# Using NGINX Controller API Management Module and NGINX App Protect to secure Open Banking transactions

Updated 1 week ago | Originally posted March 08, 2021 by Valentin Tobi F5 (/s/profile/0051T000008dpLtQAI)

Topics in this Article: nginx app protect (/s/articles?tag=nginx app protect), nginx controller (/s/articles?tag=nginx controller), nginx controller api management (/s/articles?tag=nginx controller api management), oauth (/s/articles?tag=oauth), open banking (/s/articles?tag=open banking), openid connect (/s/articles?tag=openid connect), security (/s/articles?tag=security)

Previous Article 16 of 16:Using NGINX Controller API Management Module and NGINX App Protect to secure Open Banking transactions

As Open Banking is concerned primarily with managing access to exposed banking APIs, the security aspect has always been of paramount importance.

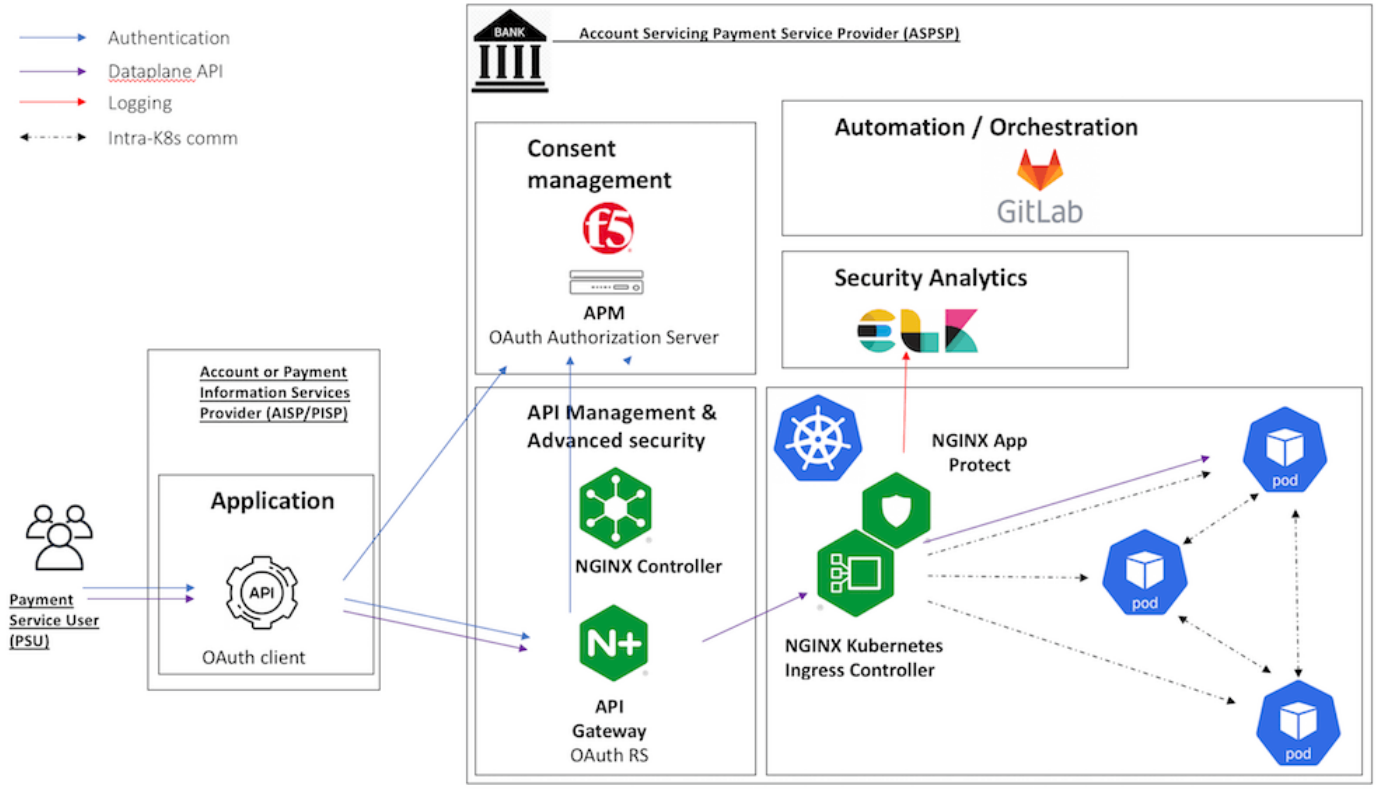
Securing Open Banking deployments is a vast topic, as security controls are distributed among different functions, such as user authentication at the Identity Provider level, user authorization and basic API security at the API Gateway level and advanced API security at the WAF level.

In this article we will explore how two NGINX products, Controller API Management Module and App Protect, can be deployed to secure Open Banking's OAuth Authorization Code flow.

## Physical setup

The setup used to support this article comprises of NGINX Controller API Management Module, providing API Management functions through an instance of NGINX API Gateway and NGINX App Protect deployed on a Kubernetes Ingress Controller providing advanced security for the Kubernetes-deployed demo application, Arcadia Finance.

These elements are being deployed and configured in an automated fashion using a Gitlab CI/CD pipeline. The visualization for NGINX App Protect is provided by NAP dashboards deployed in ELK.




**Note:** For the purpose of supporting this lab, APM was configured as an OAuth Authorization Server supporting OpenID Connect. Its configuration, along with the implementation details of the third party banking application (AISP/PISP), acting as an OAuth Client, is beyond the scope of this article.

In an OAuth Authorization Code Open Banking flow, the PSU (End User) is initiating an API request through the Account or Payment Information Services Provider (AISP/PISP Application) which first redirects the end user to the Authorization Server.

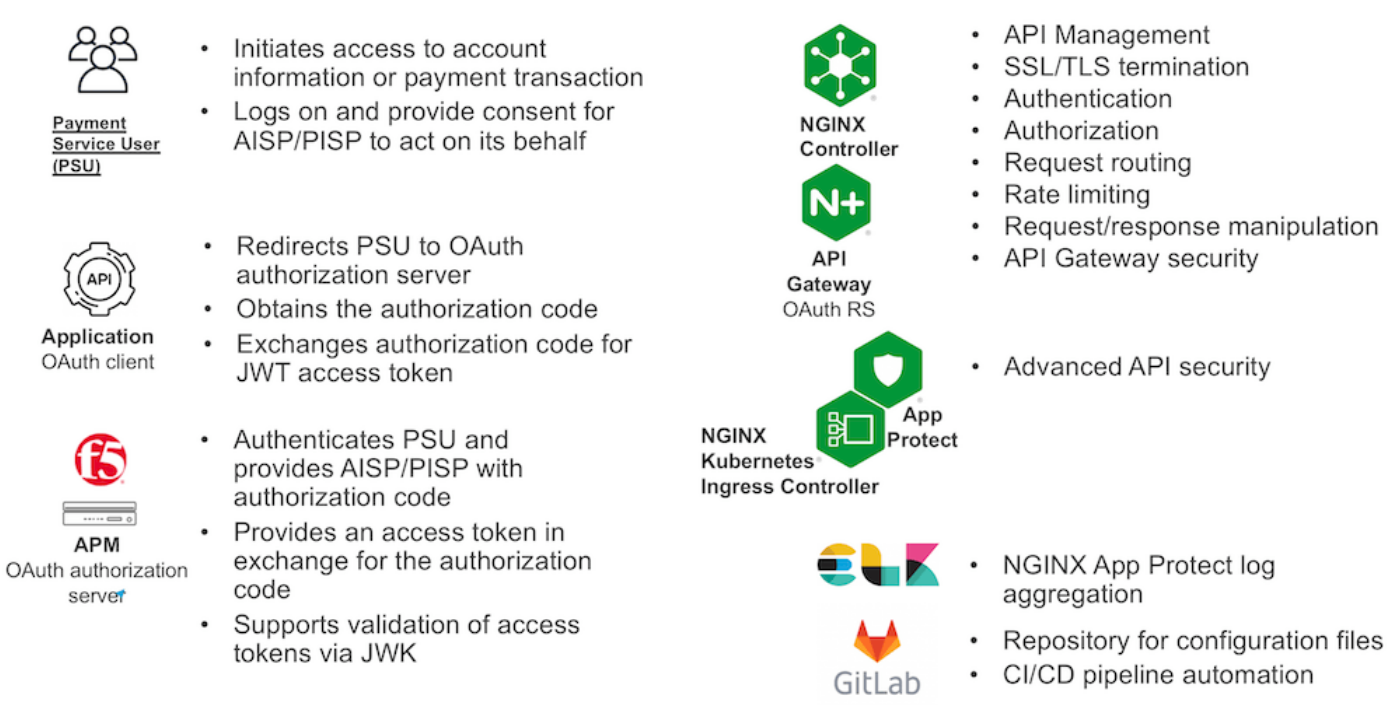
Strong Customer Authentication is being performed between the end user and Authorization Server which, if successful, will issue an authorization code and redirect the user back to the AISP/PISP Application.

The AISP/PISP Application will exchange the authorization code for an ID Token and a JWT Access Token, the latter will be attached as a bearer token to the initial end-user API request which will then be forwarded to the API Gateway.

The API Gateway will authenticate the signature of the JWT Access Token by downloading the JSON Web Key (JWK) from the Authorization Server and may apply further security controls by authorising the API call based on JWT claims and/or apply rate limits.

Worth noting here is the security function of the API Gateway, which provides positive security by allowing only calls conforming to published APIs, in addition to authentication and authorization functions.

The Web Application Firewall function, represented here by the NGINX App Protect deployed on the Kubernetes Ingress Controller (KIC), will add negative security protection, by checking the request against a database of attack signatures, and advanced API security, by validating the API request against the OpenAPI manifest and providing Bot detection capabilities.




## Configuration

To configure the NGINX Controller API Management Module, first create an Application by sending a POST request to '*https://{{ my_controller }}/api/v1/services/environments/env_prod/apps*' having the following body:

```
{
  "metadata": {
    "name": "app_api",
    "displayName": "API Application Arcadia",
    "description": "",
    "tags": []
  },
  "desiredState": {}
}
```

Then create an Identity Provider, pointed at the Authorization Server's JWK endpoint, by sending a PUT request to '*https://{{ my_controller }}/api/v1/security/identity-providers/bank_idp*' having the following body:

```
{
  "metadata": {
    "name": "bank_idp",
    "tags": []
  },
  "desiredState": {
    "environmentRefs": [
      {
        "ref": "/services/environments/env_prod"
      }
    ],
    "identityProvider": {
      "type": "JWT",
      "jwkFile": {
        "type": "REMOTE_FILE",
        "uri": "https://bank.f5lab/f5-oauth2/v1/jwks",
        "cacheExpire": "12h"
      }
    }
  }
}
```

Create an API definition by sending a PUT request to '*https://{{ my_controller }}/api/v1/services/api-definitions/arcadia-api-def/versions/v1*' with the following body:

```
{
  "metadata": {
    "name": "v1",
    "displayName": "arcadia-api-def"
  },
  "desiredState": {
    "specs": {
      "REST": {
        "openapi": "3.0.0",
        "info": {
          "version": "v1",
          "title": "arcadia-api-def"
        },
        "paths": {}
      }
    }
  }
}
```

Then import the OpenAPI definition by sending a PUT request to '*https://{{ my_controller }}/api/v1/services/api-definitions/arcadia-api-def/versions/v1/import*' with the OpenAPI JSON as a request body.

Publish the API definition by sending a PUT request to '*https://{{ my_controller }}/api/v1/services/environments/env_prod/apps/app_api/published-apis/prod-api*', with the following body:

```
{
  "metadata": {
    "name": "prod-api",
    "displayName": "prod-api",
    "tags": []
  },
  "desiredState": {
    "apiDefinitionVersionRef": {
      "ref": "/services/api-definitions/arcadia-api-def/versions/v1"
    },
    "gatewayRefs": [
      {
        "ref": "/services/environments/env_prod/gateways/gw_api"
      }
    ]
  }
}
```

Declare the necessary back-end components (in this example *webapi-kic.nginx-udf.internal* Kubernetes workload) by sending a PUT to '*https://{{ my_controller }}/api/v1/services/environments/env_prod/apps/app_api/components/cp_moneytransfer_api*' with the following body:

```
{
  "metadata": {
    "name": "cp_moneytransfer_api",
    "displayName": "cp_moneytransfer_api",
    "tags": []
  },
  "desiredState": {
    "ingress": {
      "uris": {
         "/api/rest/execute_money_transfer.php": {
          "php": {
            "get": {
              "description": "Send money to a friend",
              "parameters": [
                {
                  "in": "body",
                  "name": "body",
                  "required": true,
                  "schema": {
                    "type": "object"
                  }
                }
              ],
              "responses": {
                "200": {
                  "description": "200 response"
                }
              }
            },
            "matchMethod": "EXACT"
          }
        }
      },
      "gatewayRefs": [
        {
          "ref": "/services/environments/env_prod/gateways/gw_api"
        }
      ]
    },
    "backend": {
      "ntlmAuthentication": "DISABLED",
      "preserveHostHeader": "DISABLED",
      "workloadGroups": {
        "wl_mainapp_api": {
          "loadBalancingMethod": {
            "type": "ROUND_ROBIN"
          },
          "uris": {
            "http://webapi-kic.nginx-udf.internal:30276": {
              "isBackup": false,
              "isDown": false,
              "isDrain": false
            }
          }
        }
      }
    },
    "programmability": {
      "requestHeaderModifications": [
        {
          "action": "DELETE",
```

```
          "applicableURIs": [],
          "headerName": "Host"
        },
        {
          "action": "ADD",
          "applicableURIs": [],
          "headerName": "Host",
          "headerValue": "k8s.arcadia-finance.io"
        }
      ]
    },
    "logging": {
      "errorLog": "DISABLED",
      "accessLog": {
        "state": "DISABLED"
      }
    },
    "security": {
      "rateLimits": {
        "policy_1": {
          "rate": "5000r/m",
          "burstBeforeReject": 0,
          "statusCode": 429,
          "key": "$binary_remote_addr"
        }
      },
      "conditionalAuthPolicies": {
        "policy_1": {
          "action": "ALLOW",
          "comparisonType": "CONTAINS",
          "comparisonValues": [
            "Payment"
          ],
          "sourceType": "JWT_CLAIM",
          "sourceKey": "scope",
          "denyStatusCode": 403
        }
      },
      "identityProviderRefs": [
        {
          "ref": "/security/identity-providers/bank_idp"
        }
      ],
      "jwtClientAuth": {
        "keyLocation": "BEARER"
      }
    },
    "publishedApiRefs": [
      {
        "ref": "/services/environments/env_prod/apps/app_api/published-apis/prod-api"
      }
    ]
  }
}
```

Note the 'security' block, specifying the JWT authentication, the Identity Provider from where to download the JWK, the authorization check applied on each request and the rate limit policy.

The configuration used to deploy NGINX App Protect on the Kubernetes Ingress Controller can be consulted here (https://devcentral.f5.com/s/articles/Example-NGINX-App-Protect-deployed-on-Kubernetes-Ingress-Controller?page=1).

## Summary

In this article we showed how NGINX Controller API Management Module and NGINX App Protect can be deployed to protect API calls as part of the OAuth Authorization Code flow which is a fundamental flow of Open Banking.

## Links

UDF lab environment link (https://udf.f5.com/b/5f47d4ef-0175-43df-a342-99eca4458e4e#documentation).

Previous Article 16 of 16:Using NGINX Controller API Management Module and NGINX App Protect to secure Open Banking transactions

### Topics in this Article:

nginx app protect (/s/articles?tag=nginx app protect) nginx controller (/s/articles?tag=nginx controller) nginx controller api management (/s/articles?tag=nginx controller api management) oauth (/s/articles?tag=oauth) open banking (/s/articles?tag=open banking) openid connect (/s/articles?tag=openid connect) security (/s/articles?tag=security)