Example cronjob

1. **User Context:** The `crontab -l` output shows that these jobs run as user, `flengyel`. For the system to work, the `flengyel` user must be a member of the `appsecretaccess` group and must have logged in at least once since being added to that group.
2. **Execution Flow:** The execution flow is `cronjob -> wrapper_script.sh -> python_script.py`. The wrapper script sets up the environment (`LD_LIBRARY_PATH`, virtual environment activation) before executing the main logic.
3. **The Target:** The `cunyfirst.py` script, with its hardcoded line `constr = 'oracle+cx_oracle://flengyel:this is a big secret' + datasrc`, is the part we will now secure.

The wrapper script, `dailyoim_venv.sh`, needs **no changes**. Its job is to set up the environment, which it already does correctly.

How to Modify `cunyfirst.py` to Use the Credential Store

We will modify `cunyfirst.py` to call the reusable Python helper module (`secret_retriever.py`) to fetch the Oracle password at runtime.

Here is the "before and after" of your `cunyfirst.py` script.

Original `cunyfirst.py` (Relevant Part)

```
# ... imports ...
dbhost = 'IAMPRDDB1.cuny.edu'
dbport = '2483'
dbsid  = 'PDIMOIG_HUD'
datasrc = cx.makedsn(dbhost, dbport,service_name = dbsid)
# This line contains the hardcoded secret password
constr  = 'oracle+cx_oracle://flengyel:this is a big
secret' + datasrc

engine = sqlalchemy.create_engine(constr,
max_identifier_length=128)
# ... rest of script ...
```

Revised `cunyfirst.py` (Secure Version)

This version dynamically fetches the password from the credential store.

```
#!/usr/bin/env python3

import csv
import sqlalchemy
import cx_Oracle as cx
import sys

# --- Start of modifications for secure password handling
---

# Add the shared helper library's parent directory to
```

```
Python's path
# This makes the 'import secret_retriever' call work
reliably in cron.
HELPER_LIB_PATH = "/usr/local/lib/ansible_secret_helpers"
if HELPER_LIB_PATH not in sys.path:
    sys.path.append(HELPER_LIB_PATH)

try:
    import secret_retriever
    except ImportError:
# A clear error message is crucial for debugging cronjobs
        print(f"CRITICAL ERROR: Could not import
secret_retriever from {HELPER_LIB_PATH}. Check path and
permissions.", file=sys.stderr)
        sys.exit(1)

try:
# Retrieve the 'oracle_db' password using the helper
function
    oracle_password =
secret_retriever.get_password("oracle_db")
    if not oracle_password:
        raise RuntimeError("Retrieved empty password from
credential store.")
except Exception as e:
    print(f"CRITICAL ERROR: Failed to retrieve Oracle DB
password. Error: {e}", file=sys.stderr)
    sys.exit(1)

# --- End of modifications ---

dbhost = 'IAMPRDDB1.cuny.edu'
```

```
dbport = '2483'
dbsid  = 'PDIMOIG_HUD'

datasrc = cx.makedsn(dbhost, dbport,service_name = dbsid)

# Dynamically construct the connection string using the
retrieved password
# The username 'flengyel' is still here. This could also
be managed as a secret if needed.
constr = f'oracle+cx_oracle://flengyel:
{oracle_password}@{datasrc}'

# As a security measure, clear the plaintext password from
memory as soon as it's used.
oracle_password = None

# print(constr) # For debugging, do not leave uncommented

engine = sqlalchemy.create_engine(constr,
max_identifier_length=128)
conn = engine.connect()

stmt  = "SELECT DISTINCT "
stmt += "REST OF SQL NOT SHOWN "

result = conn.execute(stmt)

fh = open('daily_oim_usr_report.txt', 'w')
outcsv = csv.writer(fh, delimiter='\t')

outcsv.writerows(result)
```

```
fh.close()
# Corrected: close() is a method call
conn.close()
engine.dispose()
```

Summary of How It Works with Your Cronjob

1. At 1:00 AM, `cron` runs `/ims/cunyfirst/dailyoim_venv.sh` as the user `flengyel`.
2. The wrapper script (`dailyoim_venv.sh`) sets `LD_LIBRARY_PATH`, changes directory, and activates the Python virtual environment.
3. It then executes the **modified** `cunyfirst.py` script.
4. The Python script, now running as `flengyel` (who is a member of `appsecretaccess`), executes the `secret_retriever.get_password("oracle_db")` function.
5. This function reads the GPG passphrase from `/opt/credential_store/.gpg_passphrase` and uses it to decrypt `/opt/credential_store/oracle_db_password.txt.gpg`.
6. The decrypted password is returned and used to build the `sqlalchemy` connection string.
7. The script connects to the Oracle database and proceeds as before.

This integration is seamless and requires no changes to the cron schedule or wrapper scripts. The only change is making the Python script itself responsible for securely fetching the credentials it needs.