DCP Cloud Native Integration: Engineering Report

Daniel Schlatter, Sohil Vaidya, Thai Tran and Zack Kromer

What is Secure Data Protection Suite(SecurDPS)?

One of comforte’s key focus areas is enterprise wide data-centric protection. There are various mechanisms that can be used for data-centric protection, with one of the most prevalent ones nowadays being tokenization, more specifically, security tokenization, aka. low value tokenization. comforte today offers a data-centric protection suite, SecurDPS, which includes tokenization as one of the key techniques to protect sensitive data.

Our Objective

The general goal of comforte is not just to provide the protection technology, but also to make it as easy as possible for customers to integrate SecurDPS in various applications and services. Given the continuously growing adoption of cloud(-native) frameworks, technologies and new architecture paradigms like "serverless" or "Function-as-a-Service" (FaaS), the overarching goal of this project is to make integration of SecurDPS into cloud(-native) environments easy. The integration of SecurDPS in FaaS enables us to extrapolate the benefits of the as-a-Service platform including but not limited to reduction in cost for server management, pay for what you consumed, event driven function invocation in almost any languages, and much more.

To achieve this overarching goal the key objective of this project is to design and build the capabilities to allow consumption of protection service provided by SecurDPS in a serverless/FaaS way, and in support of some key serverless/FaaS frameworks. We successfully built these capabilities with 4 frameworks, OpenFaaS, Knative, Kubeless, and Azure Functions. Our implementation allows users to easily access these tokenization services from anywhere, for example, one could send a cURL request to a server set up with our project, and the data they sent would come back tokenized. This broadens access to these tokenization services significantly, and removes much of the setup overhead for comfortes clients.

Testing

To facilitate testing of the frameworks, cURL requests were sent via Postman to a Python server, which is on the same machine as the functions, this Python server then redirected the requests to the functions, and takes the output from the functions and sends it to the original sender. The primary issues brought up by the first round of testing were that when returning an error, the functions did not set the proper HTTP code for the respective error. Furthermore the functions often timed out when receiving empty strings as input. Proper error codes were implemented for all of the frameworks except Kubeless, which does not seem to support returning HTTP codes other than 200 in the Java runtime. Error handling for empty strings was also added as well. The feedback from the second round of testing was positive, however noting that the handling of empty strings could be more consistent.

Conclusion

Given comfote’s SecurDPS we were able to work to integrate into 4 different frameworks, OpenFaaS, Knative, Kubeless, and Azure Functions. Throughout these different FaaS frameworks we were able run SecurDPS as a function to take in JSON and CSV inputs so that we could tokenize those inputs and return a tokenized output. This was all done with ease using cURL requests sent via Postman to a Python server.