NetOps Password Ansible
User Guide

Overview

This Ansible enables you to update the data repository, NetOps Portal MySQL, and DX NetOps Spectrum passwords.

The Ansible playbook follows this basic flow:

1. Shut down the component services.
2. Update the database password.
3. Update the database password in component configuration files.
4. Start the component services.
5. Perform a basic health check of the component.

Important! If you copy-paste the Ansible commands from this document, replace the single dash or dashes that prefix the `tags` parameter, as autocorrect tends to break those dashes, which can cause problems. Delete the dashes for the `tags` parameter, and then re-add those dashes.

Prerequisites

Inventory

An inventory must be provided to Ansible.
For more information about the format of the inventory host groups, see the `inventory.example` file.

Variables

Defaults are provided for many of the configurable options of the Ansible Playbook. Prior to executing the Ansible playbook, refer to the `group_vars/all` file, and make any changes according to the specifics of your DX NetOps deployment.

Important! Data Aggregator and Data Repository related variables defined in the group_vars/all file have been updated for consistency with the rest of the Ansible playbook and roles.
Variables formerly prefixed with “dataaggregator_” are now prefixed with “data_aggregator_”.
Variables formerly prefixed with “datarepository_” are now prefixed with “data_repository_”.

Portal/MySQL

1. Stop the NetOps Portal services by issuing the following commands: (begin Portal outage)
   ansible-playbook -i inventory --tags portal_stop update-netops-passwords.yml

2. Update the passwords in the NetOps Portal MySQL instance by issuing the following commands:
   ansible-playbook -i inventory -e <vars…> --tags portal_mysql_update_password update-netops-passwords.yml

   This procedure applies the password change in the MySQL database. Success indicates that CyberArk can commit the new password; failure indicates that the new password could not be applied.

   Note: NetOps Portal requires the MySQL 'root' user password and 'netqos' user password to match. Therefore, this procedure applies the new password to both the 'root' user and the 'netqos' user in MySQL.

   vars:

   ```
   portal_mysql_password="<current password>"
   portal_mysql_password_new="<new password>"
   ```

   Note: This task does some initial validation of the passwords you have supplied to make sure that the Ansible can connect to the database using those credentials. Both the current password and the new password are used (in sequence) to determine which password is currently in use. You will see a failure in the Ansible output (red text) for one or both of the passwords. If both passwords fail, the overall procedure will fail. If one of the passwords succeeds, the procedure will continue on and overall success/failure of the procedure will be indicated by the exit code of the Ansible command.

3. Update the passwords in NetOps Portal configuration files by issuing the following command:
   ansible-playbook -i inventory -e <vars…> --tags portal_update_password update-netops-passwords.yml

   This procedure applies the new MySQL password to the NetOps Portal configuration files.

   Issue this command once with the new MySQL password.

   vars:

   ```
   portal_mysql_password_new="<new password>"
   ```

4. Start NetOps Portal services by issuing the following command:
   ansible-playbook -i inventory --tags portal_start update-netops-passwords.yml

5. Perform a basic health check to verify that NetOps Portal is running by issuing the following command:
   ansible-playbook -i inventory --tags portal_health_check update-netops-passwords.yml

Data Aggregator/Vertica

1. Stop the Data Aggregator services by issuing the following command:
   ansible-playbook -i inventory -e <vars…> --tags data_aggregator_stop update-netops-passwords.yml

   This procedure connects to the Vertica database as the `dauser` to verify the integrity of the ETL tables prior to shutting down the data aggregator. As a result, you must provide the `data_repository_da_password` variable. This is the current password for the `dauser`.

   vars:

   ```
   data_repository_da_password='<current vertica password for the 'dauser'>'
   ```

2. Update the passwords in the data repository (Vertica database) by issuing the following command:
   ansible-playbook -i inventory -e <vars…> --tags data_repository_update_password update-netops-passwords.yml

   This procedure is responsible for applying the password change in the Vertica database. Success indicates that CyberArk can commit the new password; failure indicates that the new password could not be applied.

   This procedure accepts a list of one or more user credentials to be updated in the Vertica database.

   CyberArk can call this procedure twice: once for the `dradmin` user password, and once for the `dauser` user password by passing the credentials for each user separately to each invocation of the procedure.

   vars:

```
data_repository_credentials='[{"user":"<dradmin or
dauser>","password":"<current
password>","password_new":"<new password>"}]'
```

3. Update the passwords in the data aggregator configuration files by issuing the following command:
   ansible-playbook -i inventory -e <vars…> --tags data_aggregator_update_password update-netops-passwords.yml

   This procedure applies the password change to the data aggregator configuration files, and accepts a list of one or more user credentials to be updated in the configuration files. Issue this command for each user (`dradmin` and `dauser`) when their credentials have changed in the data repository (Vertica). The data aggregator requires the latest credentials for both users.

   You can issue this command with a list of credentials (`dradmin`, `dauser`), or you can issue the command separately, once for each user (`dradmin`, `dauser`).

   vars:
   ```
   data_repository_credentials='[{"user":"<dradmin or
   dauser>","password":"<current
   password>","password_new":"<new password>"}]'
   ```

4. Start NetOps Data Aggregator services by issuing the following command:
   ansible-playbook -i inventory --tags data_aggregator_start update-netops-passwords.yml

5. Update the ActiveMQ JMX password on Data Aggregator and Data Collectors by issuing the following command:
   ansible-playbook -i inventory --tags activemq_update_jmx_password -e data_aggregator_rest_user=<DA_USER> -e data_aggregator_rest_password=<DA_PASSWORD> -e jmx_password=<EXISTING_JMX_PASSWORD> -e jmx_password_new=<NEW_JMX_PASSWORD> update-netops-passwords.yml

6. Perform a basic health check to verify that NetOps Data Aggregator is running by issuing the following command:
   ansible-playbook -i inventory --tags data_aggregator_health_check update-netops-passwords.yml

Spectrum

Prerequisites

Prior to executing the Ansible Playbook:

Update the `spectrum_install` and `spectrum_install_owner` variables in the `group_vars/all` file according to the specifics of your DX NetOps deployment.

Execution

1. Perform an initial integrity check by issuing the following command:
   `ansible-playbook -i inventory --tags spectrum_init update-netops-passwords.yml`

   This procedure connects to the Spectrum server, and then performs initial/integrity checks for the following:
   a. The `update_mysql_pwd.pl` script exists or not?
   b. Does the `custom_mysql_config_editor` binary exist or not?
   c. Existing root have access to the DB using the existing MySQL encrypted file or not?
      If root access fails, the playbook can perform corrective action when the `spectrum_root_password` or `spectrum_root_password_new` variable is provided:
      - Verify that the `spectrum_root_password` variable has access to the DB. If yes, update the configuration file with the accessible password.
      - Verify that the `spectrum_root_password_new` variable has access to the DB. If yes, update the configuration file with the accessible password.

2. Stop the Services by issuing the following command:
   `ansible-playbook -i inventory --tags spectrum_stop update-netops-passwords.yml`

   This procedure connects to the Spectrum server, and then stops the Spectrum services for example Tomcat server and Archive manager based on the Spectrum installation.

3. Update the password:

   This procedure is responsible for applying the password change in the MySQL database. Success indicates that CyberArk can commit the new password; failure indicates that it cannot apply the new password.

You can update one or more user passwords at the same time using the `ansible-playbook`.

Spectrum uses the MySQL encrypted login-path option to connect to MySQL server. It does not require that you specify the old password.

In the following commands, the `spectrum_update` tag does the following:

a. Verifies if the new password has been updated in the configuration file or not?
b. Updates the MySQL encrypted configuration file.
c. Updates the MySQL DB password
d. Verifies the configuration files have DB access with a given user.

The following commands are examples of how to update the password for the corresponding users.

Single user only:

a. root user:

```
ansible-playbook -i inventory --tags
spectrum_update_mysql_password -e '{
spectrum_root_password_new : 123456}'
update-netops-passwords.yml
```

The `spectrum_root_password_new` variable is used to update the new password for the root user.

b. OC_user user:

```
ansible-playbook -i inventory --tags
spectrum_update_mysql_password -e '{
spectrum_OC_user_password_new : 123456}'
update-netops-passwords.yml
```

The `spectrum_OC_user_password_new` variable is used to update the new password for the OC_user user.

c. OC_admin user:

```
ansible-playbook -i inventory --tags
spectrum_update_mysql_password -e '{
spectrum_OC_admin_password_new : 123456}'
update-netops-passwords.yml
```

The `spectrum_OC_admin_password_new` variable is used to update the new password for the OC_admin user.

d. SPEC_admin user:
```
ansible-playbook -i inventory --tags
spectrum_update_mysql_password -e '{
spectrum_SPEC_admin_password_new : 123456}'
update-netops-passwords.yml
```

The `spectrum_SPEC_admin_password_new` variable is used to update the new password for the SPEC_admin user.

e. CR_user user:
```
ansible-playbook -i inventory --tags
spectrum_update_mysql_password -e '{
spectrum_CR_user_password_new : 123456}'
update-netops-passwords.yml
```

The `spectrum_CR_user_password_new` variable is used to update the new password for the CR_user user.

f. Multiple Users:
```
ansible-playbook -i inventory --tags
spectrum_update_mysql_password -e '{
spectrum_root_password_new : 123456
,spectrum_SPEC_admin_password_new : 123456 ,
spectrum_CR_user_password_new : 123456 }'
update-netops-passwords.yml
```

4. Validate by issuing the following command:
```
ansible-playbook -i inventory --tags spectrum_validate -e '{
spectrum_root_password_new : 123456
,spectrum_SPEC_admin_password_new : 123456 }'
update-netops-passwords.yml
```

This procedure is to validate provided MySQL passwords are set as expected.

5. Start the Spectrum Services by issuing the following command:
```
ansible-playbook -i inventory --tags spectrum_start
update-netops-passwords.yml
```

This procedure connects to the Spectrum server, and then starts the Spectrum services, for example, the Tomcat server and Archive manager based on the Spectrum installation.

6. Perform a health check by issuing the following command:

ansible-playbook -i inventory --tags spectrum_health_check update-netops-passwords.yml

This procedure performs a health check to verify if the Spectrum services are up and running or not.

Virtual Network Assurance

Update VNA MySql password:

Prerequisites

Prior to executing the Ansible Playbook:

Update the `vna_install` and `vna_install_owner vna_mysql_install` variables in the `group_vars/all` file according to the specifics of your DX NetOps deployment.

Default values:

vna_install - /opt/CA/VNA
vna_install_owner - root
vna_mysql_install - /opt/CA/MySql

This procedure is responsible for applying the password change in the MySQL database. Success indicates that CyberArk can commit the new password; failure indicates that it cannot apply the new password.

You can update one or more user passwords at the same time using the `ansible-playbook`.

The following command the `vna_set_mysql_password` tag does the following:

a. Performs basic validations and skip if validation passed
b. Validates the user permissions to update the password and configurations
c. Validates the status of wildfly server
d. Verifies if the new password has been updated in the configuration file or not?
e. Updates the MySQL DB password
f. Updates the JBoss configurations with new password
g. Reverts back to old password if failed to update jboss configuration
h. Restart wildfly service

1. Update the MySql database password in the Virtual Network Assurance configuration files by issuing the following command:
   ansible-playbook -i inventory.example --tags
   `vna_set_mysql_password -e vna_mysql_root_user=<DB_ROOT_USER> -e vna_mysql_root_password_new=<NEW_PASSWORD> -e vna_mysql_root_password=<EXISTING_PASSWORD>`
   update-netops-passwords.yml

   `Note: Vna_mysql_root_user is optional and default is "root" if not specified.`

Update VNA Rest password:

Prerequisites

Prior to executing the Ansible Playbook:

Update the `vna_install`, `vna_install_owner, data_aggregator_rest_scheme, and data_aggregator_rest_port` variables in the `group_vars/all` file according to the specifics of your DX NetOps deployment.

Default values:

vna_install - /opt/CA/VNA
vna_install_owner - root
data_aggregator_rest_scheme - http
Data_aggregator_rest_port - 8581

This procedure is responsible for changing and applying the VNA Rest password.. Success indicates that CyberArk can commit the new password; failure indicates that it cannot apply the new password.

You can update one or more user passwords at the same time using the `ansible-playbook`.

The following command the `vna_set_rest_password` tag does the following:

a. Performs basic validations and skip if validation passed
b. Validates the user permissions to update the password and configurations
c. Validates the status of wildfly server
d. Updates the VNA Rest login password
e. Updates the JBoss configurations with new password
f. Closes the existing client connections created with old password
g. Restart wildfly service

Update the Rest password in the Virtual Network Assurance configuration files by issuing the following command:

```
ansible-playbook -i inventory.example --tags
vna_set_rest_password  -e
spectroserver_host_name=<SPECTROSEVER_HOSTNAME> -e
vna_host_name=<VNA_HOSTNAME> -e data_aggregator_host_name
=<DA_HOSTNAME> -e data_aggregator_rest_user=<DA_USER> -e
data_aggregator_rest_password=<DA_PASSWORD> -e
vna_rest_password=<EXISTING_REST_PASSWORD> -e
vna_rest_password_new=<NEW_REST_PASSWORD>
update-netops-passwords.yml
```

Note:
Use the above full configuration when VNA is integrated with PM and Spectrum. Also, use DA proxy hostname for `data_aggregator_host_name`, if PM is configured with High Availability(HA).

Exclude the below ansible extra parameter, If VNA and Spectrum integration is not enabled

```
-e spectroserver_host_name=<SPECTROSEVER_HOSTNAME>
```

Exclude the below ansible extra parameters, If VNA and PM integration is not enabled

```
-e vna_host_name=<VNA_HOSTNAME>
-e data_aggregator_host_name =<DA_HOSTNAME>
-e data_aggregator_rest_user=<DA_USER>
-e data_aggregator_rest_password=<DA_PASSWORD>
```

Network Flow Analysis

Prerequisites

Prior to executing the Ansible Playbook:

Update the `nfa_user` and `nfa_new_password` variables in the `group_vars/all` file according to the specifics of your DX NetOps deployment. Also update `nfa_installnfa`, `nfa_install_owner` and `nfa_windows_install` if required.

Note: Since NFA Console is installed in Windows, windows host setup will be required for ansible control node to communicate with it. Please follow https://docs.ansible.com/ansible/latest/user_guide/windows_setup.html for the same. Based on

the host setup, ansible properties can be adjusted in `[nfa_windows_server:vars]` in `inventory.example`.

Terminology

Linux inventories refers to NFA linux harvesters. Windows inventories can have Harvester or Console or both. Depending on the installation, they will be considered for password updates for windows inventories. If both (Harvester and Console) are present, both will be considered for password updates.

Additional Note: Linux specific properties `nfa_install` and `nfa_install_owner` etc will be considered for all the linux inventories and similarly windows specific properties `nfa_windows_install` etc will be considered for all the windows inventories.

Execution

Following commands can be used to run ansible playbook for nfa.

#Run ansible playbook for complete NFA (it includes both windows and linux inventories)

```
ansible-playbook   -i inventory.example  --tags nfa
update-netops-passwords.yml
```

#Run ansible playbook targeting only nfa linux hosts

```
ansible-playbook   -i inventory.example  --tags nfa_linux
update-netops-passwords.yml
```

#Run ansible playbook targeting only nfa windows hosts

```
ansible-playbook   -i inventory.example  --tags nfa_windows
update-netops-passwords.yml
```

#Run ansible playbook targeting only nfa linux hosts and specific operations(i.e. start, stop, set_passwords, validate etc)

```
ansible-playbook   -i inventory.example  --tags nfa_linux_stop
update-netops-passwords.yml
```

Following section covers more details about granular operations that are supported for Windows and Linux hosts and provides commands for the same in detail.

NFA Linux Harvester Hosts

1. Perform an initial integrity check by issuing the following command:
   ansible-playbook -i inventory.example --tags nfa_linux_init update-netops-passwords.yml

   This procedure connects to the NFA linux harvester servers, and then performs initial/integrity checks for the following:
   a. Check if the NFA harvester is installed or not?
   b. Check if harvester-services.sh binary exists or not?
   c. Check if DBPasswordUtil binary exists or not?

2. Stop the Services by issuing the following command:
   ansible-playbook -i inventory.example --tags nfa_linux_stop update-netops-passwords.yml

   This procedure connects to the NFA Linux Harvester server, and then stops the services for example harvester, poller, reaper and collpollws etc.

3. Update the password:

   This procedure is responsible for applying the password change in the MySQL database. Success indicates that CyberArk can commit the new password; failure indicates that it cannot apply the new password.

You can update one user password at a time using the `ansible-playbook`.
NFA does not require that you specify the old password.
In the following commands, the `nfa_linux_set_passwords` tag does the following:

a. Updates mysql user password with the new password.
b. Updates `<NFA_Installed_Path>/DBUsers/ReporterAnalyzer.ini` file with new password.

The following commands are examples of how to update the password for the corresponding users.

Single user only:

- root user:

```
ansible-playbook -i inventory.example --tags
nfa_linux_set_passwords -e '{ nfa_user : root
,nfa_new_password : nfa12345 }' update-netops-passwords.yml
```

- netqos user:

```
ansible-playbook -i inventory.example --tags
nfa_linux_set_passwords -e '{ nfa_user : netqos
,nfa_new_password : nfa12345 }' update-netops-passwords.yml
```

4. Validate by issuing the following command:

```
ansible-playbook -i inventory.example --tags nfa_linux_validate
-e '{ nfa_user : netqos ,nfa_new_password : nfa12345 }'
update-netops-passwords.yml
```

This procedure is to validate whether the MySQL password is correctly updated or not.

5. Start the NFA Linux Services by issuing the following command:

```
ansible-playbook -i inventory.example --tags nfa_linux_start
update-netops-passwords.yml
```

This procedure connects to the NFA linux harvester servers, and then starts the NFA services, for example, harvester,reaper, poller and collpollws etc.

NFA Windows Hosts

Same commands can be run for NFA windows hosts as well with following changes in the tags.

1. Perform an initial integrity check by issuing the following command:
   ansible-playbook -i inventory.example --tags nfa_windows_init update-netops-passwords.yml

   This procedure connects to the NFA windows servers, and then performs initial/integrity checks for the following:
   - Check if NFA is installed or not?
   - Check if required parameters are present while running playbook or not?

   For ex `nfa_user` and `nfa_new_password`.

2. Stop the Services by issuing the following command:
   ansible-playbook -i inventory.example --tags nfa_windows_stop update-netops-passwords.yml

   This procedure connects to the NFA Windows server, and then stops the services for harvester and console based on the NFA setup.

3. Update the password:

   This procedure is responsible for applying the password change in the MySQL database. Success indicates that CyberArk can commit the new password; failure indicates that it cannot apply the new password.

   You can update one user password at a time using the `ansible-playbook`.
   NFA does not require that you specify the old password.
   In the following commands, the `nfa_windows_set_passwords` tag does the following:
   a. Updates mysql user password with the new password.
   b. Updates `<NFA_Installed_Path>/DBUsers/ReporterAnalyzer.ini` file with new password.
   c. If the user is netqos, password will also be updated in `<NFA_Installed_Path>\Portal\SSO\webapps\sso\WEB-INF\sso.properties`.

The following commands are examples of how to update the password for the corresponding users.

Single user only:

- root user:

```
ansible-playbook -i inventory.example --tags
nfa_windows_set_passwords -e '{ nfa_user : root
,nfa_new_password : nfa12345 }' update-netops-passwords.yml
```

- netqos user:

```
ansible-playbook -i inventory.example --tags
nfa_windows_set_passwords -e '{ nfa_user : netqos
,nfa_new_password : nfa12345 }' update-netops-passwords.yml
```

4. Validate by issuing the following command:

```
ansible-playbook -i inventory.example --tags nfa_windows_validate
-e '{ nfa_user : netqos ,nfa_new_password : nfa12345 }'
update-netops-passwords.yml
```

This procedure is to validate whether the MysQL password is correctly updated or not.

5. Start the NFA Linux Services by issuing the following command:

```
ansible-playbook -i inventory.example --tags nfa_windows_start
update-netops-passwords.yml
```

This procedure connects to the NFA windows servers, and then starts the NFA services harvester and console based on the NFA setup configuration.

6. Do iisreset once password is updated.

```
ansible-playbook -i inventory.example --tags iisreset
update-netops-passwords.yml
```