Name: Torrecampo, Juan Piolo S.	**Date Performed:** May 2, 2023
Course/Section: CPE 234 - CPE32S3	**Date Submitted:** May 9, 2023
Instructor: Engr. Taylar	**Semester and SY:** 2022-2023

Activity 13: Security Automation

1. Objectives

- Provide procedures for automating and streamlining various security processes needed to identify, triage, and respond to security events using Ansible.

2. Discussion

Firewall policy management with Ansible security automation

As a security operator, you can use Ansible security automation to manage multiple firewall policies. Create and delete firewall rules to block or unblock a source IP address from accessing a destination IP address.

An organization's network firewall is the first line of defense against an attack and a vital component for maintaining a secure environment. As a security operator, you construct and manage secure networks to ensure that your firewall only allows inbound and outbound network traffic defined by your organization's firewall policies. A firewall policy consists of security rules that protect the network against harmful incoming and outgoing traffic.

Managing multiple firewall rules across various products and vendors can be both challenging and time consuming for security teams. Manual workflow processes that involve complex tasks can result in errors and ultimately cause delays in investigating an application's suspicious behavior or stopping an ongoing attack on a server. When every solution in a security portfolio is automated through the same language, both security analysts and operators can perform a series of actions across various products in a fraction of the time. This automated process maximizes the overall efficiency of the security team.

Ansible security automation interacts with a wide variety of security technologies from a range of vendors. Ansible enables security teams to manage different products, interfaces, and workflows in a unified way to produce a successful deployment. For example, your security team can automate tasks such as blocking and unblocking IP and URLs on supported technologies such as enterprise firewalls.

3. Tasks

Task 1: Automate Firewall Rules

Ansible security automation enables you to automate various firewall policies that require a series of actions across various products. You can use an Ansible role, such as the acl_manager role to manage your Access Control Lists (ACLs) for many firewall devices such as blocking or unblocking an IP or URL. Roles let you automatically load related vars, files, tasks, handlers, and other Ansible artifacts based on a known file structure. After you group your content in roles, you can easily reuse them and share them with other users.

Creating a new firewall rule

Use the acl_manager role to create a new firewall rule for blocking a source IP address from accessing a destination IP address.

Prerequisites

- You have installed Ansible 2.9 or later
- You have access to the Check Point Management server to enforce the new policies

1. Install the acl_manager role using the ansible-galaxy command.

   ```
   $ ansible-galaxy install ansible_security.acl_manager
   ```

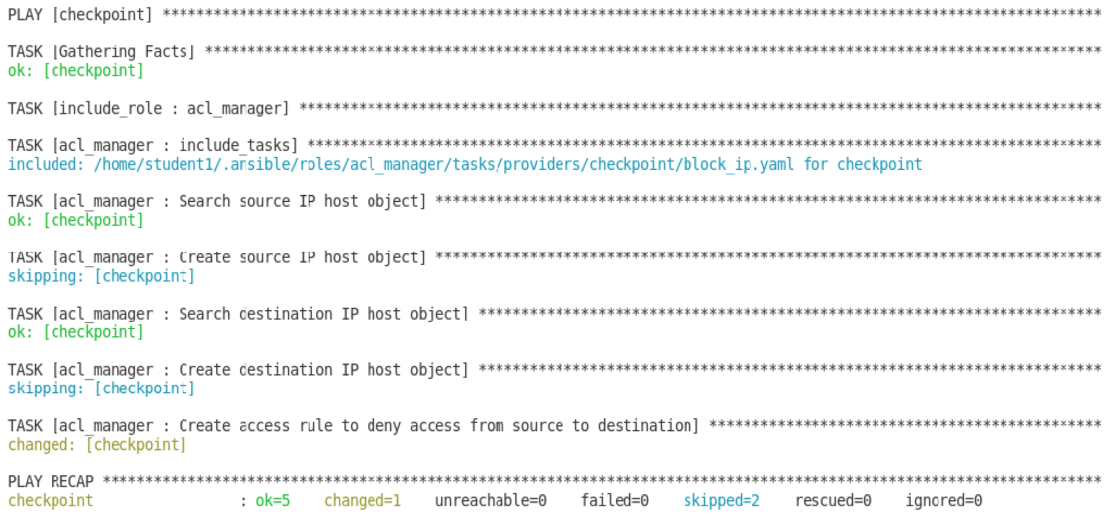
2. Create a new playbook and set the following parameter. For example, source object, destination object, access rule between the two objects and the actual firewall you are managing, such as Check Point:

   ```
   - name: block IP address
     hosts: checkpoint
     connection: httpapi

     tasks:
       - include_role:
           name: acl_manager
           tasks_from: block_ip
         vars:
           source_ip: 172.17.13.98
           destination_ip: 192.168.0.10
           ansible_network_os: checkpoint
   ```

3. Run the playbook

   ```
   $ ansible-navigator run --ee false <playbook.yml>
   ```

   Below is a sample output: (provide your output)



```
PLAY [checkpoint] **************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************
ok: [checkpoint]

TASK [include_role : acl_manager] **********************************************************************************************

TASK [acl_manager : include_tasks] *********************************************************************************************
included: /home/student1/.ansible/roles/acl_manager/tasks/providers/checkpoint/block_ip.yaml for checkpoint

TASK [acl_manager : Search source IP host object] ******************************************************************************
ok: [checkpoint]

TASK [acl_manager : Create source IP host object] ******************************************************************************
skipping: [checkpoint]

TASK [acl_manager : Search destination IP host object] *************************************************************************
ok: [checkpoint]

TASK [acl_manager : Create destination IP host object] *************************************************************************
skipping: [checkpoint]

TASK [acl_manager : Create access rule to deny access from source to destination] **********************************************
changed: [checkpoint]

PLAY RECAP *********************************************************************************************************************
checkpoint                 : ok=5    changed=1    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
```

4. **Verification**

   You have created a new firewall rule that blocks a source IP address from accessing a destination IP address. Access the MGMT server and verify that the new security policy has been created.

Task 2: Deleting a Firewall Rule

Use the acl_manager role to delete a security rule.

Prerequisites

- You have installed Ansible 2.9 or later
- You have access to the firewall MGMT servers to enforce the new policies

1. Install the acl_manager role using the ansible-galaxy command.
   $ ansible-galaxy install ansible_security.acl_manager

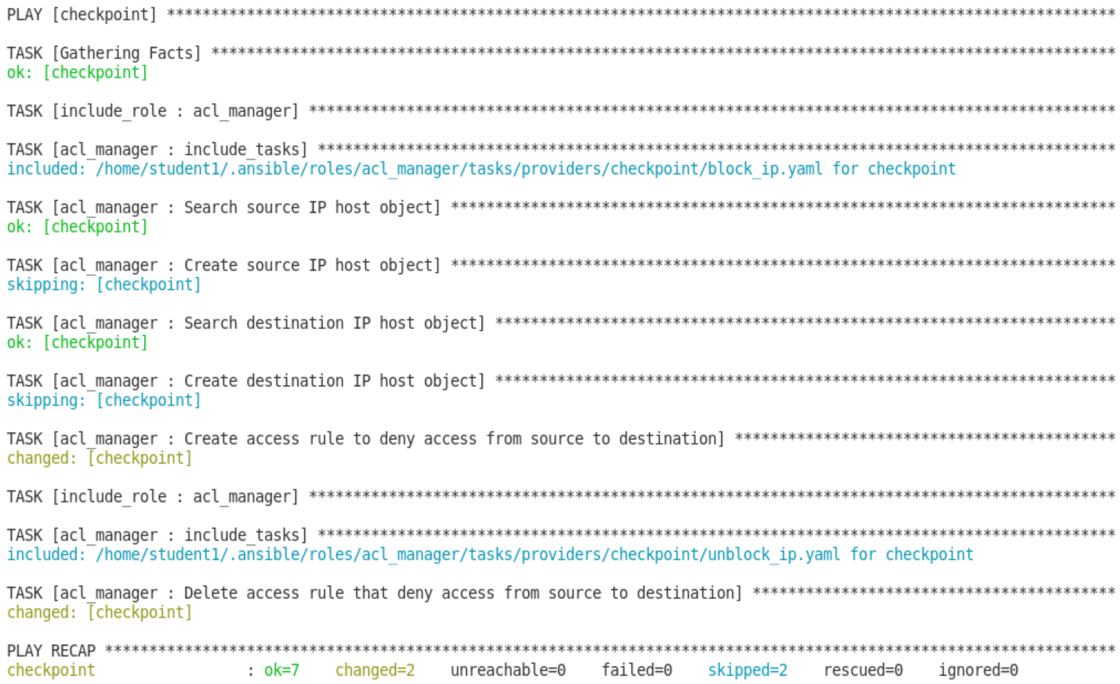
2. Using CLI, create a new playbook with the acl_manger role and set the parameters (e.g., source object, destination object, access rule between the two objects):

```
- name: delete block list entry
  hosts: checkpoint
  connection: httpapi

    - include_role:
        name: acl_manager
        Tasks_from: unblock_ip
      vars:
        source_ip: 192.168.0.10
        destination_ip: 192.168.0.11
        ansible_network_os: checkpoint
```

3. Run the playbook

   $ ansible-navigator run --ee false <playbook.yml>

   Below is a sample output: (provide your output)



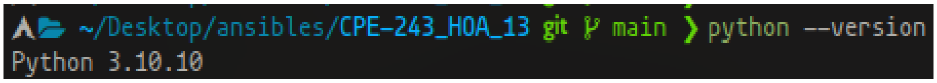
```
PLAY [checkpoint] ***************************************************************************************************

TASK [Gathering Facts] **********************************************************************************************
ok: [checkpoint]

TASK [include_role : acl_manager] ***********************************************************************************

TASK [acl_manager : include_tasks] **********************************************************************************
included: /home/student1/.ansible/roles/acl_manager/tasks/providers/checkpoint/block_ip.yaml for checkpoint

TASK [acl_manager : Search source IP host object] *******************************************************************
ok: [checkpoint]

TASK [acl_manager : Create source IP host object] *******************************************************************
skipping: [checkpoint]

TASK [acl_manager : Search destination IP host object] **************************************************************
ok: [checkpoint]

TASK [acl_manager : Create destination IP host object] **************************************************************
skipping: [checkpoint]

TASK [acl_manager : Create access rule to deny access from source to destination] ***********************************
changed: [checkpoint]

TASK [include_role : acl_manager] ***********************************************************************************

TASK [acl_manager : include_tasks] **********************************************************************************
included: /home/student1/.ansible/roles/acl_manager/tasks/providers/checkpoint/unblock_ip.yaml for checkpoint

TASK [acl_manager : Delete access rule that deny access from source to destination] *********************************
changed: [checkpoint]

PLAY RECAP **********************************************************************************************************
checkpoint                 : ok=7    changed=2    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
```

4. **Verification**

   You have deleted the firewall rule. Access the MGMT server and verify that the new security policy has been removed.

4. Output (screenshots and explanations). Run the playbook and explain the outputs.

1. Checking and installing prerequisites.

python => 2.9 or higher	 ~/Desktop/ansibles/CPE-243_HOA_13 git main ❯ python --version Python 3.10.10

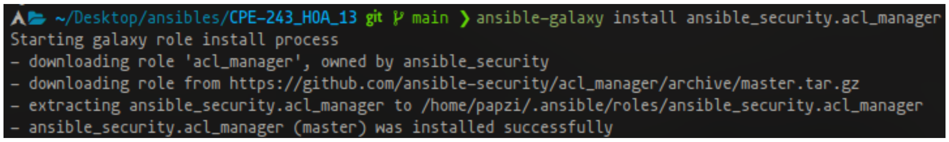
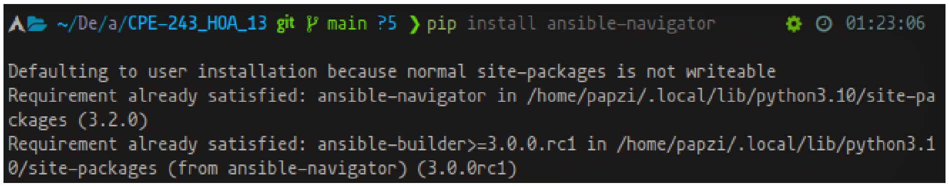
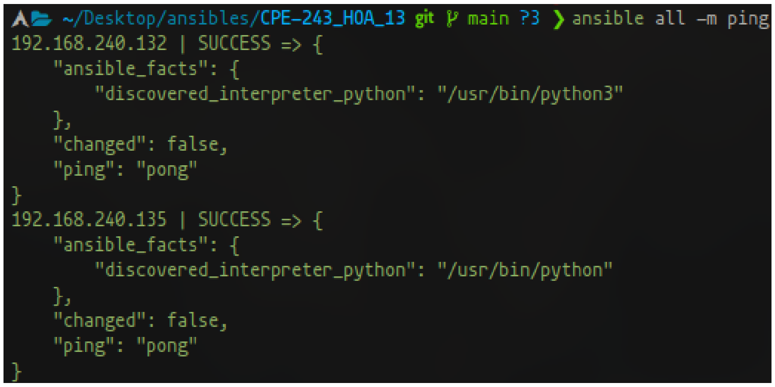
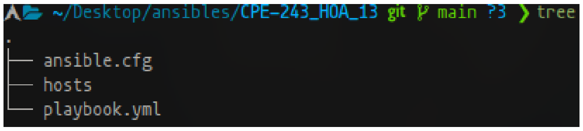
acl_manager role	 ~/Desktop/ansibles/CPE-243_HOA_13 git main ❯ ansible-galaxy install ansible_security.acl_manager Starting galaxy role install process - downloading role 'acl_manager', owned by ansible_security - downloading role from https://github.com/ansible-security/acl_manager/archive/master.tar.gz - extracting ansible_security.acl_manager to /home/papzi/.ansible/roles/ansible_security.acl_manager - ansible_security.acl_manager (master) was installed successfully
Installing ansible-navigator	 ~/De/a/CPE-243_HOA_13 git main ?5 ❯ pip install ansible-navigator 01:23:06 Defaulting to user installation because normal site-packages is not writeable Requirement already satisfied: ansible-navigator in /home/papzi/.local/lib/python3.10/site-packages (3.2.0) Requirement already satisfied: ansible-builder>=3.0.0.rc1 in /home/papzi/.local/lib/python3.10/site-packages (from ansible-navigator) (3.0.0rc1)

Table 1. The table above shows the prerequisites at the first column and the proof for the second column.



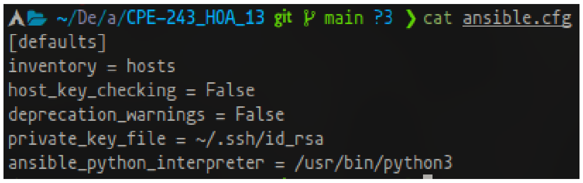
```
~/Desktop/ansibles/CPE-243_HOA_13 git main ?3 ❯ ansible all -m ping
192.168.240.132 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
192.168.240.135 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
```

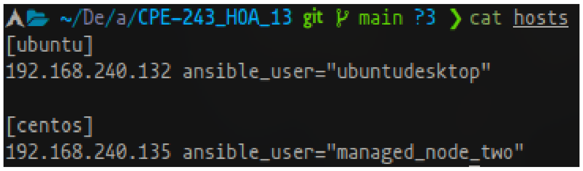
Figure 1. The screenshot above shows that the ansible can connect to 2 servers.



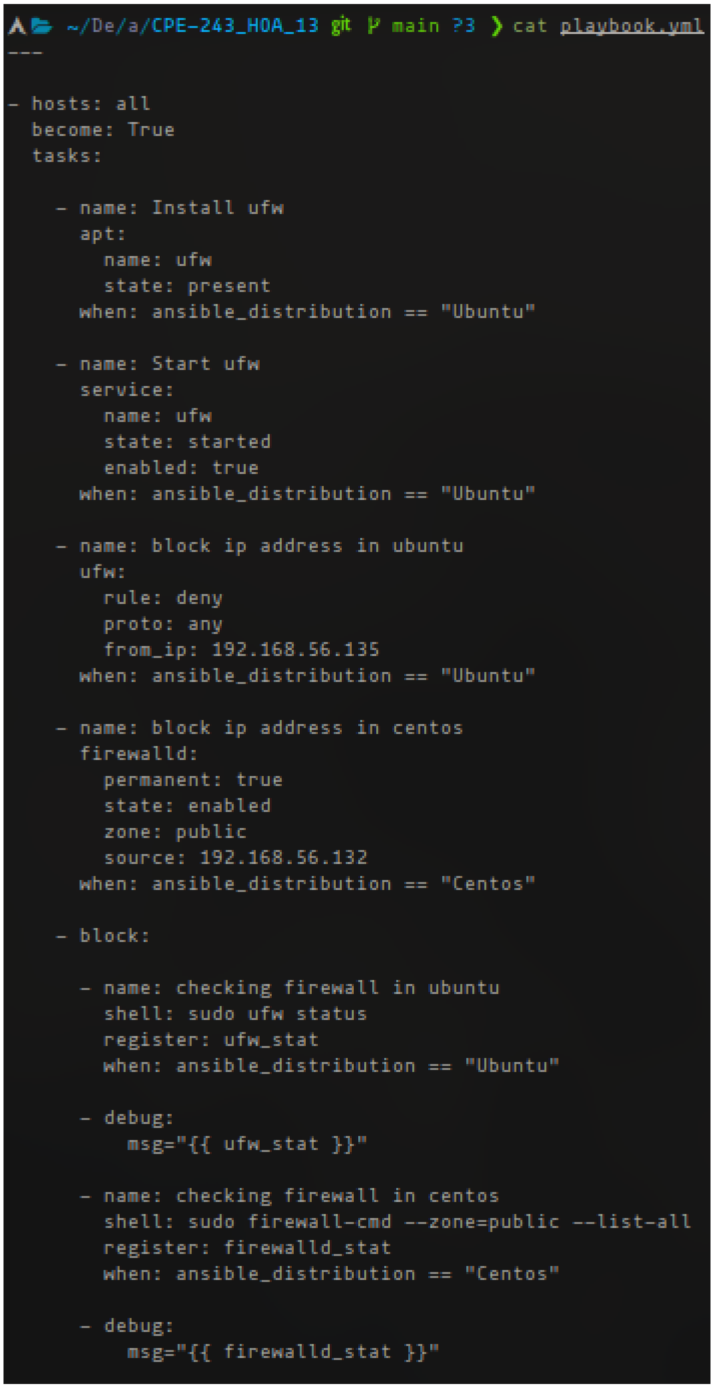
```
~/Desktop/ansibles/CPE-243_HOA_13 git main ?3 ❯ tree
.
├── ansible.cfg
├── hosts
└── playbook.yml
```

Figure 2. The screenshot above tree file structure of the repository.

ansible.cfg	
	~/De/a/CPE-243_HOA_13 git main ?3 ❯ cat ansible.cfg
	[defaults]
	inventory = hosts
	host_key_checking = False
	deprecation_warnings = False
	private_key_file = ~/.ssh/id_rsa
	ansible_python_interpreter = /usr/bin/python3

hosts	 ~/De/a/CPE-243_HOA_13 git main ?3 ❯ cat hosts [ubuntu] 192.168.240.132 ansible_user="ubuntudesktop" [centos] 192.168.240.135 ansible_user="managed_node_two"

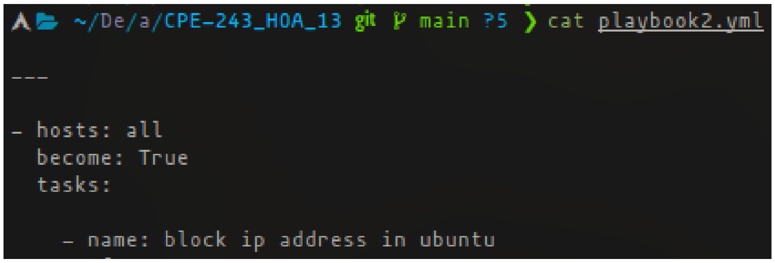
playbook.yml (block)



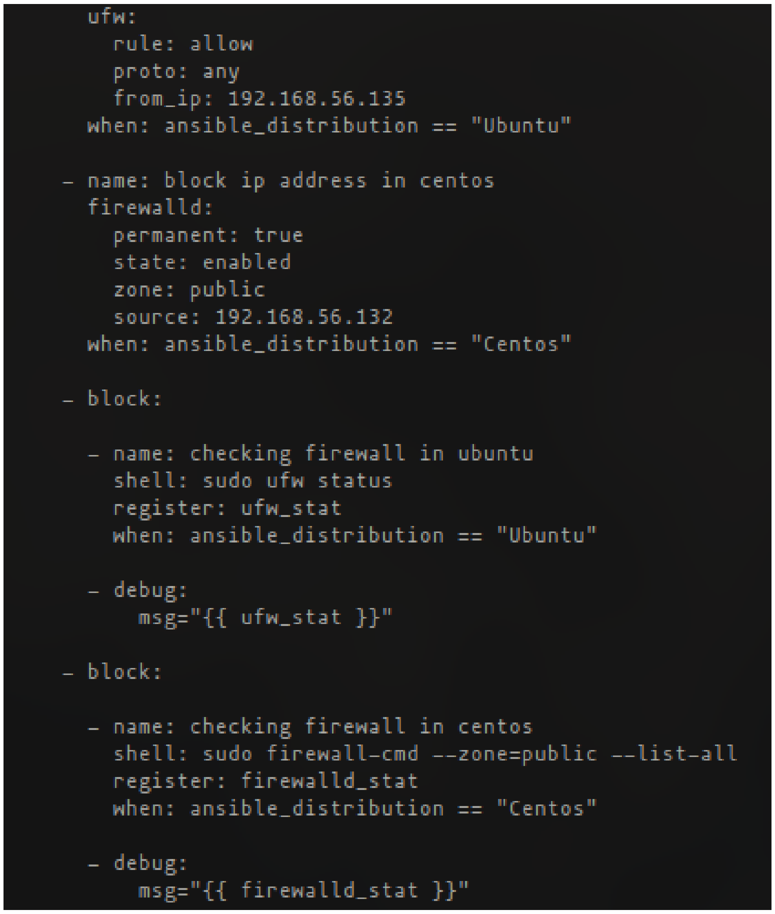
```
~/De/a/CPE-243_HOA_13 git main ?3 ❯ cat playbook.yml
---

- hosts: all
  become: True
  tasks:

    - name: Install ufw
      apt:
        name: ufw
        state: present
      when: ansible_distribution == "Ubuntu"

    - name: Start ufw
      service:
        name: ufw
        state: started
        enabled: true
      when: ansible_distribution == "Ubuntu"

    - name: block ip address in ubuntu
      ufw:
        rule: deny
        proto: any
        from_ip: 192.168.56.135
      when: ansible_distribution == "Ubuntu"

    - name: block ip address in centos
      firewalld:
        permanent: true
        state: enabled
        zone: public
        source: 192.168.56.132
      when: ansible_distribution == "Centos"

    - block:

      - name: checking firewall in ubuntu
        shell: sudo ufw status
        register: ufw_stat
        when: ansible_distribution == "Ubuntu"

      - debug:
          msg="{{ ufw_stat }}"

      - name: checking firewall in centos
        shell: sudo firewall-cmd --zone=public --list-all
        register: firewalld_stat
        when: ansible_distribution == "Centos"

      - debug:
          msg="{{ firewalld_stat }}"
```

c4

playbook2.yml (delete)

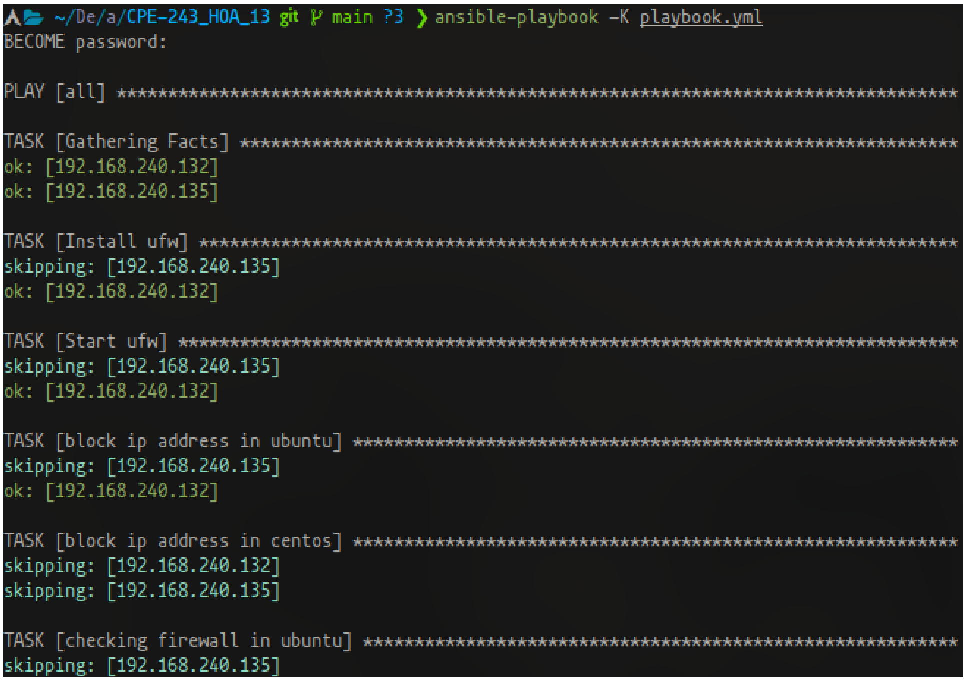


```
~/De/a/CPE-243_HOA_13 git main ?5 ❯ cat playbook2.yml

---

- hosts: all
  become: True
  tasks:

    - name: block ip address in ubuntu
```

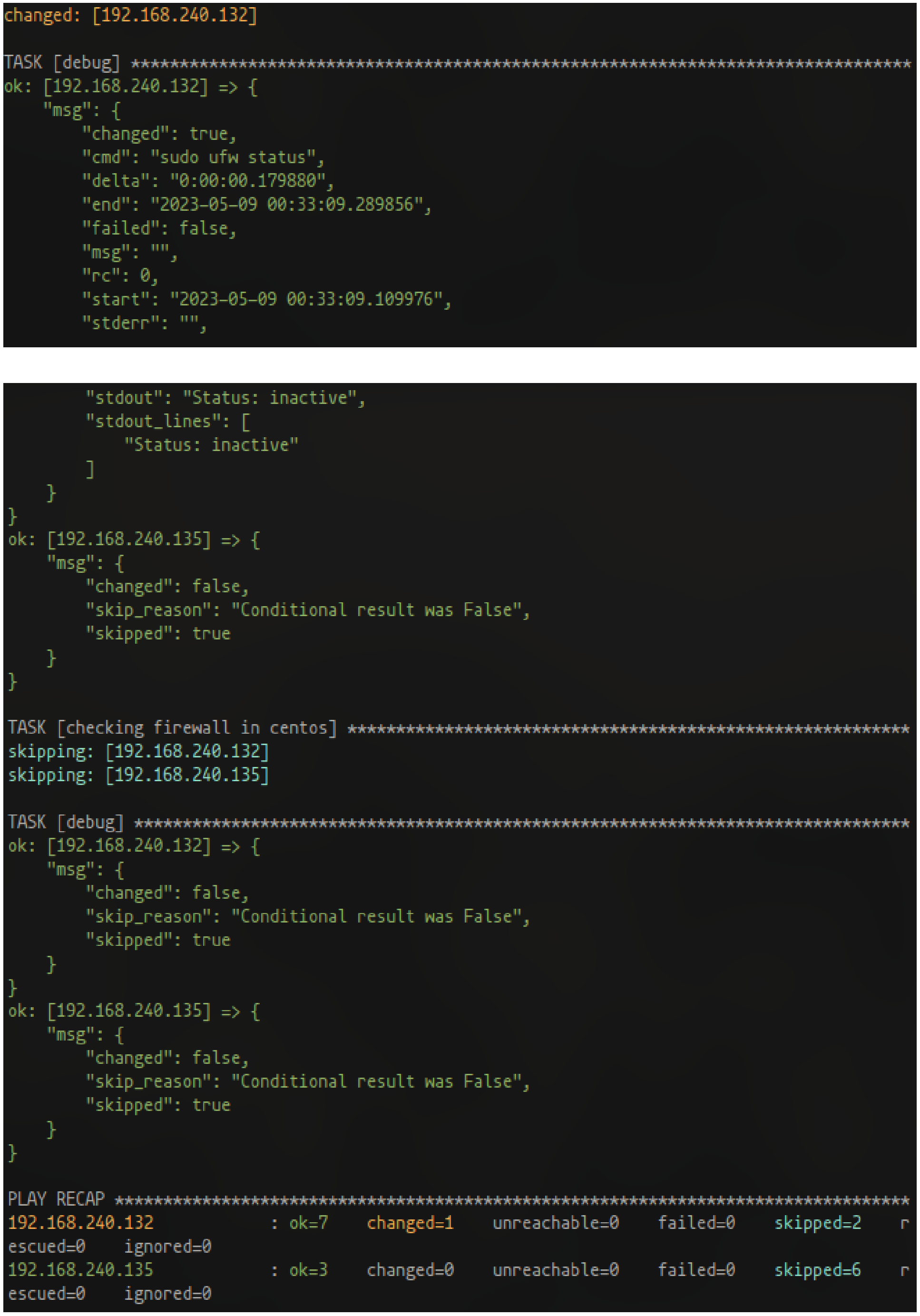


```
    ufw:
      rule: allow
      proto: any
      from_ip: 192.168.56.135
    when: ansible_distribution == "Ubuntu"

  - name: block ip address in centos
    firewalld:
      permanent: true
      state: enabled
      zone: public
      source: 192.168.56.132
    when: ansible_distribution == "Centos"

  - block:

    - name: checking firewall in ubuntu
      shell: sudo ufw status
      register: ufw_stat
      when: ansible_distribution == "Ubuntu"

    - debug:
        msg="{{ ufw_stat }}"

  - block:

    - name: checking firewall in centos
      shell: sudo firewall-cmd --zone=public --list-all
      register: firewalld_stat
      when: ansible_distribution == "Centos"

    - debug:
        msg="{{ firewalld_stat }}"
```

Table 2. The table above shows the scripts inside of the repository.



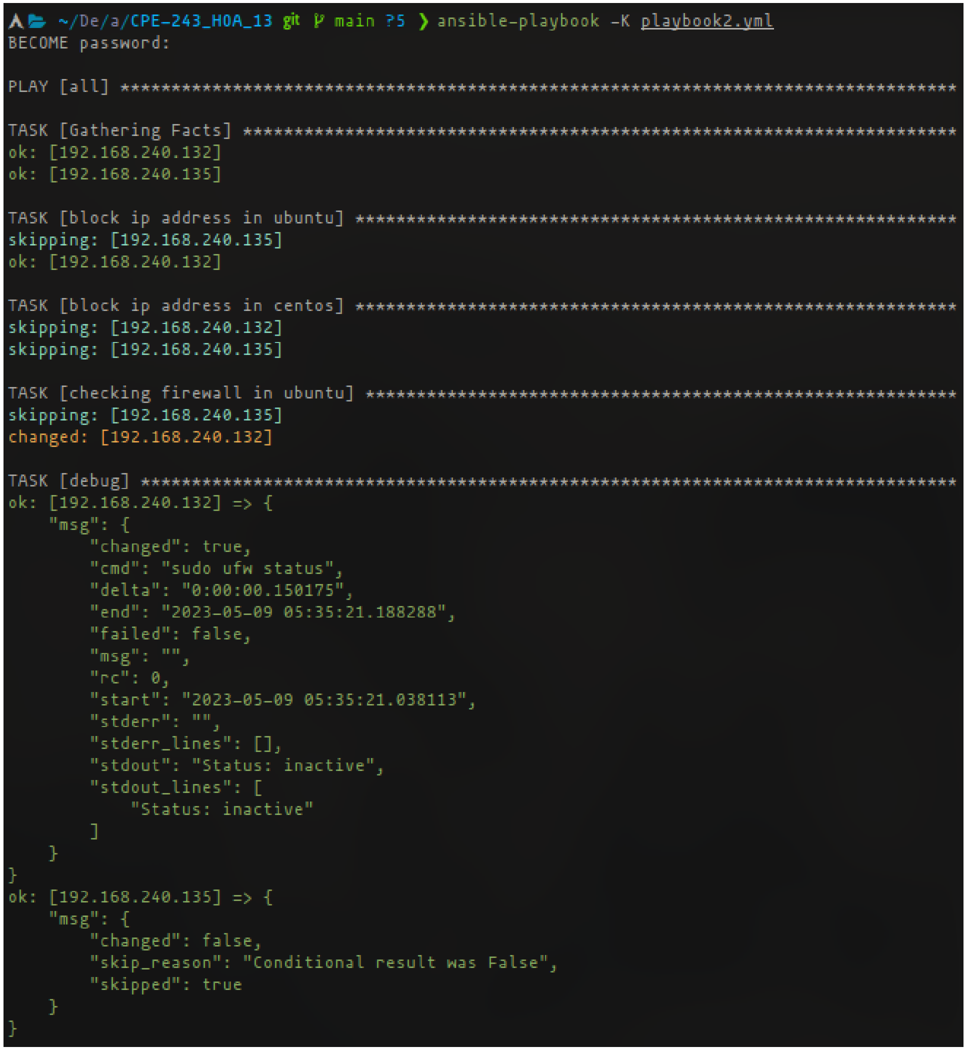
```
~/De/a/CPE-243_HOA_13 git main ?3 ❯ ansible-playbook -K playbook.yml
BECOME password:

PLAY [all] ***********************************************************************

TASK [Gathering Facts] ***********************************************************
ok: [192.168.240.132]
ok: [192.168.240.135]

TASK [Install ufw] ***************************************************************
skipping: [192.168.240.135]
ok: [192.168.240.132]

TASK [Start ufw] *****************************************************************
skipping: [192.168.240.135]
ok: [192.168.240.132]

TASK [block ip address in ubuntu] ************************************************
skipping: [192.168.240.135]
ok: [192.168.240.132]

TASK [block ip address in centos] ************************************************
skipping: [192.168.240.132]
skipping: [192.168.240.135]

TASK [checking firewall in ubuntu] ***********************************************
skipping: [192.168.240.135]
```



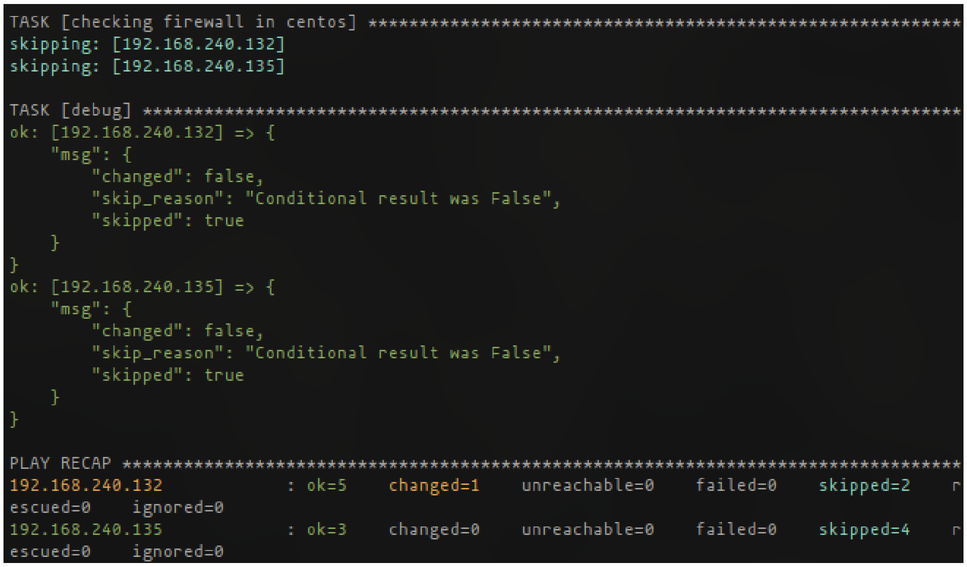
```
changed: [192.168.240.132]

TASK [debug] *********************************************************************************
ok: [192.168.240.132] => {
    "msg": {
        "changed": true,
        "cmd": "sudo ufw status",
        "delta": "0:00:00.179880",
        "end": "2023-05-09 00:33:09.289856",
        "failed": false,
        "msg": "",
        "rc": 0,
        "start": "2023-05-09 00:33:09.109976",
        "stderr": "",
```

```
        "stdout": "Status: inactive",
        "stdout_lines": [
            "Status: inactive"
        ]
    }
}
ok: [192.168.240.135] => {
    "msg": {
        "changed": false,
        "skip_reason": "Conditional result was False",
        "skipped": true
    }
}

TASK [checking firewall in centos] ***********************************************************
skipping: [192.168.240.132]
skipping: [192.168.240.135]

TASK [debug] *********************************************************************************
ok: [192.168.240.132] => {
    "msg": {
        "changed": false,
        "skip_reason": "Conditional result was False",
        "skipped": true
    }
}
ok: [192.168.240.135] => {
    "msg": {
        "changed": false,
        "skip_reason": "Conditional result was False",
        "skipped": true
    }
}

PLAY RECAP ***********************************************************************************
192.168.240.132            : ok=7    changed=1    unreachable=0    failed=0    skipped=2    r
escued=0    ignored=0
192.168.240.135            : ok=3    changed=0    unreachable=0    failed=0    skipped=6    r
escued=0    ignored=0
```

Figure 4. The output above shows the result after running the paybook.



```
~/De/a/CPE-243_HOA_13 git  main ?5 ❯ ansible-playbook -K playbook2.yml
BECOME password:

PLAY [all] ***************************************************************************

TASK [Gathering Facts] ***************************************************************
ok: [192.168.240.132]
ok: [192.168.240.135]

TASK [block ip address in ubuntu] ****************************************************
skipping: [192.168.240.135]
ok: [192.168.240.132]

TASK [block ip address in centos] ****************************************************
skipping: [192.168.240.132]
skipping: [192.168.240.135]

TASK [checking firewall in ubuntu] ***************************************************
skipping: [192.168.240.135]
changed: [192.168.240.132]

TASK [debug] *************************************************************************
ok: [192.168.240.132] => {
    "msg": {
        "changed": true,
        "cmd": "sudo ufw status",
        "delta": "0:00:00.150175",
        "end": "2023-05-09 05:35:21.188288",
        "failed": false,
        "msg": "",
        "rc": 0,
        "start": "2023-05-09 05:35:21.038113",
        "stderr": "",
        "stderr_lines": [],
        "stdout": "Status: inactive",
        "stdout_lines": [
            "Status: inactive"
        ]
    }
}
ok: [192.168.240.135] => {
    "msg": {
        "changed": false,
        "skip_reason": "Conditional result was False",
        "skipped": true
    }
}
```



```
TASK [checking firewall in centos] ***************************************************
skipping: [192.168.240.132]
skipping: [192.168.240.135]

TASK [debug] *************************************************************************
ok: [192.168.240.132] => {
    "msg": {
        "changed": false,
        "skip_reason": "Conditional result was False",
        "skipped": true
    }
}
ok: [192.168.240.135] => {
    "msg": {
        "changed": false,
        "skip_reason": "Conditional result was False",
        "skipped": true
    }
}

PLAY RECAP ***************************************************************************
192.168.240.132            : ok=5    changed=1    unreachable=0    failed=0    skipped=2    r
escued=0    ignored=0
192.168.240.135            : ok=3    changed=0    unreachable=0    failed=0    skipped=4    r
escued=0    ignored=0
```

Figure 5. The output above shows the result after running the playbook2.

Reflections:

Answer the following:

1. What is the importance of applying firewall policy management?

    - Firewall policy management is crucial for network security as it helps secure networks by controlling access to resources and blocking unauthorized traffic. It also aids in compliance with regulatory requirements for industries that mandate the use of firewalls to protect sensitive data, and helps organizations manage risks by identifying and prioritizing vulnerabilities and threats to the network. Firewall policy management enables administrators to optimize network resources and prevent unnecessary congestion by allowing only necessary traffic through the firewall, and can aid in incident response by providing logs and alerts for unauthorized access attempts. In summary, firewall policy management is a critical component of an organization's cybersecurity strategy, ensuring that the network is secure, compliant, and optimized for business operations.

Conclusions:

To sum up, the utilization of Ansible for automating and streamlining various security processes is highly advantageous in terms of identifying, triaging, and responding to security events. Automating security processes allows security teams to save time and effort when detecting and resolving security incidents, enabling them to focus on more significant security tasks. Ansible's customizable framework provides flexibility to security teams to automate security processes that are tailored to their organization's specific needs. This approach can improve the overall security posture of an organization and minimize the risks of security breaches. By implementing Ansible-based procedures for automating and streamlining security processes, organizations can enhance their security operations and protect their critical assets more effectively.