



Edge Fundamentals

Dealing with Secured Backend

Building Target Authentication

Building our Proxy against a target that uses Basic Authentication provides options to store credentials on the Edge Platform.

Which poses the question -

"From the options below, which is the best method to store credentials ?"

1. Hard Code Them
2. Replace Them at Build Time
3. Key Value Map
4. Encrypted Key Value Map
5. Node.js Vault

Hard Coding using Assign Message

```
1. <AssignMessage name="assignSetAuthHeader">
2.   <Set>
3.     <Headers>
4.       <Header name="Authorization">Basic ASDfasdfas23434radsfaasdf
5.     </Header>
6.   </Headers>
7. </Set>
8.   <AssignVariable>
9.     <Name>username</Name>
10.    <Value>cleartext_user</Value>
11.  </AssignVariable>
12.  <AssignVariable>
13.    <Name>password</Name>
14.    <Value>cleartext_password</Value>
15.  </AssignVariable>
16.  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
17.  <AssignTo createNew="false" transport="http" type="request"/>
18. </AssignMessage>
```

Build Time Replacement

```
1. <AssignMessage name=""assignSetAuthHeader"">
2.   <Set>
3.     <Headers>
4.       <Header name="Authorization">Replace Me</Header>
5.     </Headers>
6.   </Set>
7.   <AssignVariable>
8.     <Name>username</Name>
9.     <Value>replace_me</Value>
10.  </AssignVariable>
11.  <AssignVariable>
12.    <Name>password</Name>
13.    <Value>replace_me</Value>
14.  </AssignVariable>
15.  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
16.  <AssignTo createNew="false" transport="http" type="request"/>
17. </AssignMessage>
```

Build Time Replacement (contd.)

```
1. {
2.   "configurations": [{
3.     "name": "dev",
4.     "policies": [{
5.       "name": "assignSetAuthHeader.xml",
6.       "tokens": [{
7.         "value": "Basic ASDfasdfes23434redsfeasdfs",
8.         "xpath": "/AssignMessage/Set/Headers/Header/@name=Authorization"
9.       },
10.      {
11.        "value": "cleartext-username",
12.        "xpath": "/AssignMessage/AssignVariable[Name='username']/Value"
13.      },
14.      {
15.        "value": "cleartext-password",
16.        "xpath": "/AssignMessage/AssignVariable[Name='password']/Value"
17.      }
18.     ]
19.   }
20. ],
21.   "proxies": [],
22.   "targets": []
23. }]
```

Storing Credentials in a Node.js Access Vault

Step 1: Build Vault and Vault Data Via API

```
1. curl https://api.enterprise.apigee.com/v1/o/{org}/vaults
2.   -H "Content-Type: application/json"
3.   -d '{"name": "vault" }' -X POST
4.
5. curl https://api.enterprise.apigee.com/v1/o/{org}/vaults/{vault}/entries
6.   -H "Content-Type: application/json"
7.   -d '{"name": "value1", "value": "verysecret" }' -X POST
```

Step 2: Retrieve Data in Node.js App

```
1. var apigee = require('apigee-access');
2. var orgVault = apigee.getVault('vault1', 'organization');
3.   orgVault.get('key1', function(err, secretValue) {
4.     // use the secret value here
5.   });
```

Storing Credentials in a Key Value Map (encrypted)

Step 1: Create KVM and KVM Data Via UI

The screenshot shows a 'New Key Value Map' dialog box. The 'Name' field is set to 'credMap' and the 'Encrypted' checkbox is checked. Below the dialog, the 'credMap' is expanded, showing a table with two entries: 'username' with value 'foundationUser' and 'password' with value 'Test1234'. Each entry has 'Edit' and 'Delete' buttons. An '+ Entry' button is at the bottom right of the table.

KEY	VALUE	ACTIONS
username	foundationUser	Edit Delete
password	Test1234	Edit Delete

+ Entry

Storing Credentials in a Key Value Map (encrypted)

Step 2: Retrieve Data via KVM Policy

```
<KeyValueMapOperations name="getEncrypted" mapIdentifier="credsMap">
  <Scope>environment</Scope>
  <Get assignTo="private.encryptedUsername" index="1">
    <Key>
      <Parameter>username</Parameter>
    </Key>
  </Get>
  <Get assignTo="private.encryptedPassword" index="1">
    <Key>
      <Parameter>password</Parameter>
    </Key>
  </Get>
</KeyValueMapOperations>
```


Lab

Please follow the steps provided [here](#)

THANK YOU