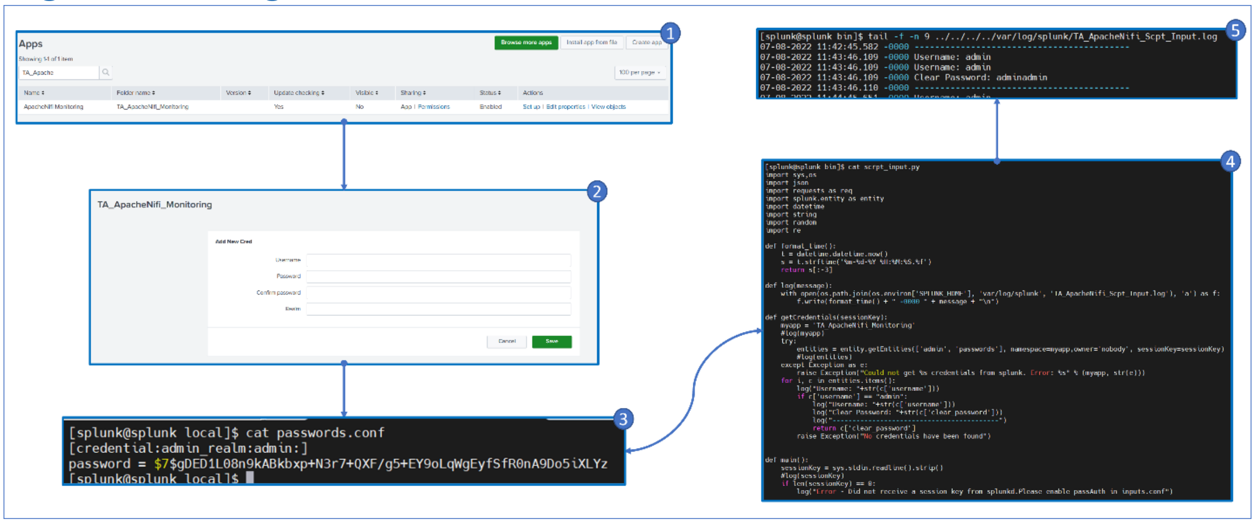
Splunk Scripted Input Technical Addon – With passwords/tokens encryption

Document Guide

End Goal Description

The Goal is to Encrypt the Credentials being used by the scripted input. There is two ways to perform Cred Encryption either though Splunk APIs or through Setup Page for your TA.

High Level Design



Name	Folder name	Version	Update checking	Visible	Sharing	Status	Actions
ApacheNifi Monitoring	TA_ApacheNifi_Monitoring		Yes	No	App | Permissions	Enabled	Set up | Edit properties | View objects

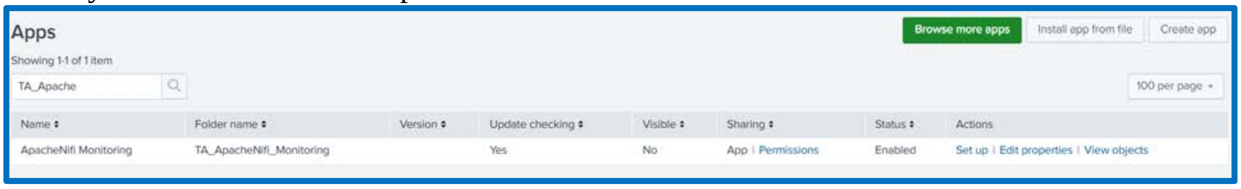
```
[splunk@splunk local]$ cat passwords.conf
[credential:admin_realm:admin:]
password = $7$gDED1L08n9kABkbxp+N3r7+QXF/g5+EY9oLqWgEyfSfR0nA9Do5iXLYz
[splunk@splunk local]$
```

```
[splunk@splunk bin]$ cat scrpt_input.py
import sys,os
import json
import requests as req
import splunk.entity as entity
import datetime
import string
import random
import re

def format_time():
    t = datetime.datetime.now()
    s = t.strftime('%m-%d-%Y %H:%M:%S.%f')
    return s[:-3]

def log(message):
    with open(os.path.join(os.environ['SPLUNK_HOME'], 'var/log/splunk', 'TA_ApacheNifi_Scpt_Input.log'), 'a') as f:
        f.write(format_time() + " -0000 " + message + "\n")

def getCredentials(sessionKey):
    myapp = 'TA_ApacheNifi_Monitoring'
    #log(myapp)
    try:
        entities = entity.getEntities(['admin', 'passwords'], namespace=myapp,owner='nobody', sessionKey=sessionKey)
        #log(entities)
    except Exception as e:
        raise Exception("Could not get %s credentials from splunk. Error: %s" % (myapp, str(e)))
    for i, c in entities.items():
        log("Username: "+str(c['username']))
        if c['username'] == "admin":
            log("Username: "+str(c['username']))
            log("Clear Password: "+str(c['clear_password']))
            log("----------------------------------------")
            return c['clear_password']
        raise Exception("No credentials have been found")


def main():
    sessionKey = sys.stdin.readline().strip()
    #log(sessionKey)
    if len(sessionKey) == 0:
        log("Error - Did not receive a session key from splunkd.Please enable passAuth in inputs.conf")
```

```
[splunk@splunk bin]$ tail -f -n 9 ../../../../var/log/splunk/TA_ApacheNifi_Scpt_Input.log
07-08-2022 11:42:45.582 -0000 ------------------------------------------
07-08-2022 11:43:46.109 -0000 Username: admin
07-08-2022 11:43:46.109 -0000 Username: admin
07-08-2022 11:43:46.109 -0000 Clear Password: adminadmin
07-08-2022 11:43:46.110 -0000 ------------------------------------------
```

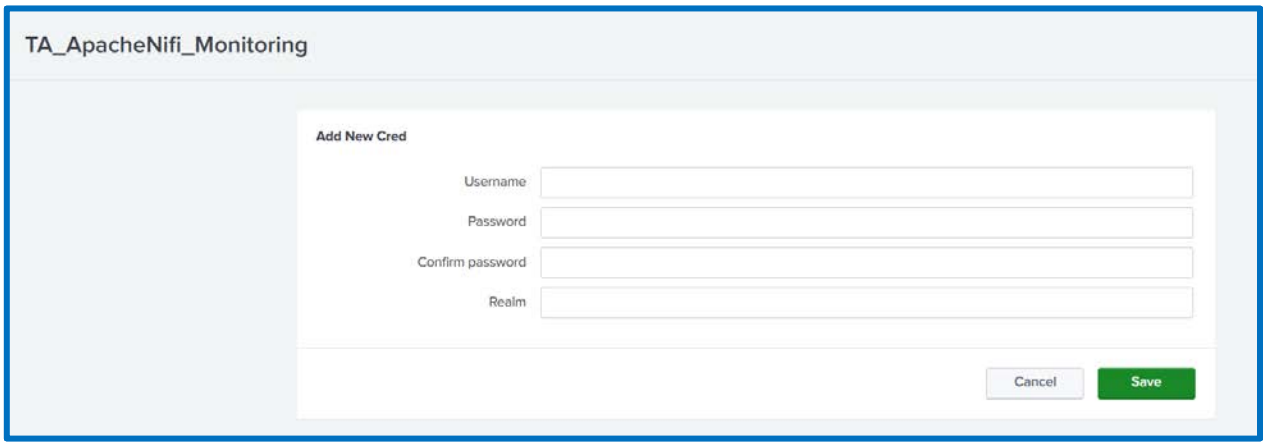
The HLD illustrates how to ensure passwords needed by the scripted input script for API usage are being stored in the platform the TA installed in. Steps are as follows:

1- After TA is being installed go to Splunk Home -> Manage Apps and search for the TA where you have to click in setup as below screen share.

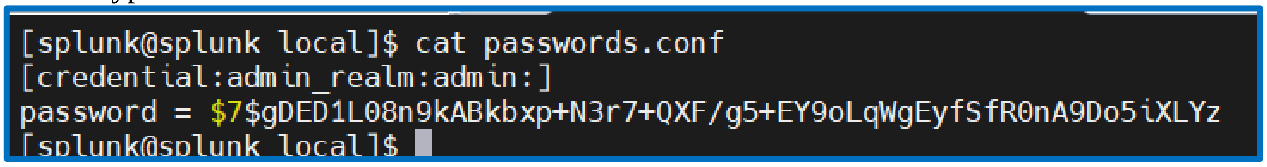


Name	Folder name	Version	Update checking	Visible	Sharing	Status	Actions
ApacheNifi Monitoring	TA_ApacheNifi_Monitoring		Yes	No	App | Permissions	Enabled	Set up | Edit properties | View objects

2- Then Once you get in, you will have a page where will let you input the username, password and realm as below screen share

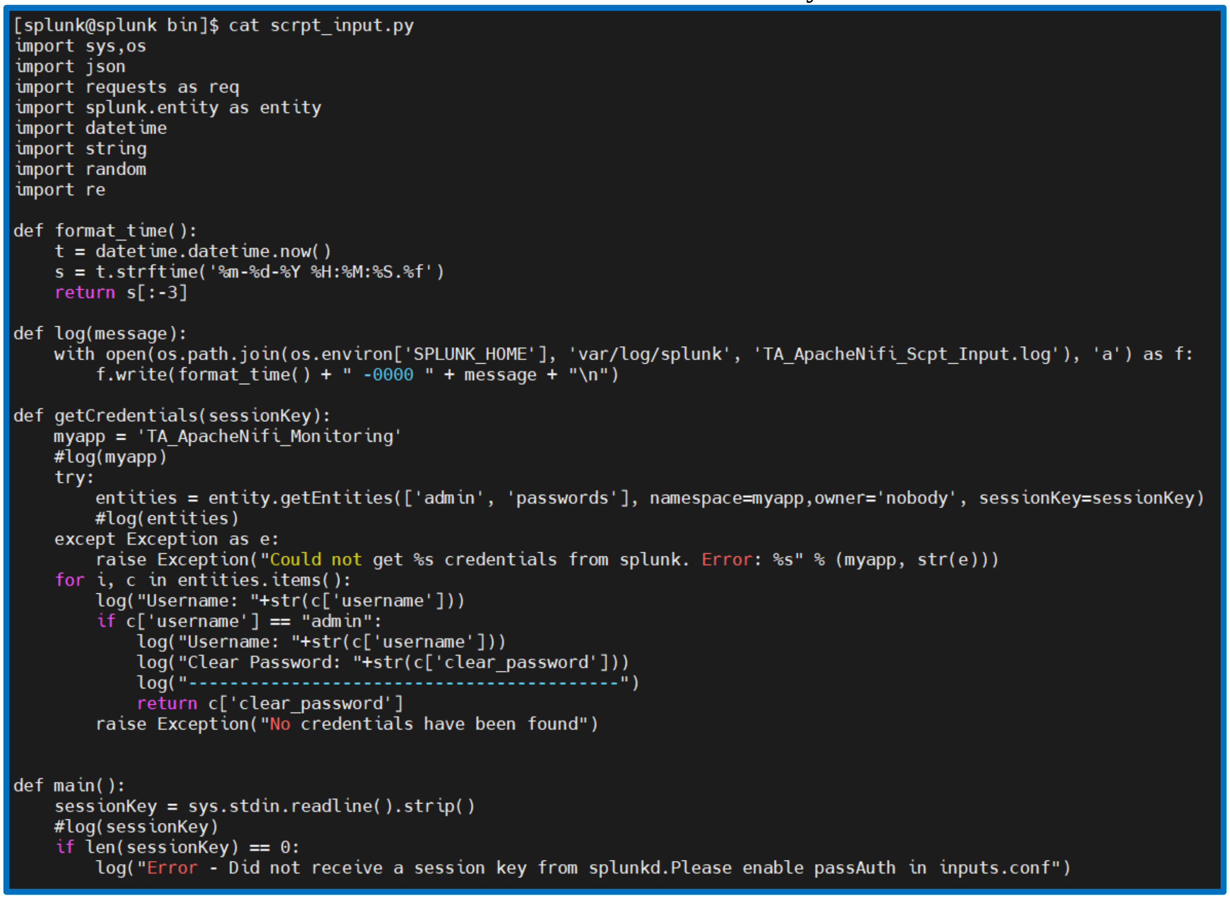


3- Then after saving it, the inputted artifacts will reflect in the background in TA scope in a file call passwords.conf under local dir as follows where you password has been saved and encrypted.



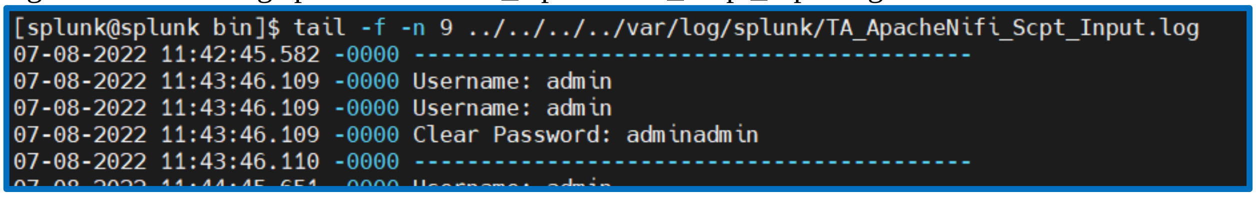
```
[splunk@splunk local]$ cat passwords.conf
[credential:admin_realm:admin:]
password = $7$gDED1L08n9kABkbxp+N3r7+QXF/g5+EY9oLqWgEyfSfR0nA9Do5iXLYz
[splunk@splunk local]$
```

4- When the script will be running, and using the function called getCredentials() will extract the credentials of the needed user in a clear text so that you can use it in API calls.



```
[splunk@splunk bin]$ cat scrpt_input.py
import sys,os
import json
import requests as req
import splunk.entity as entity
import datetime
import string
import random
import re

def format_time():
    t = datetime.datetime.now()
    s = t.strftime('%m-%d-%Y %H:%M:%S.%f')
    return s[:-3]

def log(message):
    with open(os.path.join(os.environ['SPLUNK_HOME'], 'var/log/splunk', 'TA_ApacheNifi_Scpt_Input.log'), 'a') as f:
        f.write(format_time() + " -0000 " + message + "\n")

def getCredentials(sessionKey):
    myapp = 'TA_ApacheNifi_Monitoring'
    #log(myapp)
    try:
        entities = entity.getEntities(['admin', 'passwords'], namespace=myapp,owner='nobody', sessionKey=sessionKey)
        #log(entities)
    except Exception as e:
        raise Exception("Could not get %s credentials from splunk. Error: %s" % (myapp, str(e)))
    for i, c in entities.items():
        log("Username: "+str(c['username']))
        if c['username'] == "admin":
            log("Username: "+str(c['username']))
            log("Clear Password: "+str(c['clear_password']))
            log("------------------------------------------")
            return c['clear_password']
        raise Exception("No credentials have been found")


def main():
    sessionKey = sys.stdin.readline().strip()
    #log(sessionKey)
    if len(sessionKey) == 0:
        log("Error - Did not receive a session key from splunkd.Please enable passAuth in inputs.conf")
```

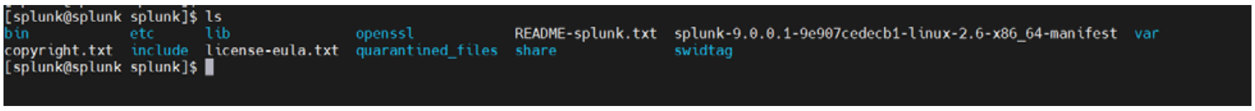
5- As the script above it extracting the password of a user called admin and printing it to a log file under var/log/splunk called TA_ApacheNifi_Scrpt_Input.log as follows:



```
[splunk@splunk bin]$ tail -f -n 9 ../../../../var/log/splunk/TA_ApacheNifi_Scpt_Input.log
07-08-2022 11:42:45.582 -0000 ------------------------------------------
07-08-2022 11:43:46.109 -0000 Username: admin
07-08-2022 11:43:46.109 -0000 Username: admin
07-08-2022 11:43:46.109 -0000 Clear Password: adminadmin
07-08-2022 11:43:46.110 -0000 ------------------------------------------
```

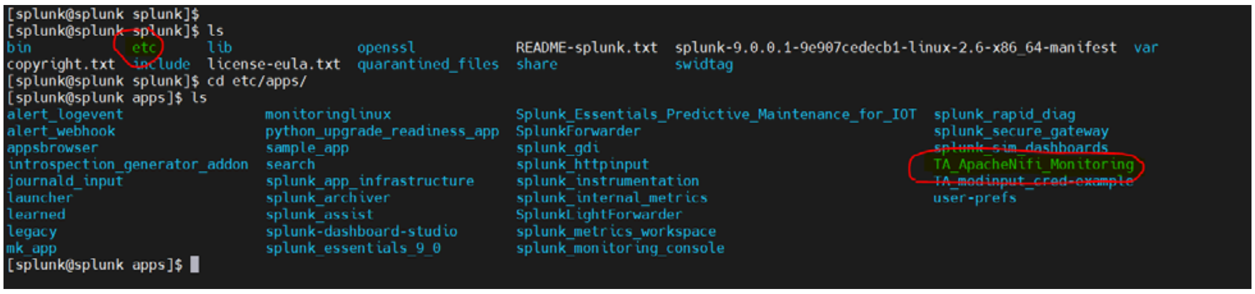
Environment Walkthrough

The current working Environment is a standalone installation of Splunk on a virtual machine.



```
[splunk@splunk splunk]$ ls
bin            etc      lib                openssl            README-splunk.txt  splunk-9.0.0.1-9e907cedecb1-linux-2.6-x86_64-manifest  var
copyright.txt  include  license-eula.txt  quarantined_files  share              swidtag
[splunk@splunk splunk]$
```

The Scripted input technical addon is installed under the etc/apps directory and call TA_ApacheNifi_Monitoring as below Screen Share.

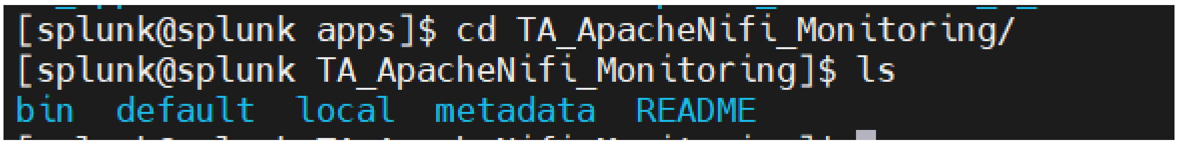


```
[splunk@splunk splunk]$
[splunk@splunk splunk]$ ls
bin            etc      lib                openssl            README-splunk.txt  splunk-9.0.0.1-9e907cedecb1-linux-2.6-x86_64-manifest  var
copyright.txt  include  license-eula.txt  quarantined_files  share              swidtag
[splunk@splunk splunk]$ cd etc/apps/
[splunk@splunk apps]$ ls
alert_logevent                 monitoringlinux               Splunk_Essentials_Predictive_Maintenance_for_IOT  splunk_rapid_diag
alert_webhook                  python_upgrade_readiness_app  SplunkForwarder                                   splunk_secure_gateway
appsbrowser                    sample_app                    splunk_gdi                                        splunk-sim_dashboards
introspection_generator_addon  search                        splunk_httpinput                                  TA_ApacheNifi_Monitoring
journald_input                 splunk_app_infrastructure     splunk_instrumentation                            TA-modinput_cred-example
launcher                       splunk_archiver               splunk_internal_metrics                           user-prefs
learned                        splunk_assist                 SplunkLightForwarder
legacy                         splunk-dashboard-studio       splunk_metrics_workspace
mk_app                         splunk_essentials_9_0         splunk_monitoring_console
[splunk@splunk apps]$
```

TA_ApacheNifi_Monitoring Technical Addon Structure

The Technical addon as below Screen Share contains standard directory structure to contain:

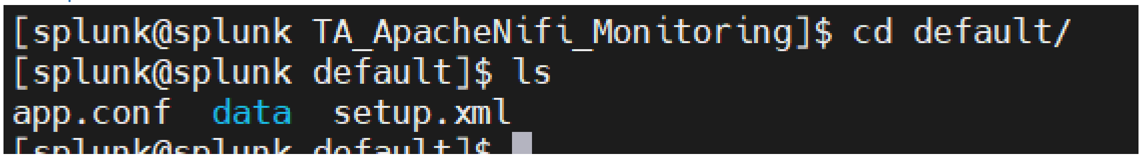
- Bin: to hold the scripted input
- Default: to hold the default configuration files and holds setup.xml as mandatory conf file.
- Local: to hold the customizable configuration files and holds inputs.conf & passwords.conf mandatory for such use-case.
- Metadata: holds the metadata files that defines the access to various abject in the TA



```
[splunk@splunk apps]$ cd TA_ApacheNifi_Monitoring/
[splunk@splunk TA_ApacheNifi_Monitoring]$ ls
bin  default  local  metadata  README
```

Below will tackle all required configuration files / Script Functions needed to perform Password Encryption for your Scripted Input Script.

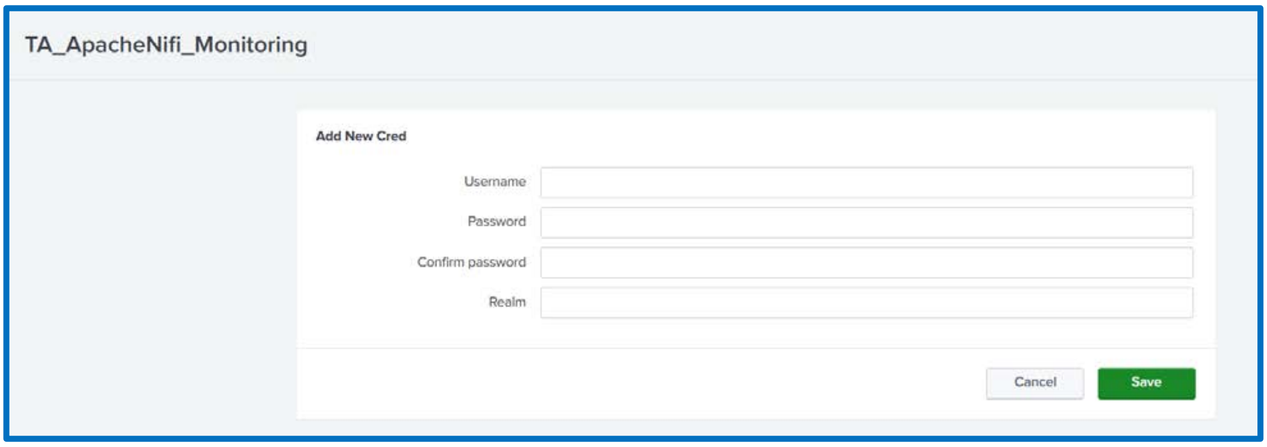
Setup.xml



```
[splunk@splunk TA_ApacheNifi_Monitoring]$ cd default/
[splunk@splunk default]$ ls
app.conf  data  setup.xml
```

Setup.xml file located under default dir and it holds the user interface to be able to input the username, password and realm. To be saved in an encrypted format & being able to retrieve it back in the scripted input script.

The UI should be looking as following:


TA_ApacheNifi_Monitoring

Add New Cred

Username	
Password	
Confirm password	
Realm	

And the code is as follows:

```
<setup>
 <block title="Add New Cred" endpoint="storage/passwords" entity="_new">
  <input field="name">
    <label>Username</label>
    <type>text</type>
  </input>
  <input field="password">
    <label>Password</label>
    <type>password</type>
  </input>
  <input field="realm">
    <label>Realm</label>
    <type>text</type>
  </input>
 </block>
</setup>
```

Scripted Input Script

Is the script that is used to act as an input to Splunk by pulling data remotely from source using for examples APIs. And to be able to do so it needs to use Credentials either Username/Passwords or Tokens.

Using the Endpoint of [Storage/Passwords] and as we stored the passwords encrypted by the Setup.xml Page using the same endpoint. We can retrieve the passwords as clear text after being saved in an encrypted format.

Below the Python Function Snippet Needed to perform that (Full Python Script Provided in the TA Attached):

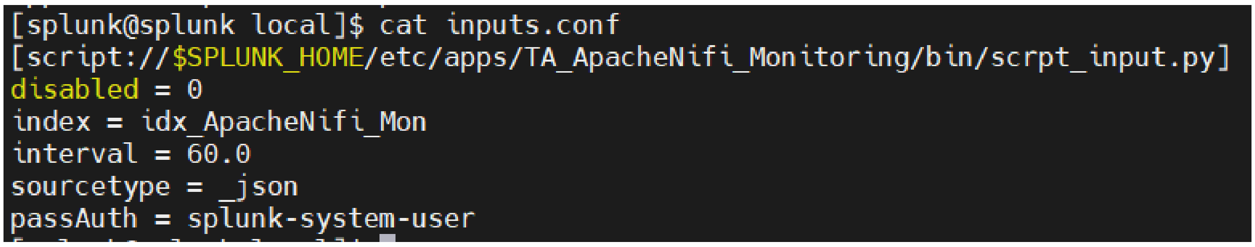
```
import splunk.entity as entity
sessionKey = sys.stdin.readline().strip()
def getCredentials(sessionKey):
    myapp = 'TA_ApacheNifi_Monitoring'
    try:
        entities = entity.getEntities(['admin', 'passwords'], namespace=myapp,owner='nobody',
sessionKey=sessionKey)
    except Exception as e:
        raise Exception("Could not get %s credentials from splunk. Error: %s" % (myapp, str(e)))
    for i, c in entities.items():
        if c['username'] == "admin":
            return c['clear_password']
        raise Exception("No credentials have been found")
```

Inputs.conf

Its the file where we define the scripted input running properties like:

- Running frequency
- The data will be ingested in which index
- The Sourcetype name
- **Most Important : passAuth** > As it will enable the script once running to pass the sessionKey used by the script to run as an standard input to the script. Where sessionKey is used to be able to retrieve the password.

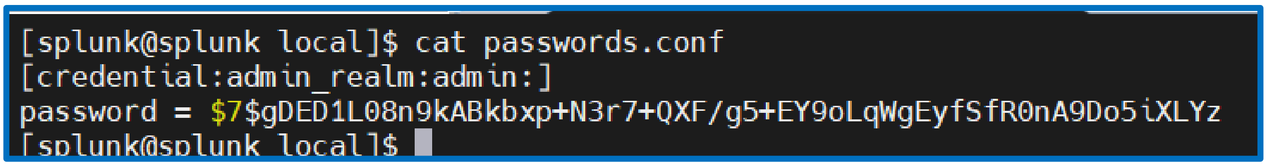
Below is an example of the inputs.conf



```
[splunk@splunk local]$ cat inputs.conf
[script://$SPLUNK_HOME/etc/apps/TA_ApacheNifi_Monitoring/bin/scrpt_input.py]
disabled = 0
index = idx_ApacheNifi_Mon
interval = 60.0
sourcetype = _json
passAuth = splunk-system-user
```

Passwords.conf

Passwords.conf is the file that holds the artifacts of the creds that are being filled in the setup page. This file is being automatically created and filled by the setup.xml. If filled manually that will not reflect into password to be encrypted.



```
[splunk@splunk local]$ cat passwords.conf
[credential:admin_realm:admin:]
password = $7$gDED1L08n9kABkbxp+N3r7+QXF/g5+EY9oLqWgEyfSfR0nA9Do5iXLYz
[splunk@splunk local]$
```

How to?

To have this successfully configured, and based-on the understanding of the above please follow the below steps:

1- Install (Untar/Create) Scripted Input TA in etc/apps under Splunk Instance
2- create setup.xml as mentioned above
3- create inputs.conf and make sure to add the [passAuth = splunk-system-user] in the scripted input stanza.
4- Within the script use the Python Code Snippet above and don't forget to change the myapp variable to your app's name. And use the function return value for the clear text password.

Adoption of different Deployment Options

If you are using an environment where it will be hard to access UI of the Splunk instance the TA installed on. So, you will not be able to update the passwords.conf using the setup page. Here is a Splunk's API that can perform the same functionality of the setup page.

```
curl -k -u <username>:<password> https://localhost:8089/servicesNS/nobody/<APP NAME>/storage/passwords -d name=<UserName> -d password=<Password> -d realm=<Realm>
```

References & Resources

- [Manage secret storage using the Splunk SDK for Python | Documentation | Splunk Developer Program]()

- Manage secret storage using the Splunk REST API | Documentation | Splunk Developer Program
- Access endpoint descriptions - Splunk Documentation
- Moh-Ayman/Splunk-TA_Scipted_Input_PassEnrypt: Splunk Scripted Input technical Addon illustration on how to Encrypt Passwords/Token used by the sripts. (github.com)