objective-C**, **Perl**, **PHP**, **PowerShell**, **Python**, **R**, **Ruby**, **Rust** (rust, rust-server), **Scala** (akka, http4s, swagger-async-httpclient), **Swift** (2.x, 3.x, 4.x, 5.x), **Typescript** (Angular1.x, Angular2.x, Fetch, jQuery, Node)
- **Server stubs**: **Ada**, **C#** (ASP.NET Core, NancyFx), **C++** (Pistache, Restbed), **Erlang**, **Go**, **Haskell** (Servant), **Java** (MSF4J, Spring, Undertow, JAX-RS: CDI, CXF, Inflector, RestEasy, Play Framework, PKMST), **Kotlin**, **PHP** (Lumen, Slim, Silex, Symfony, Zend Expressive), **Python** (Flask), **NodeJS**, **Ruby** (Sinatra, Rails5), **Rust** (rust-server), **Scala** (Finch, Lagom, Scalatra)

This capability enables SiteMinder to be more easily used by developers…where they develop….another good step supporting an organization's maturing DevOps practices.

Take 4 minutes to view a brief video covering SiteMinder's support of Codegen with a short demonstration:

https://techdocs.broadcom.com/us/en/symantec-security-software/identity-security/siteminder/12-8/programming/policy-object-rest-apis/Support-for-Swagger-Codegen-in-SiteMinder-REST-APIs.html