1-RetrieveOneViewToken

September 7, 2021

1 Retrieve an HPE OneView authentication token to access a managed iLO 5

Version 0.20

1.1 Introduction

This notebook uses the HPE OneView Ansible Galaxy collection to connect to an HPE OneView appliance, retrieves properties of a server that is managed by this appliance as well as a security token for that server's iLO. This token will be used in subsequent sections of the lab to connect to the iLO without authenticating again, using the Single Sign On (SSO) feature of HPE OneView and iLO.

For the sake of transparency, the unabated truth regarding this lab is that you will be using simulated hardware. You are going to connect to an HPE OneView appliance containing code that simulates responses from managed hardware. While these simulated responses allow one to obtain an iLO security token, they do not fully simulate an iLO. So, in subsequent sections of this lab, you will be using another simulator that fully implements the Redfish API like an iLO. The two simulators are independent of each other. In other words the token you are going to obtain in this section cannot be used in the following sections. You will just have to trust us that it really works this way with real hardware. Yes, we've tested it!

> NOTE: At the time of the writing, OneView does not support SSO against Superdome Flex servers. Hence, this notebook is valid for iLO 5 based servers only.

1.2 Environment preparation

[1]:
```
########## Environment preparation #############

# Set Student ID number
export stdid=776
Id=$(id --user --name)
NbId=1

# OneView variables
OneViewSimulatorIP=16.31.86.247

cat > ${NbId}/config.json << __EoF__
{
```

```
    "ip": "${OneViewSimulatorIP}",
    "credentials": {
      "userName": "Administrator",
      "authLoginDomain": "",
      "password": "password"
    },
    "api_version": 1800
  }
__EoF__

echo "You are ready to continue...."
```

```
You are ready to continue…
```

First, we need to install the HPE OneView Ansible Galaxy collection. This is simply done with the following command:

[2]:
```
/opt/jupyterhub/bin/ansible-galaxy collection install hpe.oneview
```

```
Starting galaxy collection install process
Process install dependency map
Starting collection install process
Installing 'hpe.oneview:6.2.0' to
'/student/student776/.ansible/collections/ansible_collections/hpe/oneview'
Downloading https://galaxy.ansible.com/download/hpe-oneview-6.2.0.tar.gz to
/student/student776/.ansible/tmp/ansible-local-43065zd8fkwh/tmpnm478tcg
hpe.oneview (6.2.0) was installed successfully
```

The collection is installed (by default) in ~/.ansible/collections. We also need to look at the requirements for this collection, as they are not installed by the `ansible-galaxy collection install` command.

[3]:
```
cat ~/.ansible/collections/ansible_collections/hpe/oneview/requirements.txt
```

```
hpICsp
hpeOneView
```

These are Python packages to be installed with `pip`. You can safely ignore `hpICsp`, which is no longer used. The `ansible` package is kind of obvious and is already installed. The `hpeOneView` package represents the Python OneView SDK, which has already been installed in your environment as well.

We are now ready to run a playbook to get a security token for an iLO of a managed server. Before running it, let's look at it. Open the playbook, it will open in a new tab. Come back to this tab for explanations. The playbook starts with these lines:

```
- hosts: localhost
  gather_facts: false
  collections:
    - hpe.oneview
```

The playbook always runs on `localhost`. This is not the traditional Ansible way where Ansible will connect to various hosts to perform its tasks. HPE OneView is an appliance and you cannot ssh into it. Hence, you cannot have an HPE OneView appliance in an Ansible hosts' inventory. This would not work. Instead, the Ansible OneView collection runs locally but connects to the REST API of an HPE OneView appliance specified in a config file we will look at in a minute.

We are not particularly interested in the details of the host we are running the playbook on, so `gather_facts: false` allows to speed up the execution a bit.

And finally we declare the collection we are going to use in this playbook.

Next is our first task:

```
- name: Get server
  oneview_server_hardware_facts:
    config: "config.json"
    name: "0000A66101, bay 7"
    options:
    - remoteConsoleUrl: true
```

`oneview_server_hardware_facts` is the role we want to use from our Ansible Galaxy collection. It essentially translates to a REST API call to HPE OneView with a GET /rest/server-hardware.

`config: "config.json"` is where we define which HPE OneView appliance we want to connect to. You can open this JSON file. It simply contains the IP address and credentials of an HPE OneView appliance (there are ways to avoid having a password in clear text like this, but they are beyond the scope of this lab), and the API version we want to use. Each version of OneView implements a version of the REST API (and a number of previous versions for backwards compatibility). See the About page of the OneView REST API Reference.

`name: "0000A66101, bay 7"` is the name of the server we want to access.

`options: - remoteConsoleUrl: true` says that we want to retrieve in our GET call the remoteConsoleUrl attribute that contains the iLO token we want.

Now we want to parse the results of our REST API call to extract just the bit we are interested in. It is in the session key parameter in the remoteConsoleUrl, and we store it in the `iloToken` variable:

```
- set_fact:
    iloToken: "{{ server_hardware_remote_console_url.remoteConsoleUrl | regex_search(regexp,
  vars:
    regexp: 'sessionkey=([0-9a-f]+)'
```

And finally we print the result:

```
- name: print result
  debug:
    var: iloToken
```

Now you can run the playbook:

[4]:
```
/opt/jupyterhub/bin/ansible-playbook 1/GetiLOSSOToken.yml
```

```
[WARNING]: provided hosts list is empty, only localhost is available. Note that
the implicit localhost does not match 'all'

PLAY [localhost] ***************************************************************

TASK [Get server] **************************************************************
ok: [localhost]

TASK [set_fact] ****************************************************************
ok: [localhost]

TASK [print result] ************************************************************
ok: [localhost] => {
    "iloToken": [
        "3b3a0c95f0115a26e80627604ddf02bb"
    ]
}

PLAY RECAP *********************************************************************
localhost                  : ok=3    changed=0    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
```

You can now move to the next lab and start talking Redfish to an iLO.