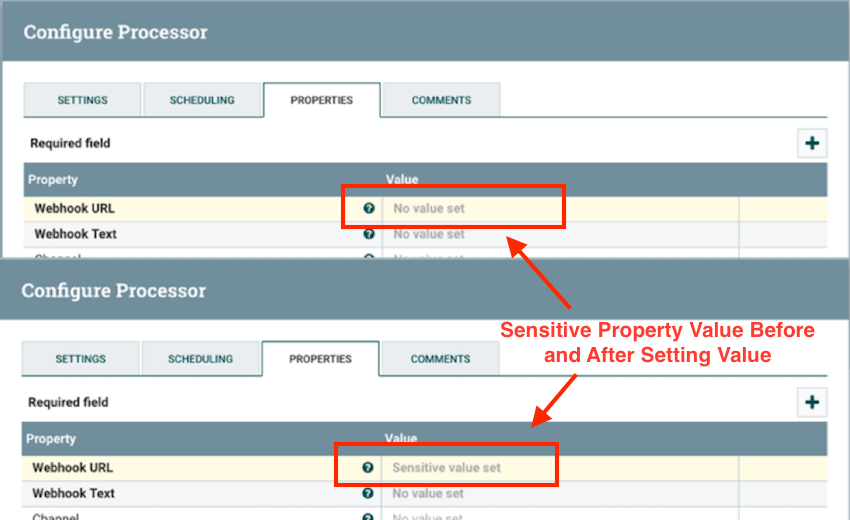
**Dealing with sensitive properties**

1. What is a Sensitive Property?

Apache NiFi processors sometimes need to store some sensitive values like a password, an access token, webhook URL etc. All these types of properties are marked as sensitive values and once we type in the value in the processor properties UI and saves, they will no longer be visible.



Sensitive Property Values

Now, these values are obviously stored in the flow.xml.gz file so that it can be used when required. To make is secure these values are encrypted

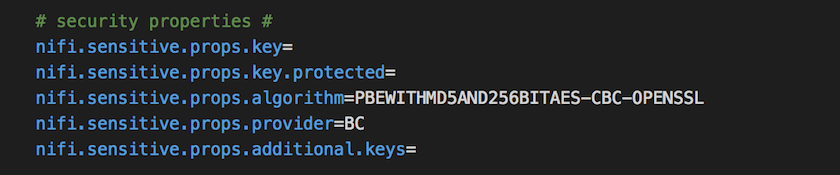
**<name>password</name>**

**<value>enc{AE07E899FFBA659B2E9F9OC77DB76DF94020NG3A2}</value>**

2. How Sensitive Properties are Encrypted?

So now we know that these sensitive values are encrypted and decrypted but how this is done is the question.

PBEWITHMD5AND256BITAES-CBC-OPENSSL is default encryption algorithm which is used. This means that actual cipher used is AES with CBC mode of operation. The default algorithm, as well as the key, are stored in the nifi.properties file under the property names nifi.sensitive.props.algorithm and nifi.sensitive.props.key



nifi.properties file with the encryption properties

The value of the key is used to drive a 256-bit key using Open-SSL EVP\_BytesToJey key derivation function(KDF). If the key property is not set in the properties file (as it is in the screenshot above), then a default value will be used by the encryption function.

Once the key is derived, a salt is generated, cipher text is available and the sensitive value is encrypted; it is then followed by concatenating the encrypted value, salt, and cipher and then finally encoding them in hexadecimal. This final value is then stored in flow.xml.gz and is wrapped in “enc{}” so that is will be easily identifiable as encrypted value.

The encryption/decryption process is performed by the core NiFi framework and is not accessible to any extension code. These values are never returned to client side ever again.

3. Setting-up/Migrating encryption key

It is highly advised that the keys are set in the properties file and default key should not be used especially in the production environment. So, we will see how to set this encryption key:

If the NiFi instance is freshly installed and there is no dataflow on the canvas yet. Then we can just go ahead and edit nifi.properties file and add a value for the property nifi.sensitive.props.key. For new instance of Apache NiFi, it is as easy as this.

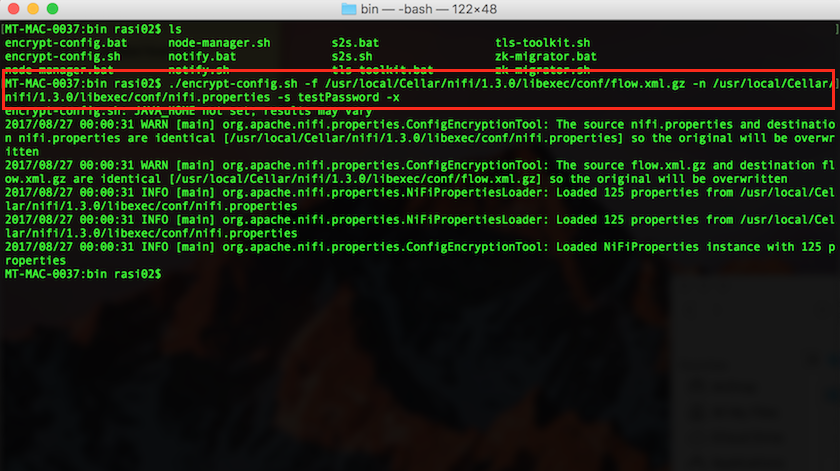
But if this is not the case and there are already dataflows on NiFi canvas, then directly editing in the properties file will not work for existing flows as the sensitive properties in those flows are already encrypted using the default values. In that case, we will need to migrate the existing flow in order to change the key. For this purpose, there is a encrypt-config.sh script available in the nifi-toolkit. This script will help us in migrating and changing to the new key.

This utility will read existing flows in flow.xml.gz and decrypt the sensitive values using the key mentioned in the properties file (or default values if no key value is provided in the properties file). Once decrypted, it encrypts all the values with the new key provided and overwrites the existing flow.xml.gz or create a new one depending on what parameters are passed.

Following are the steps to follow:

First of all, we will need the NiFi toolkit on our system.

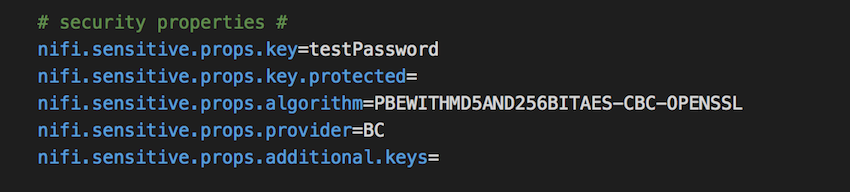
**./encrypt-config.sh -f /path/to/flow.xml.gz -n /path/to/nifi.properties -s newPassword -x**



Migrating the encryption key

Note: All the parameters are self-explanatory in the above command except -x. This is the mandatory parameter in this command as we are telling the utility to encrypt only the sensitive properties in flow.xml.gz and not to decrpyt/encrypt the nifi.properties entries. This is also a good practice to encrypt the property values.

We can verify the result by opening the nifi.properties file where we will find the nifi.sensitive.props.key is set to testpassword which we passed as a parameter of the command line utility.

1. 

nifi.properties file

Using the utility above will result into overwriting the flow.xml.gz and nifi.properties files. But there might be some cases when we need the new files to be written somewhere else and not overwriting the old files. We can use the same command for that but we need to pass two more parameters telling where we want to store the newly created files.

**./encrypt-config.sh -f /path/to/flow.xml.gz -g /path/to/dest/flow.xml.gz -n /path/to/nifi.properties -o /path/to/dest/nifi.properties -s newPassword -x**