Ansible Lockdown

> [19반] 정민석_7000

Ansible Lockdown 실행

먼저 Docker ubuntu:24.04 환경에서 Ansible Lockdown을 실행하고자 하였습니다.

https://docs.ansible.com/ansible/latest/installation_guide/installation_distros.html 을 참고하여 다음의 Dockerfile 을 작성하였습니다.

```
# Dockerfile

FROM ubuntu:24.04

WORKDIR /ansible
RUN apt update
RUN apt install -y software-properties-common
RUN add-apt-repository --yes --update ppa:ansible/ansible
RUN apt install -y ansible
RUN apt install -y vim
RUN apt install -y git

# ansible rule 중 Docker에서 필요한 task만 수행하도록 만들면 openssh-server는 필요없어짐
RUN apt install -y openssh-server

RUN git clone https://github.com/ansible-lockdown/UBUNTU24-CIS.git
```

아래 명령어을 통하여 컨테이너에 접속합니다.

```
docker build -t ansible-ubuntu24 .
docker run -it --name ansible-ubuntu24-container ansible-ubuntu24 /bin/bash
```

ansible을 사용할때에, sh에 명령어를 최소한으로 입력하는 것이, ansible의 목적상, 중요하다고 생각합니다. 더하여 root pw를 직접 지정하는 것이 아닌 ansible-vault를 통하여 지정하도록 하는 것이, ansible의 목적상, 타당하다고 생각합니다. 이에 다음의 최소한의 명령어를 통하여 사전 준비를 마칩니다.

```
cd UBUNTU24-CIS
ansible-vault encrypt_string '{{Secret}}' --name 'root_password'
New Vault password:
Confirm New Vault password:
Encryption successful
root_password: !vault |
          $ANSIBLE_VAULT;1.1;AES256
          30363031646463393431336333363765623361393230643838313934626234663531636635613834
          36343966336135636330623937396663616663343532633 90a323862373431636462396265363035
          64356234316237626438313766633161616431346537343964666562623965656566383633666536
          30333665303064353 50a653734393264323931323663616231333961343038666137383866353231
          6261
```

그리고 ansible-vault에서 출력된 root_password를 참고하여 다음의 파일을 생성합니다. 더하여 보안적인 관점에서 vault 값을 따로 파일로 분리하여 관리하는 것이 타당하다고 판단됩니다.

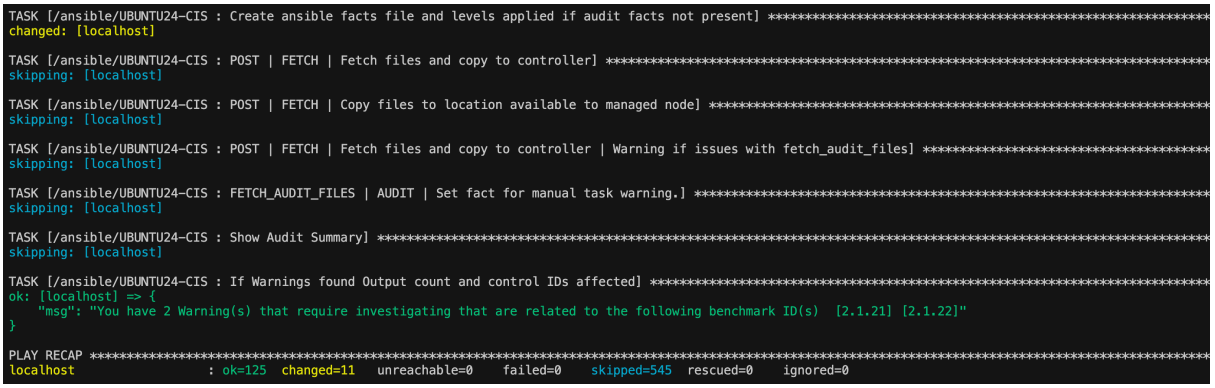
```
# inventory.yaml
---
all:
  hosts:
    localhost:
      ansible_connection: local
```

```
# playbook.yaml
---
- name: Run CIS benchmark
```

```
  hosts: localhost
  become: true
  gather_facts: true

  vars:
    root_password: !vault |
        $ANSIBLE_VAULT;1.1;AES256
          306661636137336239393134396566663665613636461333033643
436646331336430656438633
          656536643738656562613339643439393963396661653539 0a3934
353532626430383136626165 35
          633835366461396230363236336338343433616564646165376331 3
562643331343832663963663 2
          656239303135626532 0a356366306435343266633232373034346 3
383435306265340303939336 3
          3966
    ubtu24cis_rule_6_1_1_1: false
    ubtu24cis_rule_6_1_2_1_4: false
    ubtu24cis_rule_6_2_2_4: false
    ubtu24cis_rule_6_3_3: false

  pre_tasks:
    - name: Set root password
      ansible.builtin.user:
        name: root
        password: "{{ root_password }}"

    - name: Ensure /etc/default/grub exists
      ansible.builtin.file:
        path: /etc/default/grub
        state: touch
        mode: "0644"

  roles:
    - role: "{{ playbook_dir }}"
```

다음의 명령어로 2번 실행 시에 정상적으로 실행된 것으로 파악하였습니다.

```
ansible-playbook -i inventory.yaml playbook.yaml --ask-vault-pass
```



다만 한계를 보인 부분이 있습니다. 첫번째 실행시에는 아래의 에러가 나왔습니다. 그런데 두번째 실행시에는 정상적으로 실행되었습니다.

```
fatal: [localhost]: FAILED! => {"changed": false, "msg": "Service is in unknow
n state", "status": {}}
```

이유는 도커 환경에서 필요없는 task을 실행했기 때문일 것입니다. task를 잠시 살펴본 결과, section6이 필요한 것인지에 대한 의문이 들었습니다. 다만, section6의 task 각각이 정말로 Docker 환경에 필요없는 것인지 확인할 수 있는 확고한 지식이 부족하기에, 실행에서 오류를 발생시키는 section6의 일부 task만 비활성화 시켰습니다. 이런식으로 필요한 task만 실행할 수 있게 최적화한다면, Dockerfile에서의 openssh-server을 설치하지 않아도 될 것이며, 꼭 필요한 rule만 수행함으로 리소스를 아낄 수 있습니다.

https://learn.cisecurity.org/benchmarks 이런 문서를 참고하여 Docker 환경에서 필요한 rule을 선별할 수 있어보입니다.

Ansible Lockdown의 구조 분석

UBUNTU24-CIS의 주요 디렉토리 구조는 다음과 같습니다.

```
site.yml
/tasks
    main.yml
    /section_1
```

```
    main.yml
    ...
/section_2
    main.yml
    ...
....
```

여기에서 main.yml의 역할이 중요합니다. 각 디렉토리의 main.yml은 해당 디렉토리에 있는 다른 task에 의존합니다. 즉, main.yml에서 다른 task를 모아서 실행합니다. /task/main.yml은 OS, ansible version 검사 등을 진행하며 /task/section_N/main.yml을 실행하게 됩니다.

결론적으로 특정 단위별로 section을 나누고, section 내의 task는 해당 section 디렉토리의 main.yml에서 실행됩니다. 더하여 task의 main.yml은 각 섹션의 main.yml을 실행합니다. 이렇게 잘게 나뉘어진 단위를, main.yml이 적절히 의존하게 하여, 유지보수가 용이하도록 설계한 것으로 생각합니다.