Utility Scripts

This file documents the administrative helper scripts used to manage the Ansible Secrets system.

`add-secret.sh`

Purpose: This script securely encrypts and adds a new secret to the Ansible project. It automates the process of creating a GPG-encrypted password file, ensuring the correct GPG passphrase from Ansible Vault is used. This eliminates common errors from manual typos or hidden characters.

Installation

This script is designed to be run from within the Ansible project directory.

Create the file `/opt/ansible_secrets/add-secret.sh`

Add the source code below to the file.

Make it executable. It should be owned by an administrator (e.g., flengyel).

```
sudo chown flengyel:"domain users"
/opt/ansible_secrets/add-secret.sh
```

```
sudo chmod 750 /opt/ansible_secrets/add-secret.sh
```

Source Code

```
#!/bin/bash
#
# add-secret.sh - A script to securely encrypt and add a new secret
# to the Ansible Secrets project.

set -euo pipefail

# --- Configuration ---
ANSIBLE_PROJECT_DIR="/opt/ansible_secrets"
FILES_DIR="${ANSIBLE_PROJECT_DIR}/files"
VAULT_FILE="${ANSIBLE_PROJECT_DIR}/group_vars/all/vault.yml"
VENV_PATH="${ANSIBLE_PROJECT_DIR}/venv/bin/activate"

# --- Input Validation ---
if [[ $# -ne 1 ]]; then
    echo "Usage: $0 <secret_name>" >&2
    echo "Example: $0 mfa_db" >&2
    exit 1
fi

SECRET_NAME="$1"
OUTPUT_FILE="${FILES_DIR}/${SECRET_NAME}_secret.txt.gpg"

if [[ ! -d "$ANSIBLE_PROJECT_DIR" || ! -f "$VAULT_FILE" || ! -f "$VENV_PATH" ]]; then
    echo "Error: Required project files or directories not
```

```
found in '$ANSIBLE_PROJECT_DIR'." >&2
    exit 1
fi

read -sp "Enter the password for '${SECRET_NAME}': "
SECRET_PASSWORD
echo
if [[ -z "$SECRET_PASSWORD" ]]; then
    echo "Error: Password cannot be empty." >&2
    exit 1
fi

if [[ -f "$OUTPUT_FILE" ]]; then
    read -p "Warning: '${OUTPUT_FILE}' already exists.
Overwrite? (y/N) " -n 1 -r
    echo
    if [[ ! $REPLY =~ ^[Yy]$ ]]; then
        echo "Operation cancelled."
        exit 1
    fi
fi

# --- Main Logic ---
echo "--> Activating virtual environment..."
source "$VENV_PATH"

echo "--> Retrieving GPG passphrase securely from Ansible
Vault..."
GPG_PASSPHRASE=$(ansible-vault view "$VAULT_FILE" | grep
'app_gpg_passphrase:' | awk '{printf "%s", $2}' | tr -d
\''"')
```

```
if [[ -z "$GPG_PASSPHRASE" ]]; then
    echo "Error: Failed to retrieve GPG passphrase from vault." >&2
    exit 1
fi

echo "--> Encrypting new secret for '${SECRET_NAME}'..."
printf '%s' "$SECRET_PASSWORD" | gpg --batch --yes --symmetric --cipher-algo AES256 \
    --passphrase "$GPG_PASSPHRASE" \
    --output "$OUTPUT_FILE"

if [[ $? -eq 0 ]]; then
    echo "✅ Success! Encrypted secret saved to: ${OUTPUT_FILE}"
    sudo chown service_account:appsecretaccess "$OUTPUT_FILE"
    echo "--> Ownership set to service_account:appsecretaccess"
else
    echo "❌ Error: GPG encryption failed." >&2
    exit 1
fi

deactivate
echo "--> Done."
```

Usage

Must be run from within the Ansible project directory

```
cd /opt/ansible_secrets
./add-secret.sh new_secret_name
```

`secure-app.sh`

Purpose: This script applies the standard production ownership (service_account:appsecretaccess) and permissions (0750) to an application script. It includes validation to ensure it is only run on `.sh` or `.py` files.

Installation

This is a general-purpose utility and should be placed in a system-wide binary path.

Create the file `/usr/local/bin/secure-app.sh`

Add the source code below to the file.

Make it executable:

```
sudo chmod 755 /usr/local/bin/secure-app.sh
```

Source Code

```
#!/bin/bash

# --- Input Validation ---
if [[ $# -ne 1 ]]; then
```

```
    echo "Usage: $0 <path_to_script>" >&2
    exit 1
fi

SCRIPT_PATH="$1"

if [[ ! -f "$SCRIPT_PATH" ]]; then
    echo "Error: File not found at '$SCRIPT_PATH'" >&2
    exit 1
fi

extension="${SCRIPT_PATH##*.}"
case "$extension" in
    sh|py)
        echo "Valid extension (.$extension) found.
Securing script..."
        ;;
    *)
        echo "Error: Invalid file type. Script only
supports '.sh' or '.py' extensions." >&2
        exit 1
        ;;
esac

# --- Main Logic ---
echo "Setting ownership to service_account:appsecretaccess
on '$SCRIPT_PATH'"
sudo chown service_account:appsecretaccess "$SCRIPT_PATH"

echo "Setting permissions to 0750 on '$SCRIPT_PATH'"
sudo chmod 0750 "$SCRIPT_PATH"
```

```
echo "Done."

Usage
# Must be run by an administrator with sudo privileges.
sudo /usr/local/bin/secure-app.sh
/path/to/your/application_script.py
```