**API SECURITY:**

**Objective:**

* **How to secure APIs exposed through Apigee Edge.?**
* APIs are not only the connective tissue between applications, systems, and data, but also the mechanisms that allow developers to leverage and reuse these digital assets for new purposes.

**Problem statement:**

* APIs factor into almost every digital use case, and their role in security news isn’t an intrinsic flaw in APIs any more than vaults are categorically flawed simply because some of them have been cracked.
* But the headlines nevertheless reinforce an important message: if API security isn’t at the top of an enterprise’s 2019 priorities, that list of priorities is incomplete.

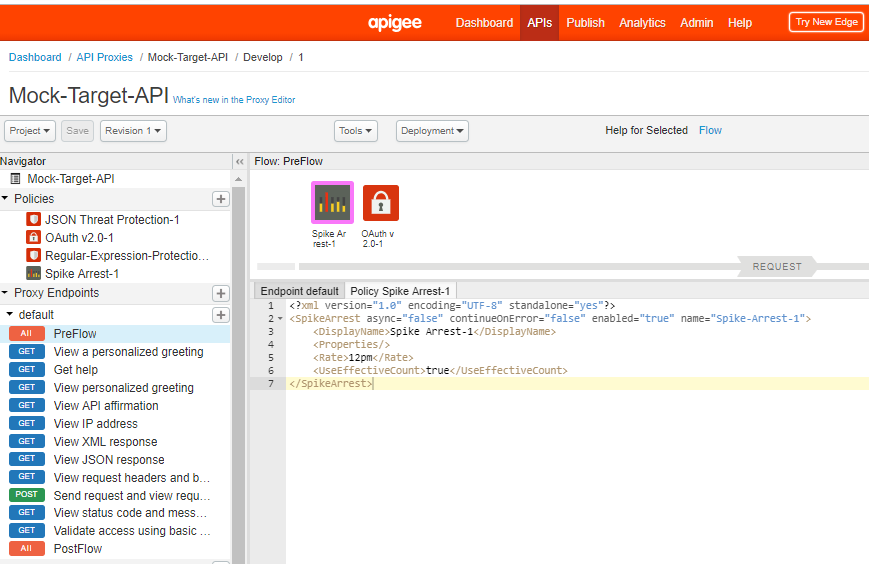
**Introduction to API Security:**

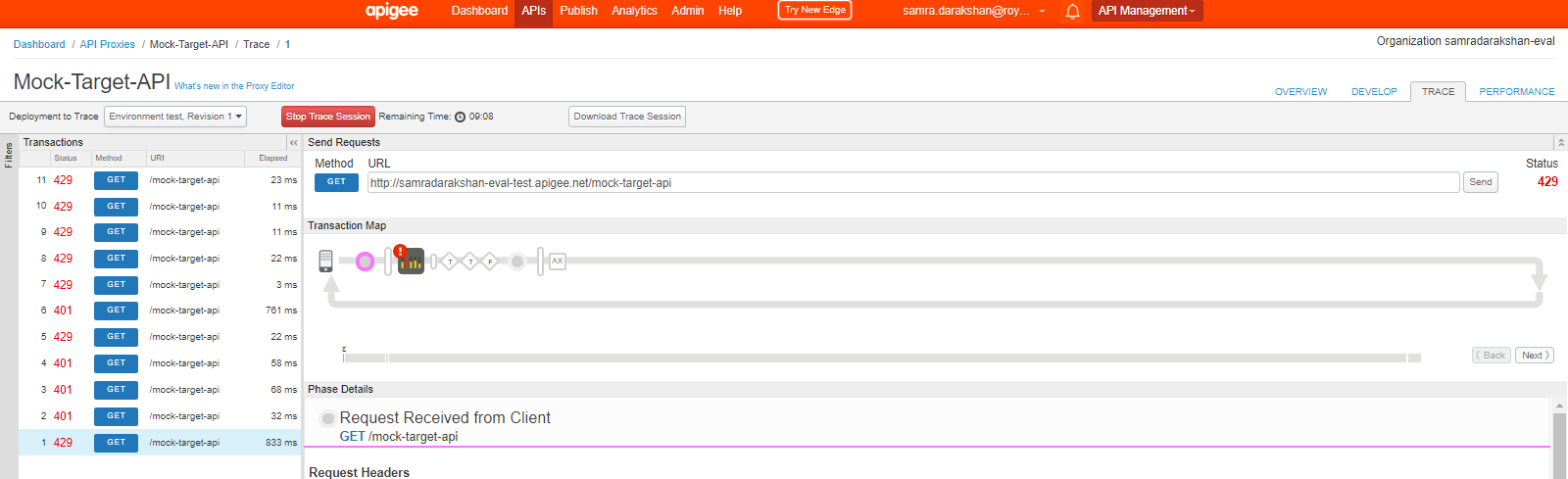
* **Throttle your API Traffic to prevent DoS**

# **Use case**

You have an existing Apigee API proxy that takes requests from the Internet and forwards them to an existing service. You have a requirement to protect your target servers (backend) from traffic spikes. You would like to protect your APIs from denial of service attacks that might lead to performance lags or downtime of your backend.

**Implementing spike arrest:**





**Spike Arrest working:**

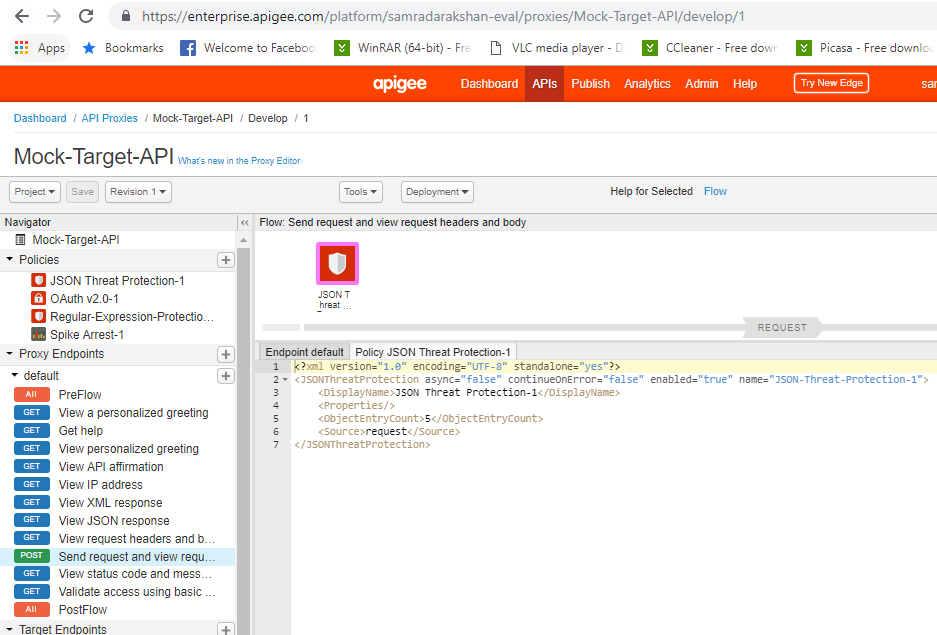
* **Per-minute** rates get smoothed into full requests allowed in intervals of **seconds**. For example, 6pm gets smoothed like this: 60 seconds (1 minute) / 6pm = 10-second intervals, or 1 request allowed every 10 seconds. A second request inside of 10 seconds will fail. Also, a 7th request within a minute will fail.
* **Per-second** rates get smoothed into full requests allowed in intervals of **milliseconds**. For example, 10ps gets smoothed like this: 1000 milliseconds (1 second) / 10ps = 100-millisecond intervals, or 1 request allowed every 100 milliseconds. A second request inside of 100ms will fail. Also, an 11th request within a second will fail.
* **Protect your APIs against threats**

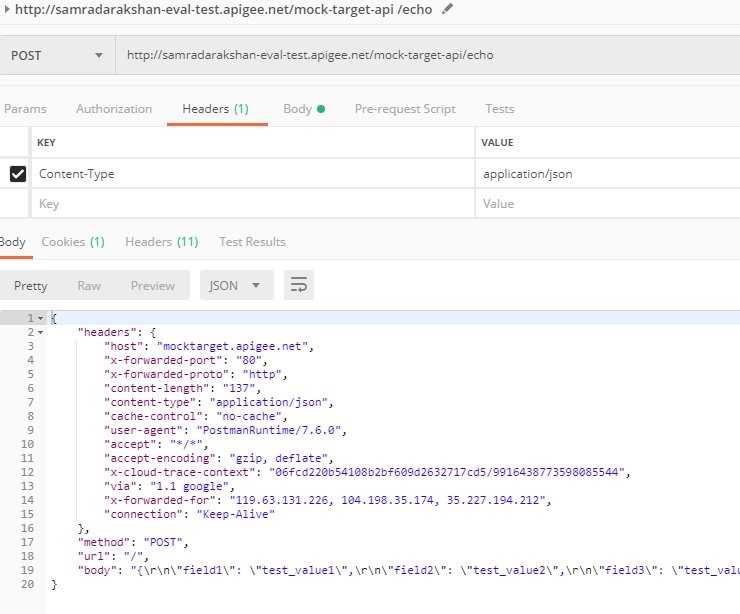
# Use case

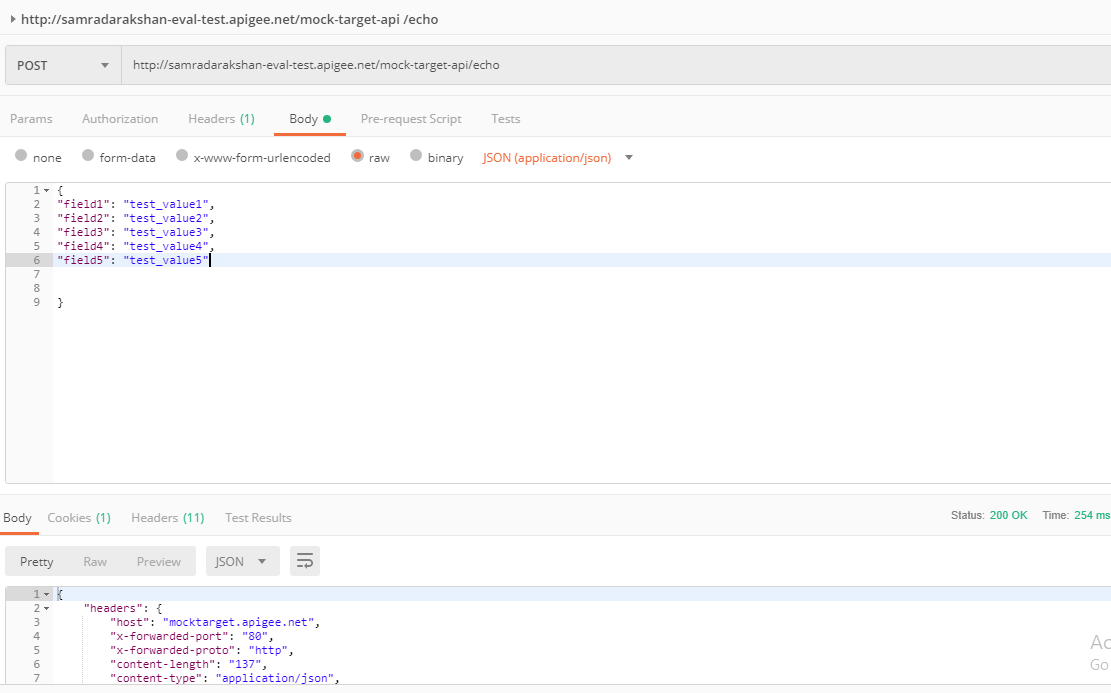
You have an existing Apigee API proxy that takes requests from the Internet and forwards them to an existing service. You have a requirement to ensure the integrity of the API message content, by protecting against threats such as JSON/XML/SQL injection and other malicious payload manipulation.

**JSON Threat Protection policy:**

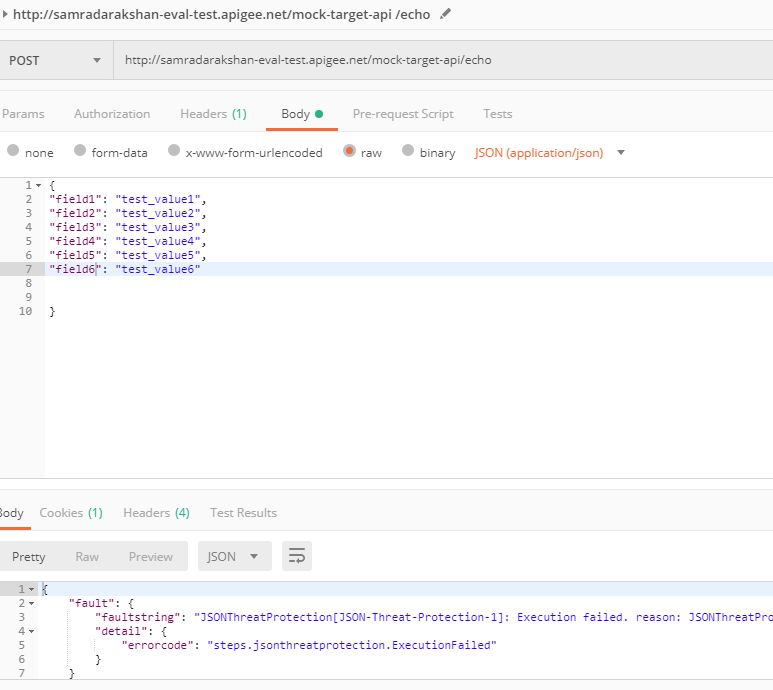
we use the JSON Threat Protection policy to ensure that the incoming API request JSON payload does not contain more than 5 fields. If the incoming payload contains more than 5 fields, the API proxy returns an error response.

* 



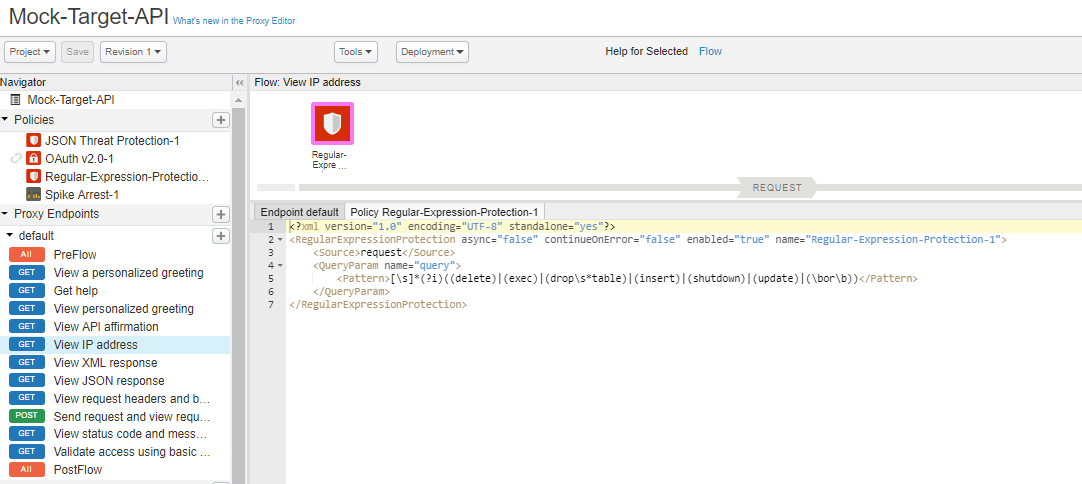


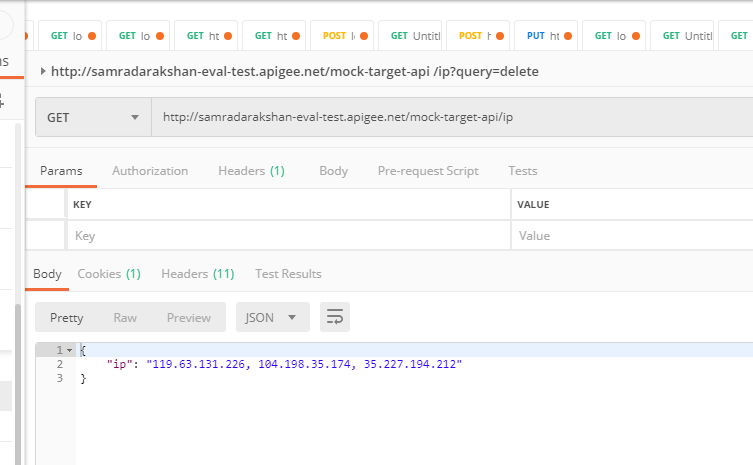
**Error occurs on 6th field:**



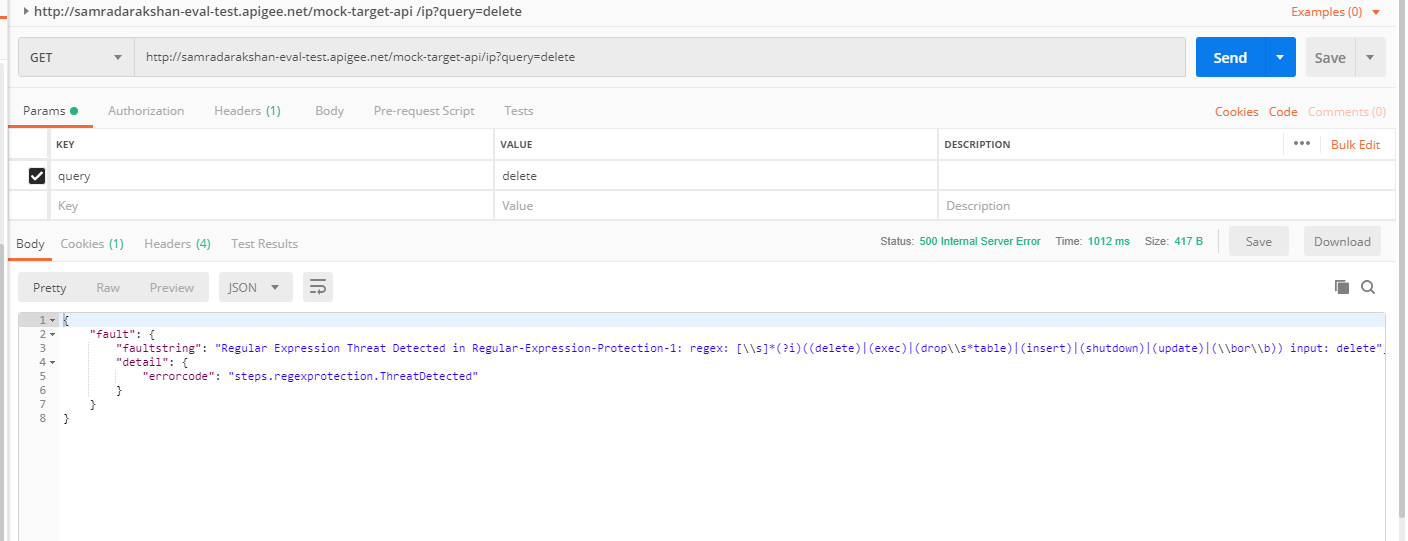
**Regular Expression Protection policy:**

Regular Expression Protection policy has been configured with a pattern that matches common SQL injection attacks. This pattern will be checked against the value of the query parameter named query, and if there is a match, the policy will return an error response.





**Error occours on query params:**



* OAuth 2.0 based API security
* **OAUTH 2 is a standard/ protocol /framework which is used to communicate between 2 or more process with which you authorize your system and decide whether you have authority or not you just used the toke, to give permissions about your application, you just get token ad used it to further**
* JWT based API security

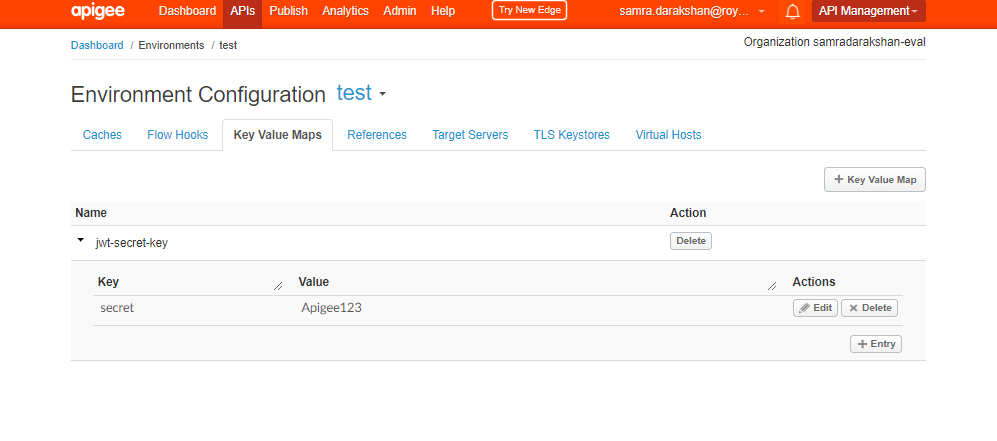
# **Use case**

You have an API that is consumed by third parties. You want to secure that API using JWT.

**JWT: just tokenization format with which you can secure your mgs, it is not that secure anyone can see the encoded msg.**

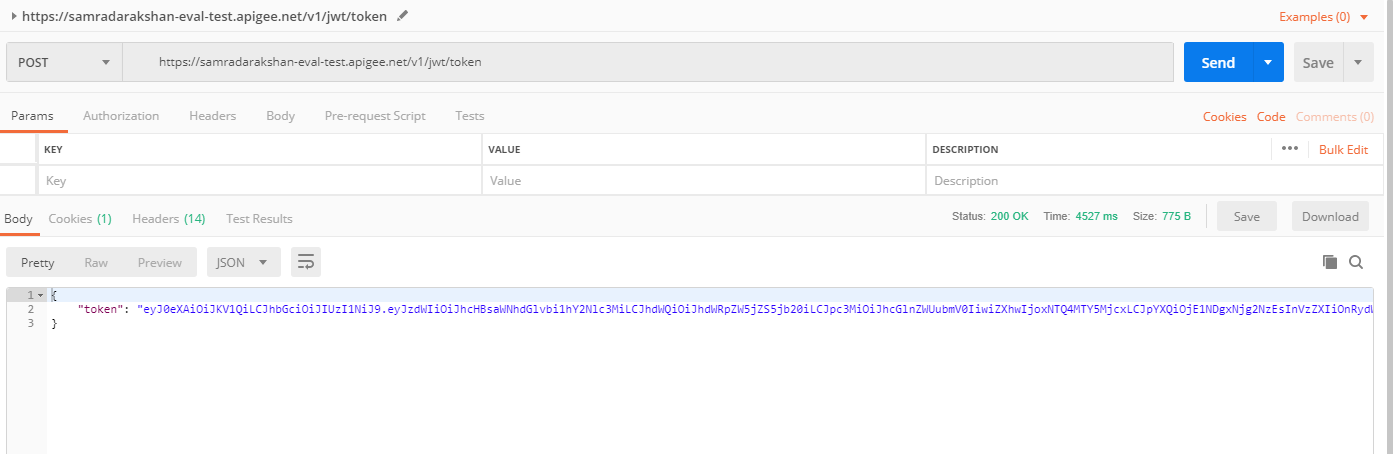
**it is not completely secure cox when you pass the mgs in header anyone can see on internet, it’s just a way you can shift the mgs client to server.**

Add JWT KVM:



This shared secret will be used to both generate and verify JWT tokens using the HS256 algorithm.

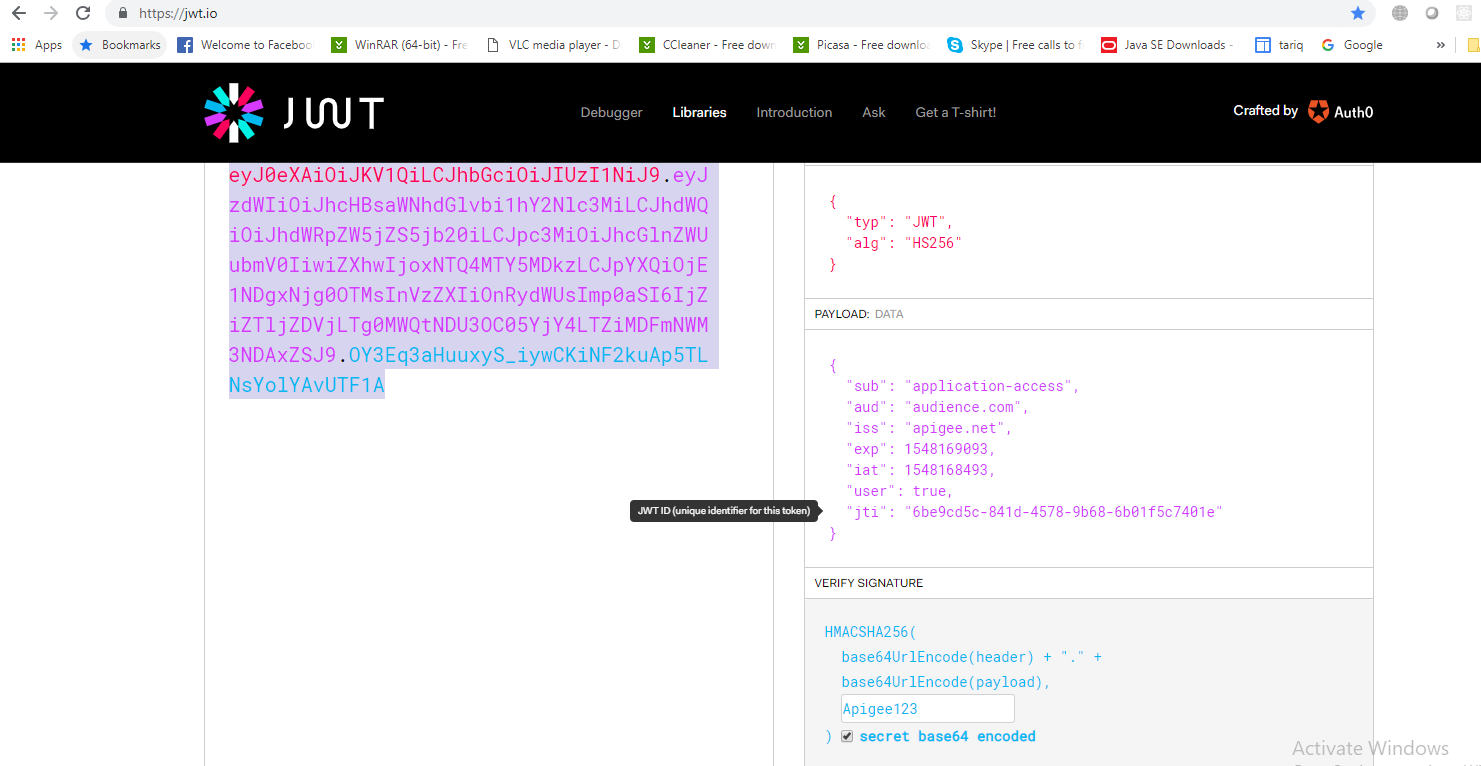
Generate token:

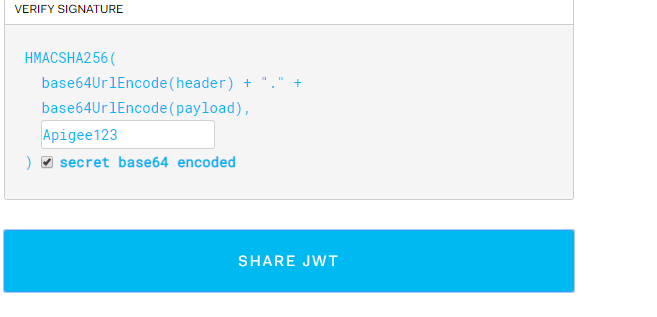


https://jwt.io/

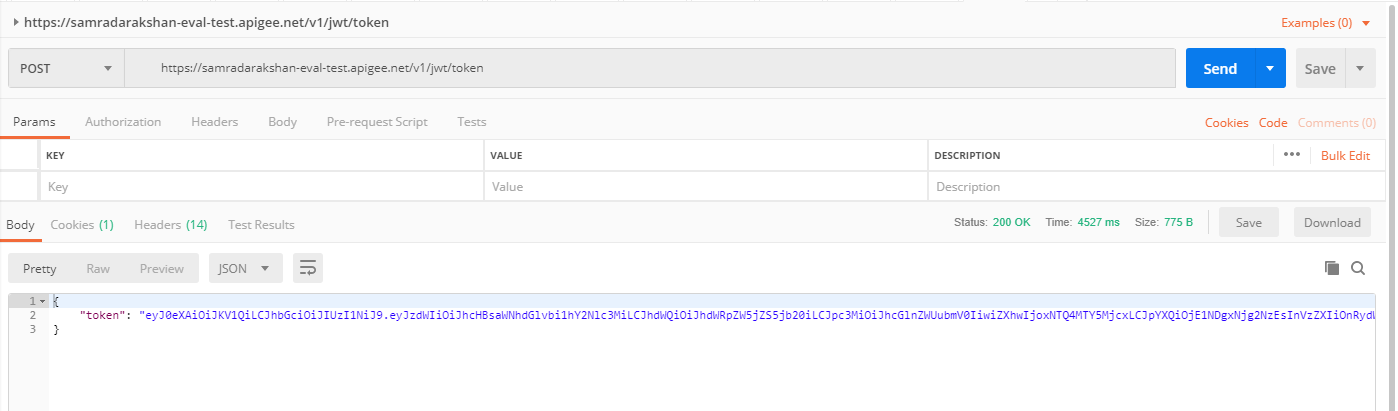
Give this token here:

And give the kvm value that is Apigee123





Paste that jwt token and response is 200 means it is verified.



* **Use Apigee with Okta Identity Protection and Oauth**

# **Use case**

You have an API that is consumed by a client application. You want to secure that API using OAuth 2.0 and use an external identity provider such as Okta, to protect the application end user identity. In this lab, we will use Apigee as the OAuth provider to protect the API endpoints using OAuth 2.0. Okta will be used to protect the application end user's identity. We will accomplish this by integrating Okta into the Apigee OAuth proxy, and implement OAuth 2.0 in resource owner / password grant type.

Make account on okta:

Use the following parameters if using the REST Client:

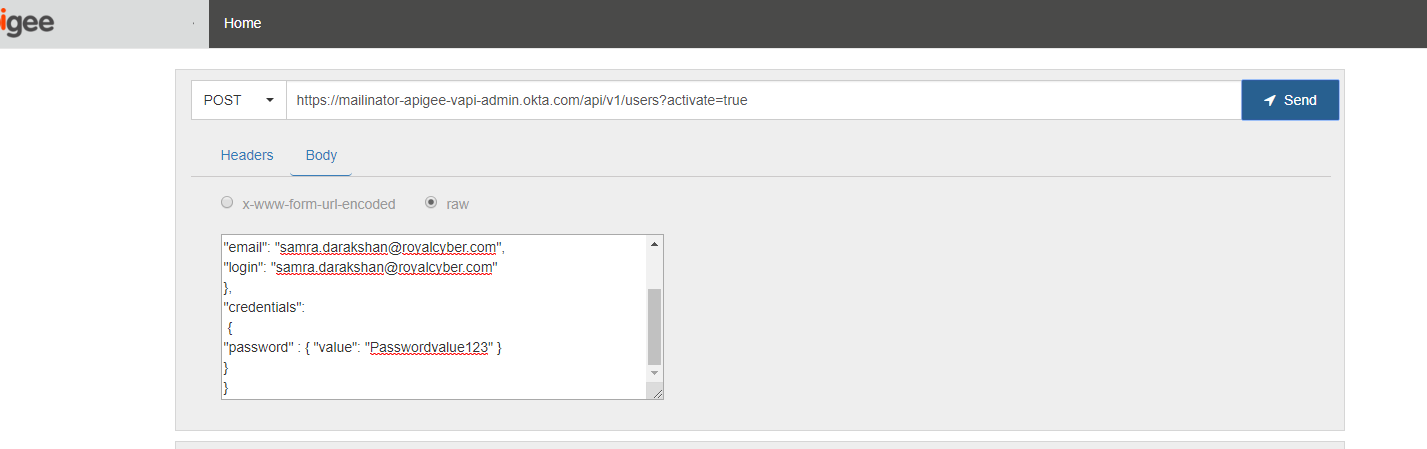
https://apigee-rest-client.appspot.com/

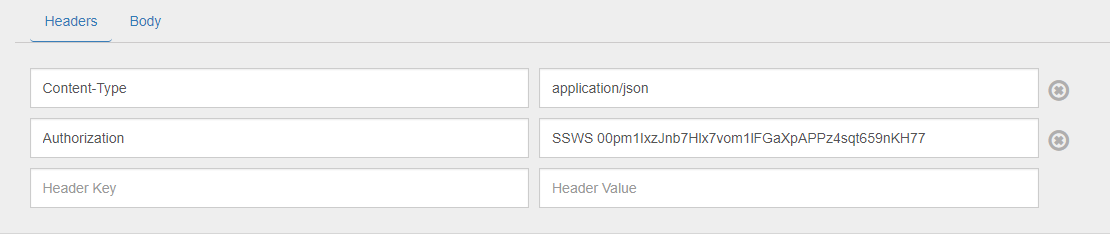
POST to URL: https://google-vapijam-security-admin.okta.com/api/v1/users?activate=true

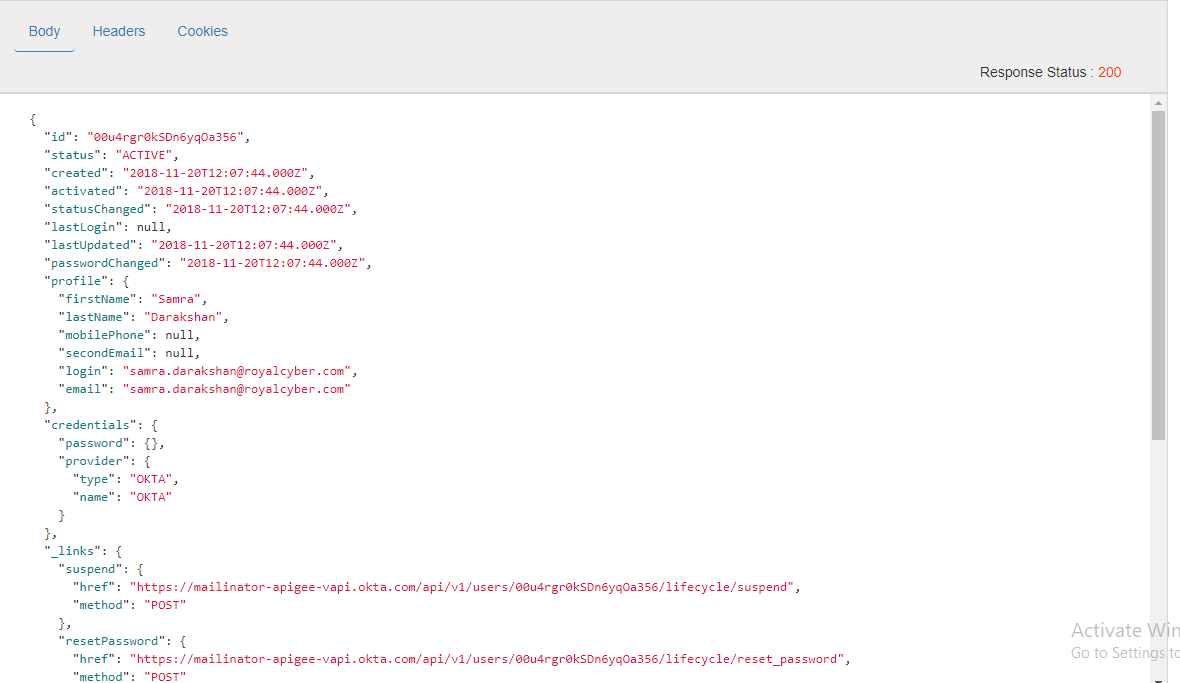
Authorization Header: SSWS 00pm1IxzJnb7Hlx7vom1lFGaXpAPPz4sqt659nKH77

First Name, last name, email, login, and password: provide your own

https://mailinator-apigee-vapi-admin.okta.com/api/v1/users?activate=true







{

"profile":

{"firstName": "Samra",

"lastName": "Darakshan",

"email": "samra.darakshan@royalcyber.com",

"login": "samra.darakshan@royalcyber.com"

},

"credentials":

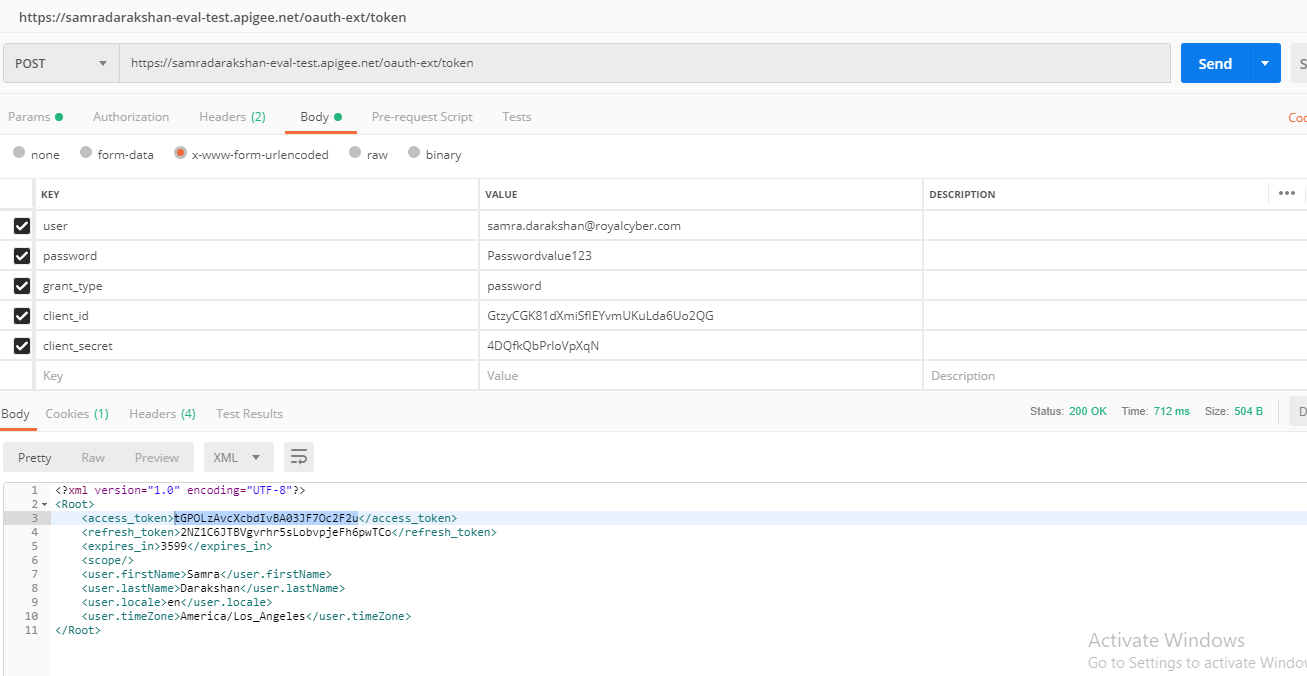
{

"password" : { "value": "Passwordvalue123" }

}

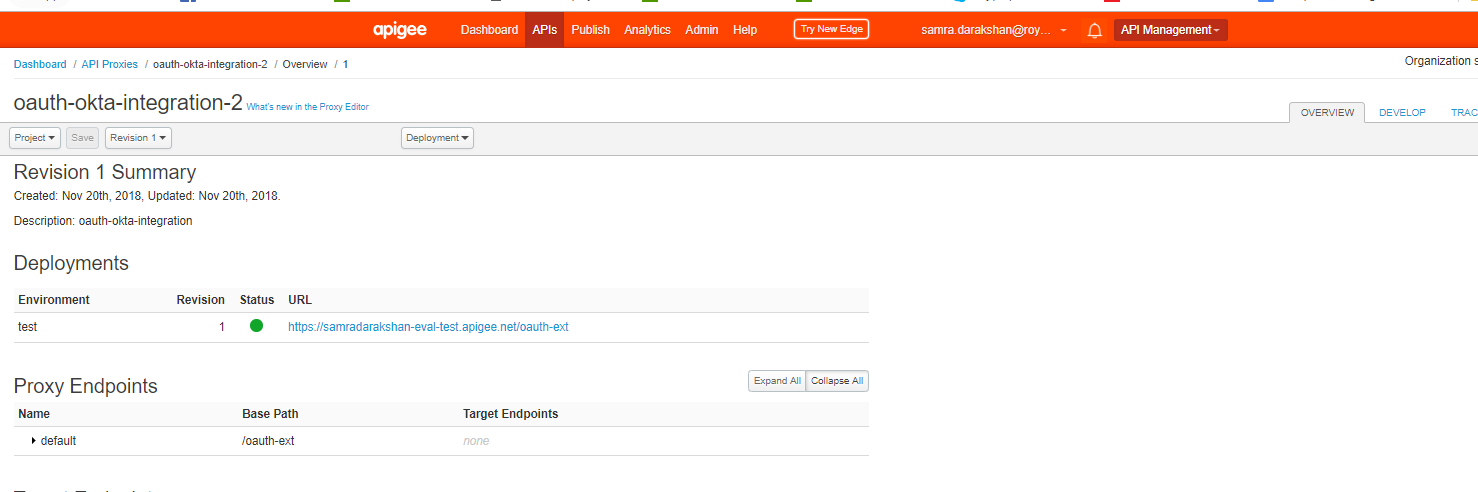
}

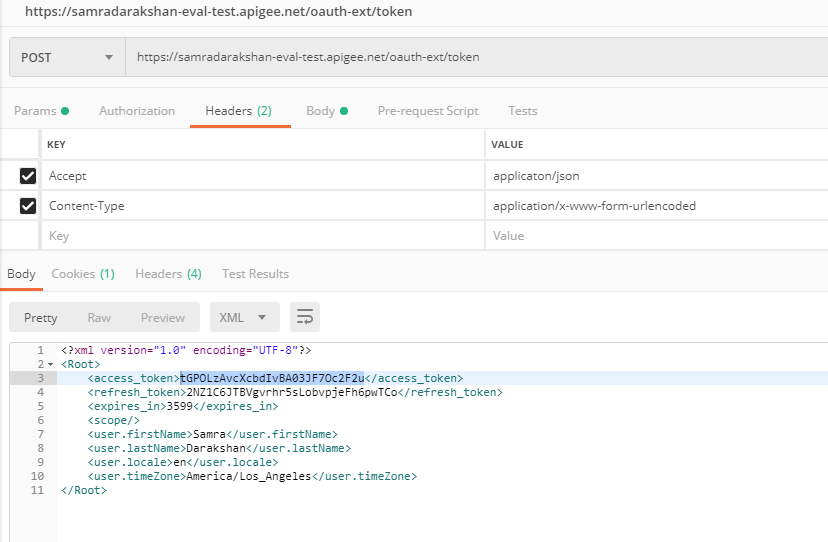
Gives credentioals for testing:



Gives token and get response:

1st test this to get token: okta integration





2nd this mock target using above token:

