Authenticating With JWT

Intro to JWT

JWT (JSON Web Token) is "an open, industry standard method for representing claims securely between two parties" [JWT.io]. There are three parts to a JWT: Header, Payload, and Signature. The Header represents the algorithm used to sign the token and a type of token that we are sending. In the Payload there is the specific data that is being sent by the signing party. The Signature contains the encoded header and payload and signed with a secret with the algorithm defined in the Header. This Signature is used to verify that the token has not changed along it path to the other party. And with tokens signed with a private key, it verifies it came from the right party. JWT's are specifically designed to be lightweight tokens to be sent over HTTP requests. You notice that many of the claims (JSON Data) in the JWT are shortened to three characters. And each section is encoded in Base64Url. This is all to make the JWT as small as it can be and send over REST request with ease. Each of section of the JWT are separated with a ".". An example JWT look something like this:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6InRlc3Rpbmdhc2QxMjMiLCJhY2NvdW50VHlwZSI6ImVtcGxveWVlIiwiaWQiOjI2LCJpYXQiOjE1MjM2MzczMTl9.GV2zesnIG4ToCfXgv_6pBe7jiuFEiIjsBEuEm9x0BZw

Using JWT in KennUWare

As you probably already know the HR team has developed an authentication mechanism to be used to log into KennUWare. Every silo can request to be authenticated with us. We will return a JWT signed with a secret key. The payload will include the user ID, the username, and the type of user. The process to authenticate a user is straight forward. To start, you will ask the authentication server to authenticate your user by providing a username and password. We will send back a JSON response that includes a status message and a JWT. Now you can verify the message to confirm it hasn't changed and that we sent it. If it looks good you can pull the user ID out of the JWT's Payload and request the user's role from HR using that user ID. Now you can send the JWT as an Authorization Bearer header to every API endpoint to authorize the user. We will go into how to do each step in a bit.

What is in our JWT

Our JWT uses HS256 algorithm. That is included in the Header. And since again this is a JWT we also include set the type to JWT. In our Payload we include three important things: the username, the user ID, and the account type. Each represent the current user that has been authenticated. Also included in the payload is "iat", which represents the time that the JWT was created. Our Signature is signed with the key "secretkey" (We know this is not the best way to do it, but it will simplify the verifying process without signing public/private keys). You will use that to verify the JWT you received. An example decoded JWT is below:

```
Header:
{
  "alg": "HS256",
  "typ": "JWT"
}
Payload:
{
  "username": "zach",
  "accountType": "employee",
  "id": 15,
  "iat": 1523546886
}
Signature
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  “secretkey”
)
```

Creating an User

To create a user, you will need to log into the HR silo and create an employee. Below are the steps in doing so.

1. Go to http://kennuware-1772705765.us-east-1.elb.amazonaws.com
2. Login with
   Username: root
   Password: admin
3. Click “Employees” on the navigation bar
4. Click “Create new”
5. Fill out all form fields
   Important: The “Role” field is what is used for user roles within the site. If you need one that is not there, please contact us on slack.
   Note: For the username field uses something unique. The page currently fails if the username is already being used. Also, don’t use your real password since they are stored in plain text currently. That way if you forget the password we can get it for you.
6. Create.

Using/Verifying JWT in Code

Alright, it’s time to implement some authentication. We will start with client-side code for those silos that use ReactJS or AngularJS. On the client side we do not want to verify our JWT.

This is because we don't want to share the secret key with our users. It's important to always keep the key secret. We'll verify on the server. Let's start by making a request to our authentication server.

NOTE: There is a chance that this code doesn't work (since I'm didn't run this code). It's here to help you all understand how to implement this.

```
var request = require("request");

var options = { method: 'POST',
  url: 'https://api-gateway-343.herokuapp.com/auth/login',
  headers:
   {'Content-Type': 'application/json'},
  body: { username: 'user', password: 'test' },
  json: true };

request(options, function (error, response, body) {
  if (error) throw new Error(error);

});
```

Next, we will check if our JWT is valid, so inside the request method we will check if the status was a success and get the payload of the JWT

```
var program = require('commander');
var rs = require('jsrsasign');
var rsu = require('jsrsasign-util');

...
request(options, function (error, response, body) {
  if (error) throw new Error(error);

  if (body.status === "Success"){
     var jwt = body.jwt.split(/\./);
     var payload = rs.KJUR.jws.JWS.readSafeJSONString(rs.b64utoutf8(jwt[1]));
  } else {
     // tell the user that the username and password
     // they provided was incorrect
  }
});
```

Now, let's call HR's API to get the employee and their role for the user that is logged in.

```
var userId;
...

     var payload = rs.KJUR.jws.JWS.readSafeJSONString(rs.b64utoutf8(jwt[1]));
     userId = payload.id;
...

var options = { method: 'GET',
  url: 'http://kennuware-1772705765.us-east-1.elb.amazonaws.com/api/employee',
  qs: { userId: userId }
};

var user;

request(options, function (error, response, body) {
  if (error) throw new Error(error);

  user = body;
});
```

Now we got our JWT, we got the employee that is logged on, now we can authorize them on our server. This is done through an Authentication Bearer Header. All calls through the API should include this REST header parameter. Here is some example code on how to do this.

```
var request = require("request");

var options = { method: 'POST',
  url: 'https://api-gateway-343.herokuapp.com/hr-api/employee',
  headers:
   { Authorization: 'Bearer ' + jwt,
     'Content-Type': 'application/json' } };

request(options, function (error, response, body) {
  if (error) throw new Error(error);

  console.log(body);
});
```

For our server side code https://auth0.com/blog/securing-asp-dot-net-core-2-applications-with-jwts/ provides a good example of how to implement JWT. For Java this is a good example: https://github.com/auth0/java-jwt. Remember we are signing our token with "secretkey".

If you have any questions feel free to contact us at #team_humanresources on slack.