

Name: Torrecampo, Juan Piolo S.	Date Performed: Apr 27, 2023
Course/Section: CPE 234 - CPE32S3	Date Submitted: Apr 27, 2023
Instructor: Engr. Taylar	Semester and SY: 2022-2023
Activity 11.1: Security and Ansible	
1. Objectives	
<ul style="list-style-type: none"> Implement security using Ansible playbook that you can run against your systems on first boot to harden them. Implement hardening tasks and how they can be accomplished in a repeatable way with Ansible 	
2. Discussion	
<p>Every sysadmin has a checklist of security-related tasks that they execute against each new system. This checklist is a great habit, as it ensures that new servers in your environment meet a set of minimum-security requirements. However, doing this work manually is also slow and error prone. It's easy to encounter configuration inconsistencies due to the manual series of steps, and there's no way to address configuration drift without manually re-running your checklist.</p> <p>Implementing your security workflow in Ansible is a great way to automate some "low hanging fruit" in your environment. This activity will teach you some of the basic steps that to harden a new system, and it shows you how to implement them using Ansible. The automations in this activity aren't earth-shattering; they're probably little things that you already do to secure your systems. However, automating these tasks ensures that your infrastructure is configured in a consistent, repeatable way across your environment.</p>	
3. Tasks	
<p>Sample environment that you can implement for the main.yml</p> <pre> \$ tree . ├── files │ └── etc │ ├── issue │ ├── motd │ ├── ssh │ │ └── sshd_config │ ├── sudoers.d │ │ └── admin │ └── inventory.ini └── main.yml 4 directories, 6 files \$ cat inventory.ini nyc1-webserver-1.example.com </pre>	

Task 1: Patching a Software

The first task is to ensure that the software of a new system is fully patched. An incredible amount of attack surface can be eliminated by merely staying vigilant about patching. Ansible makes this easy. The task below fully patches all of your packages and can easily be run regularly.

```
- name: Perform full patching
  package:
    name: '*'
    state: latest
```

Task 2: Secure Remote Access

Once the host is patched, secure remote access via SSH. First, create a local user with sudo permissions to disable remote login by the root user. The tasks below are just an example, and you will probably want to customize them to meet your needs:

```
- name: Add admin group
  group:
    name: admin
    state: present

- name: Add local user
  user:
    name: admin
    group: admin
    shell: /bin/bash
    home: /home/admin
    create_home: yes
    state: present

- name: Add SSH public key for user
  authorized_key:
    user: admin
    key: "{{ lookup('file', '~/.ssh/id_rsa.pub') }}"
    state: present

- name: Add sudoer rule for local user
  copy:
    dest: /etc/sudoers.d/admin
    src: etc/sudoers.d/admin
    owner: root
    group: root
    mode: 0440
    validate: /usr/sbin/visudo -csf %s
```

These tasks add a local "admin" user and group, add an SSH public key for the user, and add a sudo rule for the admin user that permits passwordless sudo. The SSH key will be the same public key for the user that is locally executing the Ansible playbook.

Ansible can be used to create a known-good configuration for all of the servers in your environment. This goes a long way toward enforcing a consistent security posture for one of your most critical services, especially if you remain vigilant about regularly executing Ansible across your hosts.

Ansible's copy module is used to lay down this configuration file on remote systems:

```
- name: Add hardened SSH config
  copy:
    dest: /etc/ssh/sshd_config
    src: etc/ssh/sshd_config
    owner: root
    group: root
    mode: 0600
    notify: Reload SSH
```

A sample SSH configuration file can be found below. It's mostly a default file with some additional tuning, such as disabling password authentication and barring root login. You will likely develop your own configuration file best practices to meet your organization's needs, such as only permitting a specific set of approved ciphers.

```
$ cat files/etc/ssh/sshd_config
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key
SyslogFacility AUTHPRIV
AuthorizedKeysFile .ssh/authorized_keys
PasswordAuthentication no
ChallengeResponseAuthentication no
GSSAPIAuthentication yes
GSSAPICleanupCredentials no
UsePAM yes
X11Forwarding no
Banner /etc/issue.net
AcceptEnv LANG LC_CTYPE LC_NUMERIC LC_TIME LC_COLLATE LC_MONETARY
LC_MESSAGES
AcceptEnv LC_PAPER LC_NAME LC_ADDRESS LC_TELEPHONE LC_MEASUREMENT
AcceptEnv LC_IDENTIFICATION LC_ALL LANGUAGE
AcceptEnv XMODIFIERS
Subsystem sftp /usr/libexec/openssh/sftp-server
PermitRootLogin no

PermitRootLogin no
```

Finally, notice that I use a **handler** to trigger a refresh of the sshd service. That handler is found in the **handlers** section of the playbook.

```
handlers:
  - name: Reload SSH
    service:
      name: sshd
      state: reloaded
```

Once you have SSH locked down from a configuration perspective, it's time to restrict SSH to only permitted IP addresses. If you're using the default firewalld, this is easily done by moving the SSH service to the **internal** zone and establishing a list of allowed networks. Ansible makes this simple with the firewalld module. Here is an example:

```

- name: Add SSH port to internal zone
  firewallld:
    zone: internal
    service: ssh
    state: enabled
    immediate: yes
    permanent: yes

- name: Add permitted networks to internal zone
  firewallld:
    zone: internal
    source: "{{ item }}"
    state: enabled
    immediate: yes
    permanent: yes
    with_items: "{{ allowed_ssh_networks }}"

- name: Drop ssh from the public zone
  firewallld:
    zone: public
    service: ssh
    state: disabled
    immediate: yes
    permanent: yes

```

This task uses a **with_items** loop with the **allowed_ssh_networks** variable. That variable is defined in the **vars** section of the playbook:

```

vars:
  allowed_ssh_networks:
    - 192.168.122.0/24
    - 10.10.10.0/24

```

In this case, the module restricts access to the **internal** zone to the 10.10.10.0/24 and 192.168.122.0/24 networks. The **immediate** and **permanent** parameters tell the module to apply the rules immediately and add them to firewalld's permanent rules to persist on reboot. You can confirm the configuration by looking at the generated rules.

For more information about firewalld and its configuration, check out Enable Sysadmin's firewalld post.

```
[root@nyc1-webserver-1 ~]# firewall-cmd --list-all --zone=public
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: eth0
  sources:
  services: dhcpv6-client
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:

[root@nyc1-webserver-1 ~]# firewall-cmd --list-all --zone=internal
internal (active)
  target: default
  icmp-block-inversion: no
  interfaces:
  sources: 192.168.122.0/24 10.10.10.0/24
  services: dhcpv6-client mdns samba-client ssh
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

Task 3: Disable unused software and services

With access to SSH locked down, I turn my attention to removing unused software and disabling unnecessary services. Ansible's package and service modules are up to the challenge, as seen below:

```
- name: Remove undesirable packages
  package:
    name: "{{ unnecessary_software }}"
    state: absent

- name: Stop and disable unnecessary services
  service:
    name: "{{ item }}"
    state: stopped
    enabled: no
  with_items: "{{ unnecessary_services }}"
  ignore_errors: yes
```

There are two things to notice about these tasks. First, I again use variables (and loops, for the service task) to keep my playbook short and reusable. These variables have been added to the **vars** section of the playbook:

```
vars:
  allowed_ssh_networks:
    - 192.168.122.0/24
    - 10.10.10.0/24
  unnecessary_services:
    - postfix
    - telnet
  unnecessary_software:
    - tcpdump
    - nmap-ncat
    - wpa_supplicant
```

Task 4: Security policy improvements

After performing several very concrete tasks that improve your newly-provisioned system's security posture. The last thing to do is isn't a technical security improvement—it's a process and legal-oriented control. Always ensure that systems have a *login banner* and *message of the day* to alert users about acceptable use policy. This ensures that there is no doubt that access to a system is restricted: it's printed right in the login banner and MOTD.

Installing these files is a perfect activity for Ansible's file module since they rarely change across all the servers.

```
- name: Set a message of the day
  copy:
    dest: /etc/motd
    src: etc/motd
    owner: root
    group: root
    mode: 0644

- name: Set a login banner
  copy:
    dest: "{{ item }}"
    src: etc/issue
    owner: root
    group: root
    mode: 0644
  with_items:
    - /etc/issue
    - /etc/issue.net
```

The contents of the actual files are a simple, terse explanation of authorized access and acceptable use. You will want to work with your company's legal team to ensure that the messaging is right for your organization.


```
$ cat files/etc/issue
```

```
Use of this system is restricted to authorized users only, and all  
use is subjected to an acceptable use policy.
```

```
IF YOU ARE NOT AUTHORIZED TO USE THIS SYSTEM, DISCONNECT NOW.
```

```
$ cat files/etc/motd
```

```
THIS SYSTEM IS FOR AUTHORIZED USE ONLY
```

```
By using this system, you agree to be bound by all policies found at  
https://wiki.example.com/acceptable\_use\_policy.html. Improper use of  
this system is subject to civil and legal penalties.
```

```
All activities are logged and monitored.
```

The full playbook used in this article is included below:

```
---
- hosts: all
  vars:
    allowed_ssh_networks:
      - 192.168.122.0/24
      - 10.10.10.0/24
    unnecessary_services:
      - postfix
      - telnet
    unnecessary_software:
      - tcpdump
      - nmap-ncat
      - wpa_supplicant
  tasks:
    - name: Perform full patching
      package:
        name: '*'
        state: latest

    - name: Add admin group
      group:
        name: admin
        state: present

    - name: Add local user
      user:
        name: admin
        group: admin
        shell: /bin/bash
        home: /home/admin
        create_home: yes
        state: present
```

```
- name: Add SSH public key for user
  authorized_key:
    user: admin
    key: "{{ lookup('file', '~/.ssh/id_rsa.pub') }}"
    state: present

- name: Add sudoer rule for local user
  copy:
    dest: /etc/sudoers.d/admin
    src: etc/sudoers.d/admin
    owner: root
    group: root
    mode: 0440
    validate: /usr/sbin/visudo -csf %s

- name: Add hardened SSH config
  copy:
    dest: /etc/ssh/sshd_config
    src: etc/ssh/sshd_config
    owner: root
    group: root
    mode: 0600
    notify: Reload SSH

- name: Add SSH port to internal zone
  firewallld:
    zone: internal
    service: ssh
    state: enabled
    immediate: yes
    permanent: yes
```

```
- name: Add permitted networks to internal zone
  firewallld:
    zone: internal
    source: "{{ item }}"
    state: enabled
    immediate: yes
    permanent: yes
    with_items: "{{ allowed_ssh_networks }}"

- name: Drop ssh from the public zone
  firewallld:
    zone: public
    service: ssh
    state: disabled
    immediate: yes
    permanent: yes

- name: Remove undesirable packages
  package:
    name: "{{ unnecessary_software }}"
    state: absent

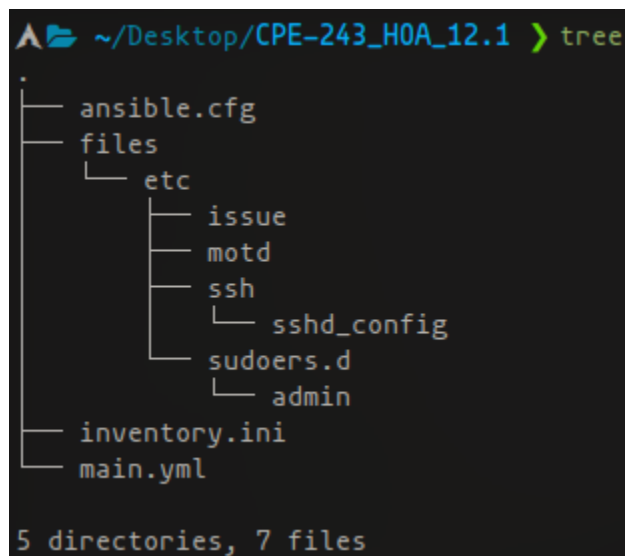
- name: Stop and disable unnecessary services
  service:
    name: "{{ item }}"
    state: stopped
    enabled: no
    with_items: "{{ unnecessary_services }}"
    ignore_errors: yes
```

```
- name: Set a message of the day
  copy:
    dest: /etc/motd
    src: etc/motd
    owner: root
    group: root
    mode: 0644

- name: Set a login banner
  copy:
    dest: "{{ item }}"
    src: etc/issue
    owner: root
    group: root
    mode: 0644
  with_items:
    - /etc/issue
    - /etc/issue.net

handlers:
- name: Reload SSH
  service:
    name: sshd
    state: reloaded
```

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



```
~/Desktop/CPE-243_H0A_12.1 > tree
.
├── ansible.cfg
├── files
│   └── etc
│       ├── issue
│       ├── motd
│       ├── ssh
│       │   ├── sshd_config
│       │   └── admin
│       └── sudoers.d
├── inventory.ini
└── main.yml

5 directories, 7 files
```

Figure 1. The screenshot above shows the file and directory structure.

Files and Its Contents	
Filename	Screenshots
ansible.cfg	 <pre>^[[~/Desktop/CPE-243_H0A_12.1 > cat ansible.cfg [defaults] inventory = inventory.ini host_key_checking = False deprecation_warnings = False private_key_file = ~/.ssh/id_rsa</pre>
inventory.ini	 <pre>^[[~/Desktop/CPE-243_H0A_12.1 > cat inventory.ini [ubuntu] 192.168.240.132 ansible_user=managed_node_one [centos] 192.168.240.131 ansible_user=managed_node_two</pre>

main.yml

```
~/Desktop/CPE-243_H0A_12.1 > cat main.yml
---
- hosts: all
  become: true
  vars:
    allowed_ssh_networks:
      - 192.168.240.0/24
    unnecessary_services:
      - postfix
      - telnet
    unnecessary_software:
      - tcpdump
      - nmap-ncat
      - wpa_supplicant

  tasks:

    - name: Perform full patching
      package:
        name: '*'
        state: latest

    - name: Add admin group
      group:
        name: admin
        state: present

    - name: Add local user
      user:
        name: admin
        group: admin
        shell: /bin/bash
        home: /home/admin
        create_home: yes
        state: present

    - name: Add SSH public key for user
      authorized_key:
        user: admin
        key: "{{ lookup('file', '~/ssh/id_rsa.pub') }}"
        state: present

    - name: Add sudoer role for local user
      copy:
        dest: /etc/sudoers.d/admin
        src: etc/sudoers.d/admin
        owner: root
        group: root
        mode: 0440
        validate: /usr/sbin/visudo -csf %s

    - name: Add hardened SSH config
      copy:
        dest: /etc/ssh/sshd_config
        src: etc/ssh/sshd_config
        owner: root
        group: root
        mode: 0600
        notify: Reload SSH

    - name: Installing firewall in ubuntu
      apt:
        name: firewall
        state: present
        when: ansible_distribution == "Ubuntu"

    - name: Add SSH port to internal zone
```

```

firewalld:
  zone: internal
  service: ssh
  state: enabled
  immediate: yes
  permanent: yes

- name: Add permitted networks to internal zone
firewalld:
  zone: internal
  source: "{{ item }}"
  state: enabled
  immediate: yes
  permanent: yes
  with_items: "{{ allowed_ssh_networks }}"

- name: Drop ssh from the public zone
firewalld:
  zone: public
  service: ssh
  state: disabled
  immediate: yes
  permanent: yes

- name: Remove undesirable packages
package:
  name: "{{ unnecessary_software }}"
  state: absent

- name: Stop and disable unnecessary services
service:
  name: "{{ item }}"
  state: stopped
  enabled: no
  with_items: "{{ unnecessary_services }}"
  ignore_errors: yes

- name: Set a message of the day
copy:
  dest: /etc/motd
  src: etc/motd
  owner: root
  group: root
  mode: 0644

- name: Set a login hammer
copy:
  dest: "{{ item }}"
  src: etc/issue
  owner: root
  group: root
  mode: 0644
  with_items:
    - /etc/issue
    - /etc/issue.net

handlers:
- name: Reload SSH
  service:
    name: sshd
    state: reloaded

```

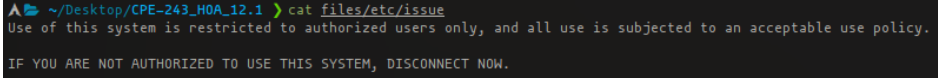
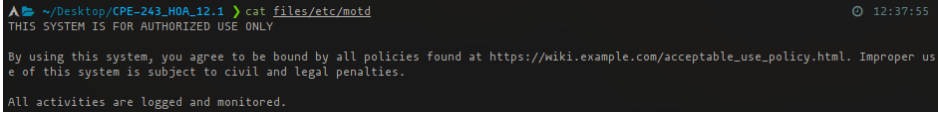
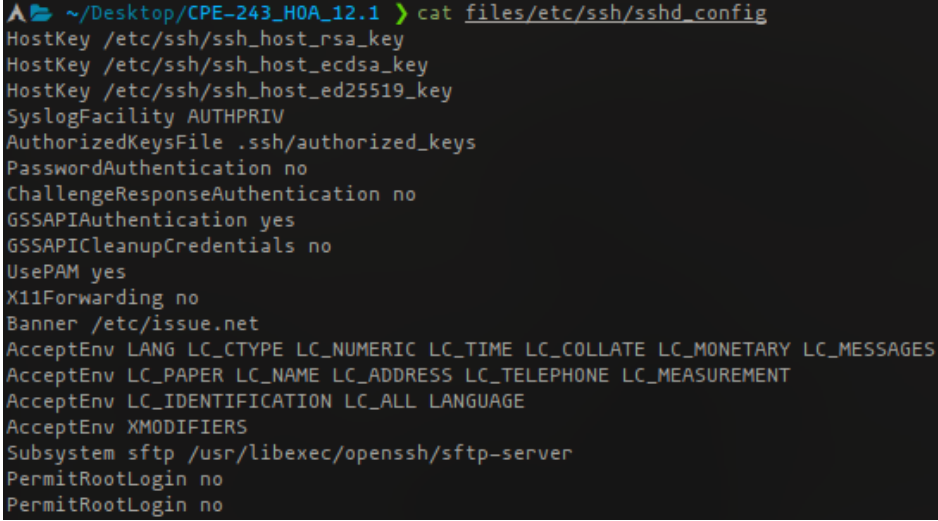
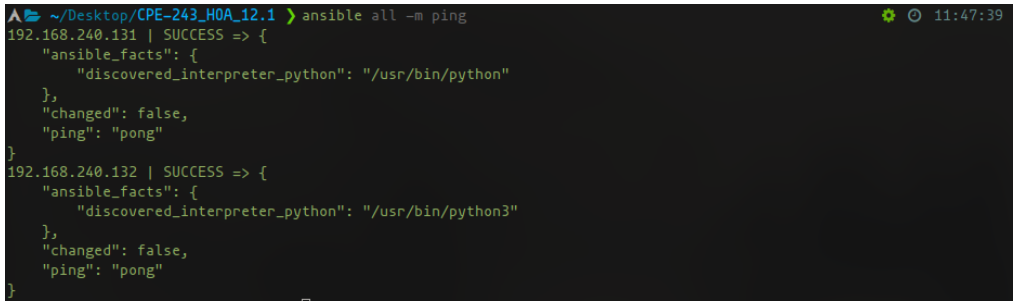

files/etc/issue	 <pre> ~/Desktop/CPE-243_H0A_12.1 > cat files/etc/issue Use of this system is restricted to authorized users only, and all use is subjected to an acceptable use policy. IF YOU ARE NOT AUTHORIZED TO USE THIS SYSTEM, DISCONNECT NOW. </pre>
files/etc/motd	 <pre> ~/Desktop/CPE-243_H0A_12.1 > cat files/etc/motd THIS SYSTEM IS FOR AUTHORIZED USE ONLY By using this system, you agree to be bound by all policies found at https://wiki.example.com/acceptable_use_policy.html. Improper use of this system is subject to civil and legal penalties. All activities are logged and monitored. </pre>
files/etc/ssh/sshd_config	 <pre> ~/Desktop/CPE-243_H0A_12.1 > cat files/etc/ssh/sshd_config HostKey /etc/ssh/ssh_host_rsa_key HostKey /etc/ssh/ssh_host_ecdsa_key HostKey /etc/ssh/ssh_host_ed25519_key SyslogFacility AUTHPRIV AuthorizedKeysFile .ssh/authorized_keys PasswordAuthentication no ChallengeResponseAuthentication no GSSAPIAuthentication yes GSSAPICleanupCredentials no UsePAM yes X11Forwarding no Banner /etc/issue.net AcceptEnv LANG LC_CTYPE LC_NUMERIC LC_TIME LC_COLLATE LC_MONETARY LC_MESSAGES AcceptEnv LC_PAPER LC_NAME LC_ADDRESS LC_TELEPHONE LC_MEASUREMENT AcceptEnv LC_IDENTIFICATION LC_ALL LANGUAGE AcceptEnv XMODIFIERS Subsystem sftp /usr/libexec/openssh/sftp-server PermitRootLogin no PermitRootLogin no </pre>

Table 1. The table above shows the codes and screenshot.



```

~/Desktop/CPE-243_H0A_12.1 > ansible all -m ping
192.168.240.131 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": false,
  "ping": "pong"
}
192.168.240.132 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}

```

Figure 2. The screenshot above shows the proof that the ssh connection between two VM are successful.

```

~ /Desktop/CPE-243_H0A_12.1 > ansible-playbook -K main.yml 13:17:04
BECOME password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.168.240.131]
ok: [192.168.240.132]

TASK [Perform full patching] *****
ok: [192.168.240.132]
ok: [192.168.240.131]

TASK [Add admin group] *****
ok: [192.168.240.131]
ok: [192.168.240.132]

TASK [Add local user] *****
ok: [192.168.240.132]
ok: [192.168.240.131]

TASK [Add SSH public key for user] *****
ok: [192.168.240.132]
ok: [192.168.240.132]

TASK [Add sudoer role for local user] *****
ok: [192.168.240.132]
ok: [192.168.240.131]

TASK [Add hardened SSH config] *****
ok: [192.168.240.132]
ok: [192.168.240.131]

TASK [Installing firewalld in ubuntu] *****
skipping: [192.168.240.131]
ok: [192.168.240.132]

TASK [Add SSH port to internal zone] *****
ok: [192.168.240.131]
ok: [192.168.240.132]

TASK [Add permitted networks to internal zone] *****
ok: [192.168.240.132] => (item=192.168.240.0/24)
ok: [192.168.240.131] => (item=192.168.240.0/24)

TASK [Drop ssh from the public zone] *****
ok: [192.168.240.132]
ok: [192.168.240.131]

TASK [Remove undesirable packages] *****
ok: [192.168.240.132]
ok: [192.168.240.131]

TASK [Stop and disable unnecessary services] *****
ok: [192.168.240.131] => (item=postfix)
failed: [192.168.240.132] (item=postfix) => {"ansible_loop_var": "item", "changed": false, "item": "postfix", "msg": "Could not find the requested service postfix: host"}
failed: [192.168.240.131] (item=telnet) => {"ansible_loop_var": "item", "changed": false, "item": "telnet", "msg": "Could not find the requested service telnet: host"}
...ignoring
failed: [192.168.240.132] (item=telnet) => {"ansible_loop_var": "item", "changed": false, "item": "telnet", "msg": "Could not find the requested service telnet: host"}
...ignoring

TASK [Set a message of the day] *****
ok: [192.168.240.132]
ok: [192.168.240.131]

TASK [Set a login hammer] *****
ok: [192.168.240.132] => (item=/etc/issue)
ok: [192.168.240.131] => (item=/etc/issue)
ok: [192.168.240.132] => (item=/etc/issue.net)
ok: [192.168.240.131] => (item=/etc/issue.net)

PLAY RECAP *****
192.168.240.131 : ok=14 changed=0 unreachable=0 failed=0 skipped=1 rescued=0 ignored=1
192.168.240.132 : ok=15 changed=0 unreachable=0 failed=0 skipped=0 rescued=0 ignored=1

~ /Desktop/CPE-243_H0A_12.1 > 1m 38s 13:18:45

```

Figure 3. The figure above shows the output after running the playbook.

Note: The task named “Stop and disable unnecessary services” ignores turning off postfix and telnet services because these packages are already removed in the previous task named “Remove undesirable packages”.

Reflections:

Answer the following:

1. How do you implement security using Ansible?

- Ansible can help implement security in your infrastructure through configuration management, role-based access control, encryption of sensitive data, compliance checking, security hardening, and auditing. By leveraging these features and best practices, you can ensure that your infrastructure is configured securely and is compliant with security policies and regulations.

Conclusions:

In conclusion, Ansible is a powerful tool that can help implement security in your infrastructure by automating hardening tasks that can be accomplished in a repeatable way. By creating an Ansible playbook that you can run against your systems on first boot, you can ensure that your systems are configured securely and in compliance with security policies and regulations. With Ansible's features and best practices, you can easily implement hardening tasks such as disabling unnecessary services, installing security patches, configuring firewalls, and more. By leveraging Ansible's automation capabilities, you can save time and ensure that your systems are configured in a consistent and secure manner.