Protecting Node-RED

Node-RED is flexible in how it accommodates authentication and authorization with OAuth, SAML and OIDC.

Node-RED Admin can be protected with a login. This login can be activated by modifying the **settings.js** file that is bundled with Node-RED. More conventionally this can be achieved through adding a username and a password hash generated with Node-RED admin.

However, **SAML** (Security Assertion Markup Language), **OAuth**, and **OIDC** can also be used in conjunction with Node-RED to generate a secure login interface.

Node-RED documents that it can utilize a wide number of strategies provided by **passport** (http://passportjs.org/). Out of the box Node-RED has authentication modules available for both Twitter and GitHub. These can be used as a template for similar OAuth-based authenticating strategies.

What is SAML?

Security Assertion Markup Language (SAML) is an XML-based open-standard for transferring identity data between two parties: an identity provider (IdP) and a service provider (SP) (Auth0 by Okta. SAML Authentication, Explained). The identity provider performs authentication and passes the user's identity, whilst the service provide trusts the identity provider, usually by certificate validation, and authorizes the given user to access the requested resource.

SAML vs OAuth vs OIDC

It is very easy to confuse SAML with OAuth as they can both be used for Single Sign-On (SSO). SAML is an **authentication** process, whilst OAuth is an **authorization** process. SAML is user specific whilst OAuth tends to be application specific (Okta. SAML vs. OAuth: Comparison and Differences).

SAML allows for users to be managed from a central location. One password unlocks all the services a user requires.

With OAuth a user typically needs to sign in to each application with the same OAuth identity provider.

OIDC (OpenID Connect) is a strategy similar to SAML, but unlike SAML it allows for data to be sent to a server via a HTTP redirect as well as to be posted directly.

Which method to choose?

Each method has its own pros and cons. However, **SAML** is the most secure. **OIDC** is the most modern. are the most secure as they allow for authentication-related data to be posted back to the service provider. They also both offer the most versatility as claims can be added to the response object.

This chapter provides instructions to apply ...

- Node-RED OAuth authentication with GitHub
- Node-RED OIDC Authentication with Azure Active Directory
- SAML: single login strategy for Node-RED

Node-RED OIDC Authentication with Azure Active Directory

OIDC authentication is possible with the Azure Active Directory.

1. To install the relevant **passport** package, open the Command Prompt and enter:

$ npm install passport-azure-ad

2. Create an Azure app with a redirect URL configured to:

http://localhost:1880/auth/strategy/callback

3. Following the installation, configure the **settings.js** file.

On **line 119** there is the start of a section called **securing Node-RED**. If enabled, you will need to **disable** the default form of authentication.

Replace any auth admin blocks with the code snippet below, adding the tenant GUID, client ID and client secret:

```
adminAuth: {
  type: "strategy",
  strategy: {
    name: "azuread-openidconnect",
    label: "Sign in with Azure",
    icon: "fa-windows",
    strategy: require("passport-azure-ad").OIDCStrategy,
    options: {
      identityMetadata: "https://login.microsoftonline.com/###TENANT-GUID###/v2.0/.well-known/openid-configuration",
      clientID: "###CLIENT-ID###",
      clientSecret: "###CLIENT-SECRET###",
      responseType: "code",
      responseMode: "query",
      redirectURL: "http://localhost:1880/auth/strategy/callback",
      allowHttpForRedirectUrl: true,
      scope: ['email', 'profile'],
      verify: function (profile, done) {
        profile.username = profile.displayname;
        done(null, profile);
      },
    },
  },
  users: function(user) {
    return Promise.resolve({username: user, permissions: "*" }];
  },
},
```

Note that:

- The callback URL is simply the URL of the Node-RED homepage followed by '/auth/strategy/callback'.
- OIDC is very similar to SAML in how it responds to authentication requests: it posts a response back to the server. However, the response type can be modified by defining response type as "code" and response mode as "query" within the passport implementation. These settings instruct the identity provider to use a redirect with an authorization token to grant a user access to the service.
- The profile image must be set via a modification to the verify callback function. From the code snippet you can see that the image is applied to the profile object from the processed OAuth response.

Note: This is an open ID strategy and that it authenticates all users on the same Azure Tenant. This is if application-level permissions have been set in Microsoft Entra.

Loggin in

The following images illustrate the authentication flow, from login through to 2FA via Microsoft Online. After authentication with 2FA the user is redirected back to Node-RED. As with OAuth based systems, a user once validated does not need to revalidate their identity to login.

SAML: single login strategy for Node-RED

The SAML protocol provides a single login strategy for Node-RED. This is a basic overview of how this works; not a recommendation. As an organization you may want to add more layers of security.

Creating an SSO with SAML via the Azure Portal

One way to create an SSO that implements SAML is to utilize the Enterprise apps feature available within the Azure Portal.

1. Go to the Azure portal (https://portal.azure.com/#home), then go through to the **Manage Entra ID** section.
2. In the left-hand side menu under **Manage** click **Enterprise Registrations**.
3. Then click **New application** to register a new Enterprise application.
4. Download the certificate from the SAML configuration page in the Azure portal. The required certificate is the base64 formatted one.
5. Copy the certificates downloaded from the identity provider over to the same directory as the **settings.js** file.

Note: There may be a limit of one SAML-based application per tenant or per developer account subscription.

More can be read on SAML with enterprise apps in the Microsoft documentation: https://learn.microsoft.com/en-us/entra/identity/enterprise-apps/add-application-portal-setup-sso.

Integrating SAML into Node-RED

1. To install the relevant **passport** package, open the Command Prompt and enter:

$ npm install passport-saml

Note: The passport saml library defined in the passport JS documentation has since node v4.0 been deprecated. It is advisable to use the new library: https://www.npmjs.com/package/@node-saml/passport-saml.

2. Following the installation, configure the **settings.js** file.

On **line 119** there is the start of a section called **securing Node-RED**. If enabled, you will need to **disable** the default form of authentication.

Replace any auth admin blocks with the code snippet below:

```
adminAuth: {
  type: "strategy",
  strategy: {
    name: "saml",
    label: "Sign in with Entra ID",
    icon: "fa-windows",
    strategy: require("@node-saml/passport-saml).Strategy,
    autoLogin: false,
    options: {
      issuer: "azure-saml-test",
      entryPoint: "https://login.microsoftonline.com/0dc3e27c-8123-4477-922f-
9bd3863cc219/saml2",
      idpCert: require("fs").readFileSync(
        require("path").join(_dirname, "sso.cer"),
      ),
      callbackURL: "http://localhost:1880/auth/strategy/callback",
      callbackMethod: "POST",
      verify: function (profile, done) {
        profile.username = profile.nodeREDadmin;
        done(null, profile);
      },
    },
  },
  users: function(user) {
    return Promise.resolve({username: user, permissions: "*" }];
  },
},
```

Note that:

- SAML differs slightly from OAuth and OIDC. An issuer, entry point, certificate, path and call back method need to be defined.
- Unlike OAuth there is no prerequisite for client and secret tokens. Instead, the certificate downloaded from the identity provider needs to be copied over to the same directory as the **settings.js** file.
- The issuer is the identity providers entity ID. The entry point is the SAML-P entry point (this is IdP specific).
- Node-RED's auto login feature is useful for bypassing the Node-RED admin login. However, this feature is not fully compatible with SAML.
  Setting autoLogin to false disables the feature and allows a user to log out of Node-RED whilst they are signed in to other services authenticated via the IdP.
- By default, Node-RED sends the callback to an endpoint listening for GET requests. To allow SAML to send a POST request instead, the callback method needs to be defined.

- The profile image must be set via a modification to the verify callback function. From the code snippet you can see that the image is applied to the profile object from the processed OAuth response.
- Auto login can be enabled, but with SAML it is advisable to set this to false. If you have logged into other system wide services, it is not possible to logout.
- SAML unlike OAuth allows you to add claims to the response. These claims contain user related meta data. As only the user email address is readily available in the processed SOAP response, we added to the identity provider a claim called ‘nodeREDadmin’.