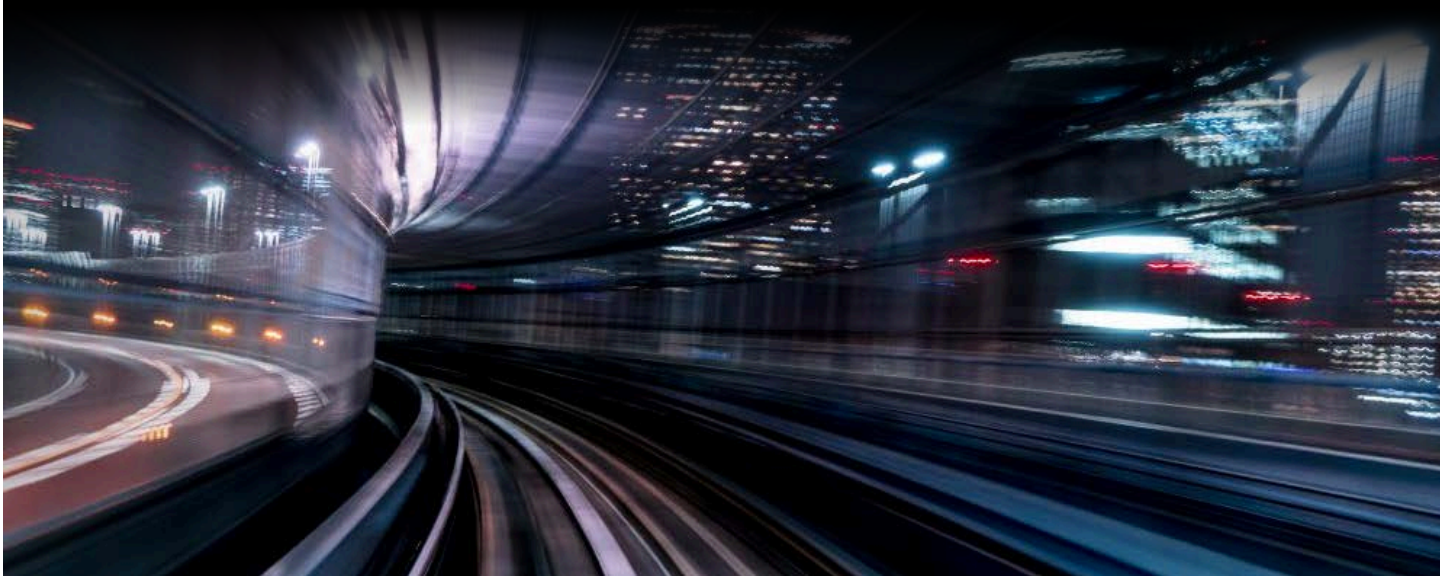
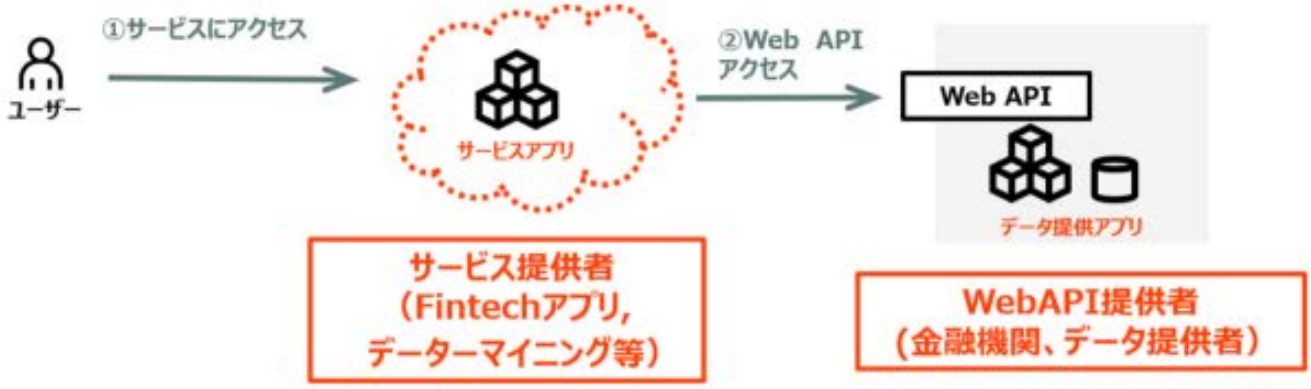
IceWall技術レポート：
「Amazon API Gateway」と「IceWall Federation OIDC/OAuth OP」との連携



1. はじめに

近年のFinTechの普及やデータマイニング需要の高まりを受け、企業が独自のAPIをWeb上に公開し、様々な情報を提供することは一般的になりつつあります。企業（API提供者）がWeb APIによって自社が持つデータを社外に公開することで、他の企業（サービス提供者）がこれを利用してユーザーに様々なサービスを提供する新しい形のサイトやビジネスが次々と登場しています。
しかし、ユーザーのプライベートな情報にサービス提供者がユーザーに代わってアクセスする事になるため、ユーザーの同意の取得やAPI提供者側でのアクセス認可をどのように行うのか等、セキュリティについて解決しなければならない課題があります。
本技術レポートではAmazon API GatewayとIceWall Federation OIDC/OAuth OPとの連携によってAmazon Web Service(AWS)上でWeb APIへのアクセス認可を行う仕組みとその動作検証についてご紹介します。

Web APIを使ったサービス



---

2. 標準規格OAuthを使ったアクセス認可の仕組み

Web APIに直接アクセスするのはユーザーではなくサービス提供者ですが、サービス提供者がユーザーのパスワードを預かって認証（ログイン）を行い、アクセス認可を受けるのは現実的ではありません。そのため、ユーザー認証はユーザーとAPI提供者の間で行い、Web APIへのアクセスをサービス提供者に許可するような仕組みが必要となります。これを実現するために標準規格であるOAuthもしくはOpenID Connectを利用します。

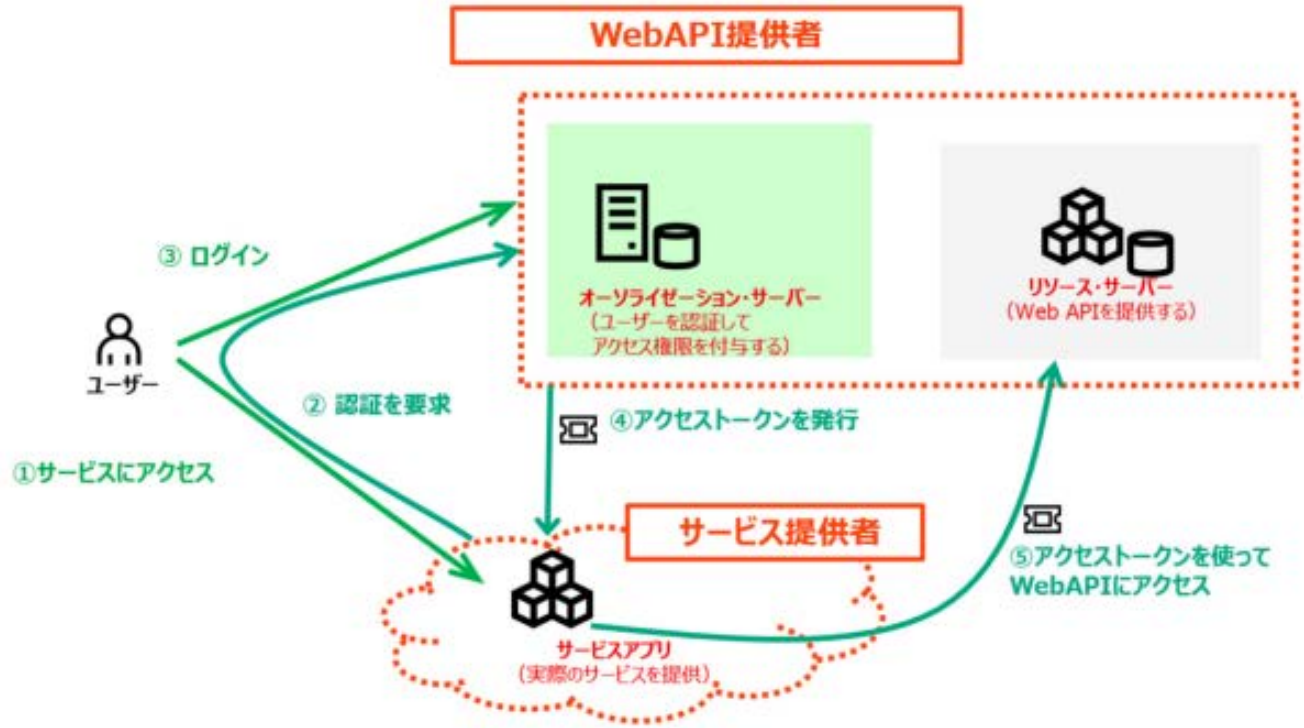
OAuth（OAuth 2.0）は一般的にWeb APIの認可で使われるプロトコルです。
OAuthでは、サービスの利用者（ユーザー）を「リソース・オーナー」と呼び、ユーザーを認証するのが「オーソライゼーション・サーバー」です。そしてAPI提供者を「リソース・サーバー」と呼び、Web APIにアクセスするサービス提供者を「クライアント」と呼びます。

「オーソライゼーション・サーバー」はユーザーの認証が完了すると「アクセストークン」を発行してアクセス権限を付与します。サービス提供者は「アクセストークン」を入手することでWeb APIへのアクセス権限を得る事ができます。

また、OpenID ConnectはOAuthの拡張であり、ユーザーの認証情報を含む署名付きトークン（IDトークンと呼ばれる）を付与する方法を規定しています。

OAuthを使ったWeb APIへのアクセス認可の概要図を右に示します。

ユーザーの同意に基づくWebAPIアクセス(OAuth)



3. Amazon API GatewayとIceWall SSOの連携

3.1. 使用するサービス・ソフトウェア

3.1.1. Amazon API Gateway

Amazon API Gatewayはユーザーが独自のAPIを簡単にデプロイできるAWS（Amazon Web Service）上のサービスです。Amazon API Gatewayでは流量制御やシステムのスケーリングを自動で行うため、インフラ環境を意識せずWeb APIをクラウド上で公開することが可能です。

Amazon API Gateway自体はOAuthなどのAPIアクセス認可プロトコルに対応していませんが、AWS Lambda上に独自の認可プログラム (Lambda Auth Function) を実装し呼び出す機能があります。

今回の連携では、この仕組みを用いてIceWall SSOのリバースプロキシ（フォワーダー）とそのバックエンドに配置したアクセスプログラムにアクセストークンを転送するようなLambda Auth Functionを作成し、受信したアクセストークンの正当性を検証します。

なおAmazon API Gatewayは認可情報のキャッシュ機能があるため、Web APIへのアクセスが起こる毎にアクセストークンの検証が行われる事はありません。

3.1.2. IceWall SSO

IceWall SSOはユーザーの認証を行い、さらに今回の連携ではLambda Auth Functionから転送されたアクセストークンの正当性の検証を行います。

3.1.3. IceWall Federation OIDC/OAuth OP

IceWall Federation OIDC/OAuth OPは、OAuthとOpenID Connectで規定されている認証・認可サーバーとしての機能を持ち、アクセストークン及びIDトークンの発行機能、ユーザーの認可情報の管理機能などを持ちます。今回の連携ではアクセストークンの発行と管理を行います。

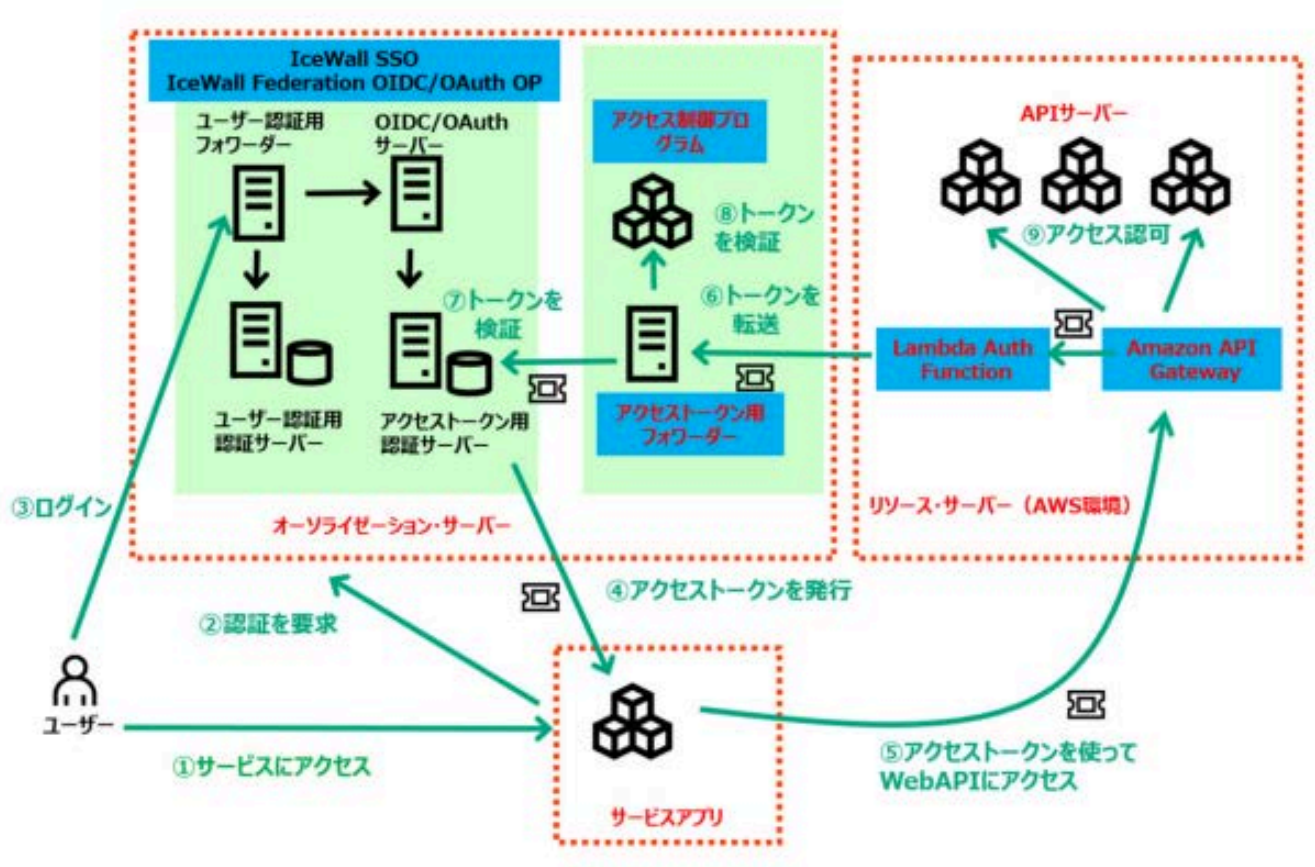
3.2. 連携システム構成

Amazon API GatewayとIceWall Federation OIDC/OAuth OPを連携したシステムの概要は右の図のとおりです。

IceWall Federation OIDC/OAuth OPが発行したアクセストークンの情報はアクセストークン用認証サーバーで管理されます。Web APIが受け取ったアクセストークンの検証も同じ認証サーバー使って行います。

処理の流れは以下のとおりです。
① ユーザーがサービスを利用するためにサービス提供者（サービスアプリ）へアクセスします。
② ユーザー認証（ログイン）が未実施の場合、サービスアプリからオーソライゼーション・サーバーへ認証を要求します。
③ ユーザーがログインします。
④ ログインが完了すると、オーソライゼーション・サーバーはアクセストークンを発行しサービスアプリに渡します。発行されたアクセストークンの情報はアクセストークン用認証サーバーで管理されます。
⑤ サービスアプリはアクセストークンを付加してWeb APIにアクセスします。
⑥ アクセストークンがLambda Auth Functionを経由してIceWall SSOのリバースプロキシ（フォワーダー）に送信されます。
⑦　IceWallフォワーダーはアクセストークン用認証サーバーを参照して受信したアクセストークンが正当なものかを確認します。アクセストークンが正当であれば後段のアクセス制御プログラムに転送します。アクセストークンが正当でなければエラーを返します。
⑧　アクセス制御プログラムは、その他アクセス認可のために必要な検証や情報の照合を行います。問題があればエラーを返します。
⑨ Lambda Auth Functionにエラーが返されなければ、Web APIへのアクセスが認可されます。

連携システム概要



4. 動作検証

実際にAmazon API GatewayとIceWall SSOの連携システムを構築して動作検証を行いました。以下にWeb APIのアクセス認可に関する部分について、作成したコードとAmazon API GatewayおよびIceWall SSOの設定方法について説明します。

4.1. Lambda Auth Functionの作成

Lambda Auth Functionのコードを以下のように作成しました。

```
'use strict';

var async   = require('async');
var request = require('request');

function verify_token(access_token, callback)
{
  var header = {'Authorization' : 'Bearer ' + access_token};

  request({
    url: 'IceWallフォワーダーのURL',
    method: 'GET',
    headers: header,
    encoding: 'utf8',
    json: true
  }, function(error, response, body) {
    if (error) {
      callback(error);
    }else {
      callback(null, body);
    }
  });
}

function generate_policy(principal_id, effect, resource)
```

```
{
  return {
    principalId: principal_id,
    policyDocument: {
      Version: '2012-10-17',
      Statement: [{
        Action: 'execute-api:Invoke',
        Effect: effect,
        Resource: resource
      }]
    }
  };
}

exports.handler = (event, context) => {
    async.waterfall([
    function(callback) {
      //OP側APIの呼び出し
      verify_token(event.authorizationToken, callback);
    },
    function(response, callback) {
      switch (response.active) {
        case true:
          context.succeed(generate_policy(response.subject, 'Allow', event.methodArn));
          break;
        default:
          context.succeed(generate_policy(response.subject, 'Deny', event.methodArn));
          break;
        }
        callback(null);
      }
    ], function (error) {
  if (error) {
    context.fail(error);
  }
```

```
});


};
```

4.2. Amazonカスタムオーソライザーの設定

作成したLambda Auth Functionをカスタムオーソライザーとして以下の手順で設定しました。(以下のAWS管理画面は2017年12月時点のものです。)

1) API Gatewayの管理画面からオーソライザータブを開き、上記のコードをデプロイした際の関数名を「Lambda関数」欄に入力します。また「トークンのソース」欄は"Authorization"と入力します。名前は任意の値を入力してください。

2) 認証を設定したいAPIのエンドポイントを選択し、「メソッドリクエスト」をクリックします。

3) 認証として先ほど作成したカスタムオーソライザーを選択します。

4.3. IceWall SSOアクセストークン用フォワーダーの設定

IceWall Federation OIDC/OAuth OPが発行し、Lambda Auth Functionによって転送されたアクセストークンの正当性を検証するために、アクセストークン用フォワーダーが動作するApacheに以下の設定を行い、アクセストークンをIceWall SSOのCookieとして変換します。

```
SetEnvIf ^[Aa]uthorization "Bearer( *)(.*)" IW_BEARER=$2
RequestHeader set Cookie IW_INFO=%{IW_BEARER}e
```

4.4. アクセス制御プログラムの作成

アクセストークン用フォワーダーの後段に配置するアクセス制御プログラムを以下のように作成しました。今回の検証では転送された内容をそのままJSON形式でエコーバックする簡単なプログラムにしました。

```
public class UserInfoServlet extends HttpServlet{

   protected void doGet(HttpServletRequest request, HttpServletResponse response){

      String uid = request.getHeader("Uid");
      String cid = request.getHeader("CLIENT-ID");
      String sub = request.getHeader("SUBJECT");
      String scope = request.getHeader("SCOPE");
      String tokenType = request.getHeader("TOKEN-TYPE");
      String tokenExp = request.getHeader("TOKEN-EXP");
      String tokenIssueTime = request.getHeader("TOKEN-ISSUE-TIME");
      String authnTime = request.getHeader("AUTHN-TIME");

      String responseJSON = null;

      if(sub == null){
         responseJSON = "{ " +
               "¥"code¥":200, " +
               "¥"error¥":¥"subject_error¥", " +
               "¥"message¥": ¥"subject not found.¥" " +
               "}";
      }else{
```

```
            responseJSON = "{ " +
                  "¥"active¥": true, " +
                  "¥"user_id¥":¥"" + uid + "¥", " +
                  "¥"client_id¥":¥"" + cid + "¥", " +
                  "¥"scope¥":¥"" + scope + "¥", " +
                  "¥"sub¥":¥"" + sub + "¥", " +
                  "¥"token_type¥":¥"" + tokenType + "¥", " +
                  "¥"token_exp¥":¥"" + tokenExp + "¥", " +
                  "¥"issue_time¥":¥"" + tokenIssueTime + "¥", " +
                  "¥"authn_time¥":¥"" + authnTime + "¥" " +
                  "}";
         }

         try {
            response.setContentType("application/json;charset=UTF-8");
            PrintWriter out;
            out = response.getWriter();
            out.print(responseJSON);
         } catch (IOException e) {
         }
      }
   }
```

アクセス制御プログラムでは、アクセストークン用認証サーバーに格納されたSCOPE値（OAuthで規定されたユーザーの認可同意範囲を示す値）やデータベース内のユーザー属性を用いることで、より詳細なアクセス制御を実装することもできます。

4.5. Web APIへのアクセス

サービスアプリがAmazon API Gatewayへアクセスする際はHTTPヘッダーに以下を追加します。

Authorization: {IceWall Federation OIDC/OAuth OPが発行したアクセストークン}

5. まとめ

Amazon API GatewayとIceWall Federation OIDC/OAuth OPの連携で、セキュアなWeb APIの公開を最小限の開発工数で実現することができます。今回、実施の検証環境を構築し動作を確認することができました。
今後もより多くの企業やサイトで Web APIの公開が進むと予想されます。クラウド環境での公開では今回のシステムをぜひご検討ください。

2018/1/31 新規掲載

執筆者　：日本ヒューレット・パッカード株式会社
Pointnext事業統括 IceWallソフトウェア本部 認証技術開発センター
砂田 健太郎

技術レポート一覧へ →

お問い合わせ先一覧

企業情報

会社情報

アクセス

お問い合わせ

採用情報

HPEについて

インクルージョン & ダイバーシティ

サステナビリティと企業責任

経営幹部

お知らせ

ニュースルーム

イベント・セミナー

新着情報

重要なお知らせ

パートナー

パートナープログラム

認定資格制度

OEMソリューション

サポート

製品サポート

ソフトウェア & ドライバー

標準保証確認

オペレーショナルサポート

教育とトレーニング

製品リサイクル

機器部品の妥当性確認

コミュニティ

HPE Japan ブログ

リソース

お客様事例

ご購入方法

オンラインストア

HPE Customer Center

Eメール登録

ドキュメントライブラリ

Resource Library

ビデオギャラリー

金融サービス

日本 (ja)