Ansible Vault Security Configuration

Overview

This document outlines the secure vault configuration implemented for the HX Infrastructure Ansible project, following official Ansible best practices for environment-based secrets management.

Vault Identity Configuration

Environment Variables Setup

The vault configuration uses environment variables to securely manage vault passwords across different environments:

```
# Development environment
export ANSIBLE_VAULT_DEV_PASSWORD_FILE="/secure/path/to/dev_vault_password"

# Test environment
export ANSIBLE_VAULT_TEST_PASSWORD_FILE="/secure/path/to/test_vault_password"

# Production environment
export ANSIBLE_VAULT_PROD_PASSWORD_FILE="/secure/path/to/prod_vault_password"
```

Vault Identity List Configuration

The `ansible.cfg` file is configured with:

```
vault_identity_list = dev@${ANSIBLE_VAULT_DEV_PASSWORD_FILE:-/dev/null},test@${ANS-
IBLE_VAULT_TEST_PASSWORD_FILE:-/dev/null},prod@${ANSIBLE_VAULT_PROD_PASSWORD_FILE:-/
dev/null}
vault_encrypt_identity = dev
vault_id_match = True
```

Security Features

1. Environment-Based Password Management

- **No hardcoded passwords**: All vault passwords are managed through environment variables
- **Environment isolation**: Separate vault IDs for dev, test, and prod environments
- **Fallback protection**: Uses `/dev/null` as fallback to prevent accidental exposure

2. Vault ID Matching Enforcement

- **Strict matching**: `vault_id_match = True` ensures passwords are only used for matching vault IDs
- **Cross-environment protection**: Prevents accidental use of wrong environment passwords

3. Default Encryption Identity

- **Consistent encryption**: `vault_encrypt_identity = dev` ensures consistent encryption for development

Usage Examples

Encrypting Variables for Specific Environments

```
# Encrypt for development environment
ansible-vault encrypt_string --vault-id dev@prompt --stdin-name 'api_key'

# Encrypt for production environment
ansible-vault encrypt_string --vault-id prod@prompt --stdin-name 'prod_api_key'
```

Running Playbooks with Multiple Vault IDs

```
# Run with specific environment vault
ansible-playbook --vault-id dev@prompt playbook.yml

# Run with multiple environments (if needed)
ansible-playbook --vault-id dev@prompt --vault-id prod@prompt playbook.yml
```

Encrypting Files

```
# Encrypt a file for specific environment
ansible-vault encrypt --vault-id prod@prompt group_vars/prod/secrets.yml
```

Best Practices

1. Password File Security

- Store vault password files outside the repository
- Set restrictive permissions (600) on password files
- Use secure, randomly generated passwords
- Rotate passwords regularly

2. Environment Separation

- Never use production vault passwords in development
- Maintain separate vault IDs for each environment
- Use different passwords for each environment

3. CI/CD Integration

- Use secure environment variable injection in CI/CD pipelines
- Never commit vault passwords to version control
- Use external secret management systems when possible

4. Access Control

- Limit access to vault password files
- Audit vault password usage
- Use role-based access for different environments

Migration from Legacy Configuration

Previous Configuration (REMOVED)

```
# SECURITY RISK - REMOVED
vault_password_file = .vault_pass
```

Current Secure Configuration

```
# Environment-based secure configuration
vault_identity_list = dev@${ANSIBLE_VAULT_DEV_PASSWORD_FILE:-/dev/null},test@${ANS-
IBLE_VAULT_TEST_PASSWORD_FILE:-/dev/null},prod@${ANSIBLE_VAULT_PROD_PASSWORD_FILE:-/
dev/null}
vault_encrypt_identity = dev
vault_id_match = True
```

Troubleshooting

Common Issues

1. **Environment variable not set**: Ensure vault password file environment variables are properly exported
2. **Permission denied**: Check file permissions on vault password files (should be 600)
3. **Vault ID mismatch**: Ensure correct vault ID is specified when encrypting/decrypting

Verification Commands

```
# Check vault configuration
ansible-config dump | grep vault

# Test vault access
ansible-vault view --vault-id dev@prompt encrypted_file.yml

# Validate environment variables
echo $ANSIBLE_VAULT_DEV_PASSWORD_FILE
```

Security Compliance

This configuration addresses:
- **Phase 2 Critical Objective**: Vault Security Configuration
- **Ansible Security Best Practices**: Environment-based secrets management
- **Zero hardcoded passwords**: All passwords managed through secure environment variables
- **Environment isolation**: Strict separation between dev/test/prod vault access

References

- Ansible Vault Guide (https://docs.ansible.com/ansible/latest/vault_guide/index.html)
- Vault Password Management (https://docs.ansible.com/ansible/latest/vault_guide/vault_managing_passwords.html)
- Ansible Security Best Practices (https://docs.ansible.com/ansible/latest/tips_tricks/ansible_tips_tricks.html#keep-vaulted-variables-safely-visible)