ACTIVIDAD 3: DESPLIEGUE DE SEGURIDAD Y AUDITORÍA EN TECHCORP

Cristóbal Suárez Abad

OPTATIVA - 2º ASIR

Objetivo: *Crear un Playbook único que configure la seguridad del servidor, gestione usuarios de forma masiva y realice una auditoría de sistema, aplicando variables, lógica condicional y gestión de eventos.*

El Escenario *La dirección de TechCorp ha detectado intentos de acceso no autorizados. Se requiere endurecer la seguridad (Hardening) de forma flexible. No queremos que los nombres de los usuarios o las rutas de los logs estén "quemados" en el código, ya que podrían cambiar en el futuro.*

Instrucciones Generales

1. *Modifica tu inventario (hosts, grupos) según sea necesario para las pruebas.*
2. *Crea un playbook llamado* `security_hardening.yml`.
3. ***Ejecución:*** *Lanza el playbook y entrega una captura del "RECAP" final.*

Requerimientos Técnicos del Playbook

1. ***Registro estado del host:***
   - *Ejecutar el comando* `uptime` *y registrar el resultado.*
2. ***Creación de usuarios:***
   - *Usando una variable* `audit_team` *con una lista de usuarios(*`auditor1`, `auditor2`, `auditor3`*), genera los usuarios y asegúrate de que pertenezcan al grupo* `sudo`.
3. ***Registros y Condicionales:***
   - *Verificar si el archivo* `/etc/ssh/sshd_config` *existe.*
   - *Solo si existe, buscar la línea* `PermitRootLogin` *y cambiarla a* `no`.
4. ***Controladores (Handlers):***
   - *Si se modifica la configuración de SSH, notificar a un handler llamado* `Reiniciar SSH` *para que actúe al final.*
5. ***Lógica de Grupos:***
   - *Si el host es* `prod` *-> Instalar* `fail2ban`.
   - *Si el host es* `db_servers` *-> Asegurar que* `nginx` *esté detenido y deshabilitado en caso de que esté instalado.*
6. ***Registro en Log:***
   - *Escribir el resultado del* `uptime` *en la ruta definida por tu variable* `log_path`*(que tiene que tener la dirección* `/tmp/techcorp_audit.txt` *), pero solo si la tarea de uptime fue exitosa.*

Índice:

1. Modificación de inventario

- Modificamos el archivo **host_vars/db_master.yml**

En el ejercicio anterior especificamos el usuario **root**. Sin embargo, para evitar conflictos, lo hemos cambiado a **'ansible'**; aunque también habría funcionado simplemente eliminando el archivo.

ansible_user: ansible
ansible_password: password
db_port: 5432

```
  GNU nano 6.2
#ansible_user: root
ansible_user: ansible
# Esto en teoria solucionaba el
ansible_password: password
db_port: 5432
```

- Preparación de variables:

Creamos las variables "*audit_team" y "log_path"*. Para ello las definimos en el archivo "group_vars/all.yml".
Además, creamos una variable para que los usuarios "ansible" puedan usar "sudo": "ansible_become_password: password".

```
  GNU nano 6.2
ansible_user: ansible
ansible_password: password
empresa: TechCorp

# Ejercicio 3
log_path: /tmp/techcorp_audit.txt
audit_team:
  - auditor1
  - auditor2
  - auditor3
# Password de sudo para nodos
ansible_become_password: password
```

2. Instalación “manual” de sudo en los nodos administrados.

Los nodos administrados no tienen “sudo” y por lo tanto no podrán instalar paquetería. Tampoco pueden instalar el propio paquete de sudo los usuarios “ansible”. Para solucionarlo, usamos el siguiente comando:

- Instalación de “sudo”. Desde un terminal del host usamos:

```
docker exec -u 0 ansible-node1 apt-get update && docker exec -u 0 ansible-node1 apt-get install -y sudo
docker exec -u 0 ansible-node2 apt-get update && docker exec -u 0 ansible-node2 apt-get install -y sudo
docker exec -u 0 ansible-node3 apt-get update && docker exec -u 0 ansible-node3 apt-get install -y sudo
```

- Añadir el usuario “ansible” a la lista de “sudoers” para que pueda usar el comando “sudo”:

```
docker exec -u 0 ansible-node1 sh -c "echo 'ansible ALL=(ALL) NOPASSWD:ALL' >> /etc/sudoers"
docker exec -u 0 ansible-node2 sh -c "echo 'ansible ALL=(ALL) NOPASSWD:ALL' >> /etc/sudoers"
docker exec -u 0 ansible-node3 sh -c "echo 'ansible ALL=(ALL) NOPASSWD:ALL' >> /etc/sudoers"
```

3.security_hardening.yml

- name: Actividad 3 - Seguridad y Auditoría TechCorp
  hosts: all
  become: yes # Activado globalmente porque ya hemos instalado y configurado "sudo"

```
- name: Actividad 3 - Seguridad y Auditor  a TechCorp
  hosts: all
  become: yes  # Activado globalmente porque ya hemos instalado y configurado "sudo"
```

“hosts: all”: Indica que el playbook se ejecutará en todos los nodos definidos en el inventario: “hosts.yml”.

Definido en ansible.cfg:

```
inventory = ./hosts.yml
```

“become: yes” Activa la escalada de privilegios. Se ejecutarán las tareas como root.

tasks:
  # 1. **Auditoría de tiempo de actividad**
  - name: Ejecutar comando uptime
    command: uptime
    register: uptime_result
    changed_when: false

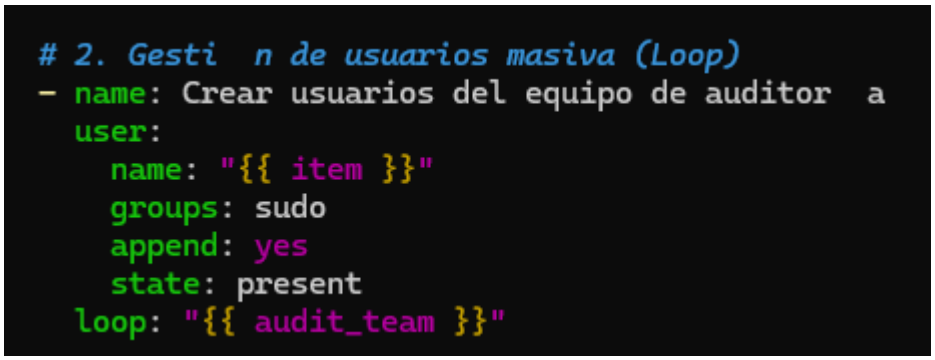
```
tasks:
  # 1. Auditor  a de tiempo de actividad
  - name: Ejecutar comando uptime
    command: uptime
    register: uptime_result
    changed_when: false
```

“command: uptime”: Ejecuta el comando “uptime”.

“register: uptime_result”: Guarda la salida del comando en una variable interna para usarla más tarde.

“changed_when: false”: Evita que Ansible marque esta tarea como "Changed" (amarillo). Leer el tiempo de actividad no modifica el sistema, por lo que debe aparecer siempre como "OK" (verde).

```
# 2. Gestión de usuarios masiva (Loop)
- name: Crear usuarios del equipo de auditoría
  user:
    name: "{{ item }}"
    groups: sudo
    append: yes
    state: present
  loop: "{{ audit_team }}"
```



“user”: hace referencia al módulo “user”[1] para la creación de cuentas de usuario.

“name: "{{ item }}"”: El nombre usa “item” que se usará en el “loop”, el cuál a su vez está definido en “audit_team”.

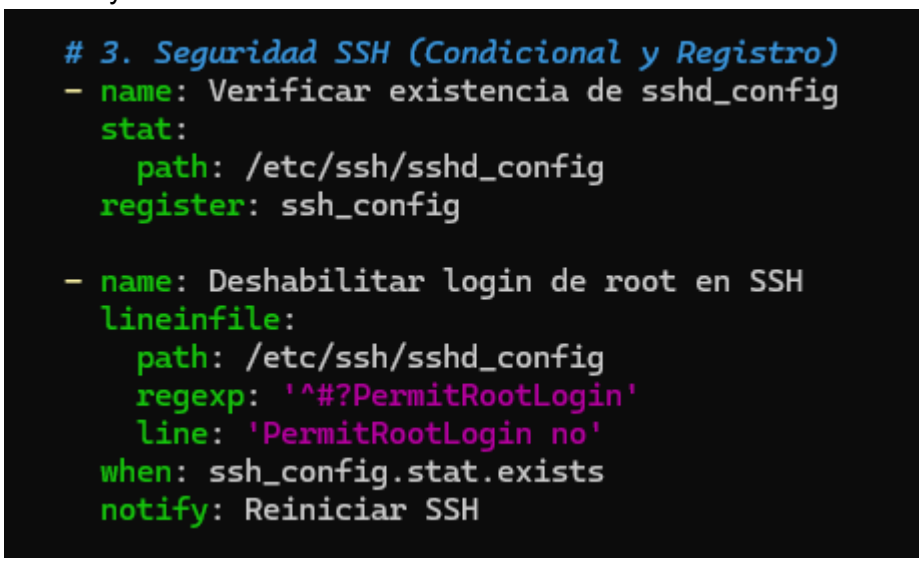
“groups: sudo”: Añade a cada usuario al grupo de administradores.

“state: present”: Asegura que, si el usuario no existe, se cree; y si existe, se mantenga.

[1] https://docs.ansible.com/projects/ansible/2.9/modules/user_module.html#user-module

3. **Seguridad SSH (Condicional y Registro)**

```
- name: Verificar existencia de sshd_config
  stat:
    path: /etc/ssh/sshd_config
  register: ssh_config

- name: Deshabilitar login de root en SSH
  lineinfile:
    path: /etc/ssh/sshd_config
    regexp: '^#?PermitRootLogin'
    line: 'PermitRootLogin no'
  when: ssh_config.stat.exists
  notify: Reiniciar SSH
```



“stat”: Módulo[2] para comprobar que el archivo “/etc/ssh/sshd_config” existe. Registra el resultado en una variable “ssh_config".

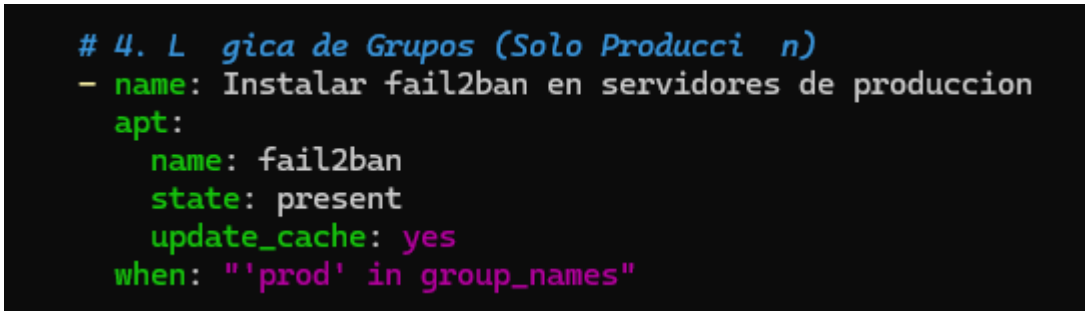
“lineinfile”: Módulo[3] para buscar y/o reemplazar líneas en archivos. Busca el archivo con “path”, después busca la línea con una expresión regular “regexp” y por último la cambia con “line”. En este caso la cambia a “PermitRootLogin no”, lo cuál evita que se pueda usar SSH con ROOT.

“when”: Si previamente se ha creado la variable “ssh_config”, se ejecutará “lineinfile”.

“notify”: Si el archivo se modifica, "avisa" al “handler” para que reinicie el servicio al final del Playbook. Hemos definido al final del documento a dicho “handler”.

[2] https://docs.ansible.com/projects/ansible/2.9/modules/stat_module.html#stat-module
[3] https://docs.ansible.com/projects/ansible/2.9/modules/lineinfile_module.html#lineinfile-module

```
# 4. Lógica de Grupos (Solo Producción)
- name: Instalar fail2ban en servidores de produccion
  apt:
    name: fail2ban
    state: present
    update_cache: yes
  when: "'prod' in group_names"
```



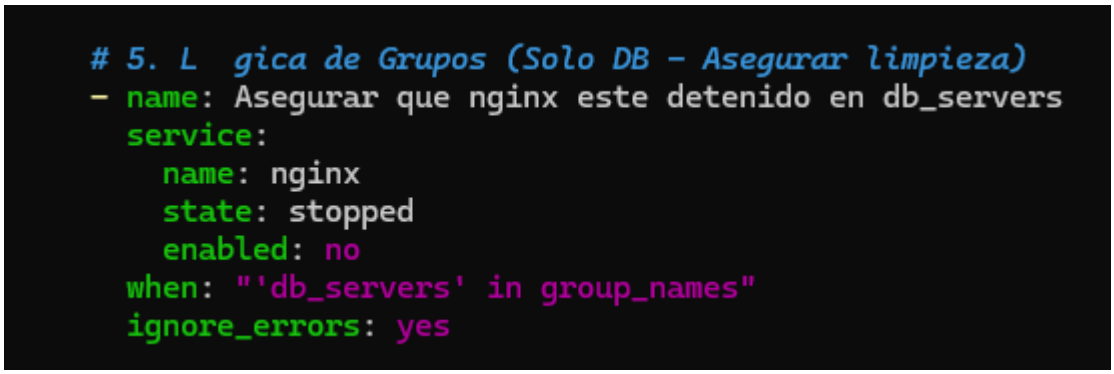
“apt”: Usa el módulo para instalar[4] paquetería.

“name”: nombre del paquete.
“state: present”: hace referencia a instalar el paquete.
“update:cache: yes”: Es el equivalente a “apt-get update”. Lo realiza antes de instalar paquetería.

“when”: Solo se ejecutará en los nodos que pertenecen al grupo prod del inventario.

[4] https://docs.ansible.com/projects/ansible/2.9/modules/apt_module.html#apt-module

```
# 5. Lógica de Grupos (Solo DB - Asegurar limpieza)
- name: Asegurar que nginx este detenido en db_servers
  service:
    name: nginx
    state: stopped
    enabled: no
  when: "'db_servers' in group_names"
  ignore_errors: yes
```



“service”: Módulo[5] para controlar servicios.

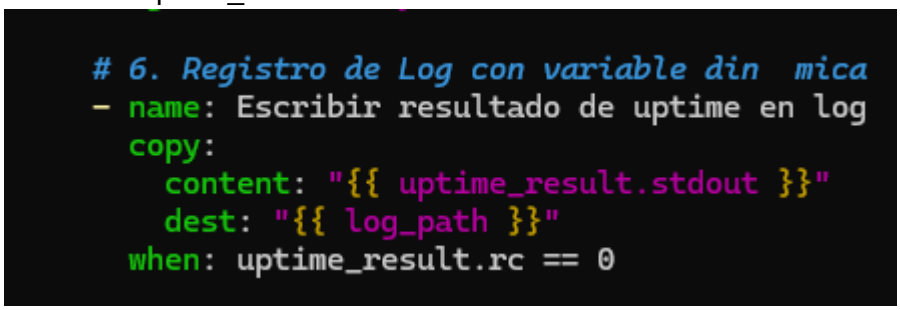
“name”: nombre del servicio.
“state”: Estado en el que queremos poner el servicio, en nuestro caso será parado.
“enabled”: Indicamos si queremos que el servicio esté activo al arrancar el nodo.

“when”: Cuando los nodos son del grupo “db_servers”.

“ignore_errors: yes”: Lo he puesto porque en “db_master” no está instalado “nginx” (no sé el motivo). De esta manera evitamos que al dar error se detenga la ejecución del Playbook.

[5] https://docs.ansible.com/projects/ansible/2.9/modules/service_module.html#service-module

```
# 6. Registro de Log con variable dinámica
- name: Escribir resultado de uptime en log
  copy:
    content: "{{ uptime_result.stdout }}"
    dest: "{{ log_path }}"
  when: uptime_result.rc == 0
```



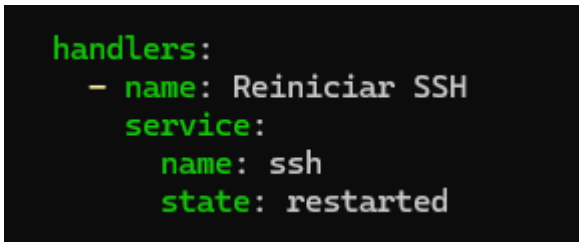
“copy”: Módulo[6] para copiar de la máquina local o remota a un directorio de la máquina remota.

“content”: Toma el texto que guardamos en la primera tarea.
“dest”: Indica el destino donde se debe copiar. El que hemos definido antes.

“when”: Solo escribe el archivo si el comando “uptime” se ejecutó correctamente (los códigos de error suelen ser “0” para vacío o nulo).

[6] https://docs.ansible.com/projects/ansible/2.9/modules/copy_module.html#copy-module

```
handlers:
  - name: Reiniciar SSH
    service:
      name: ssh
      state: restarted
```



Este es el “handler” al que llamamos para reiniciar SSH si hemos configurado la restricción de acceso a ROOT.

“service”: El módulo para controlar servicios.

“ssh”: Nombre del servicio.
“state”: Estado en el que queremos que esté. En este caso lo vamos a reiniciar.

4. Recap.

Usamos: ansible-playbook security_hardening.yml

```
PLAY [Actividad 3 - Seguridad y Auditoría TechCorp] ********************

TASK [Gathering Facts] *************************************************
ok: [db_master]
ok: [node2]
ok: [node3]

TASK [Ejecutar comando uptime] *****************************************
ok: [node2]
ok: [db_master]
ok: [node3]

TASK [Crear usuarios del equipo de auditoría] **************************
ok: [node2] => (item=auditor1)
ok: [db_master] => (item=auditor1)
ok: [node3] => (item=auditor1)
ok: [db_master] => (item=auditor2)
ok: [node3] => (item=auditor2)
ok: [node2] => (item=auditor2)
ok: [db_master] => (item=auditor3)
ok: [node2] => (item=auditor3)
ok: [node3] => (item=auditor3)
```

Como se ha comentado antes, hay un error en el nodo “db_master” → “node1”, el cual no tiene “nginx” instalado. Por eso no puede comprobar que dicho servicio esté instalado.

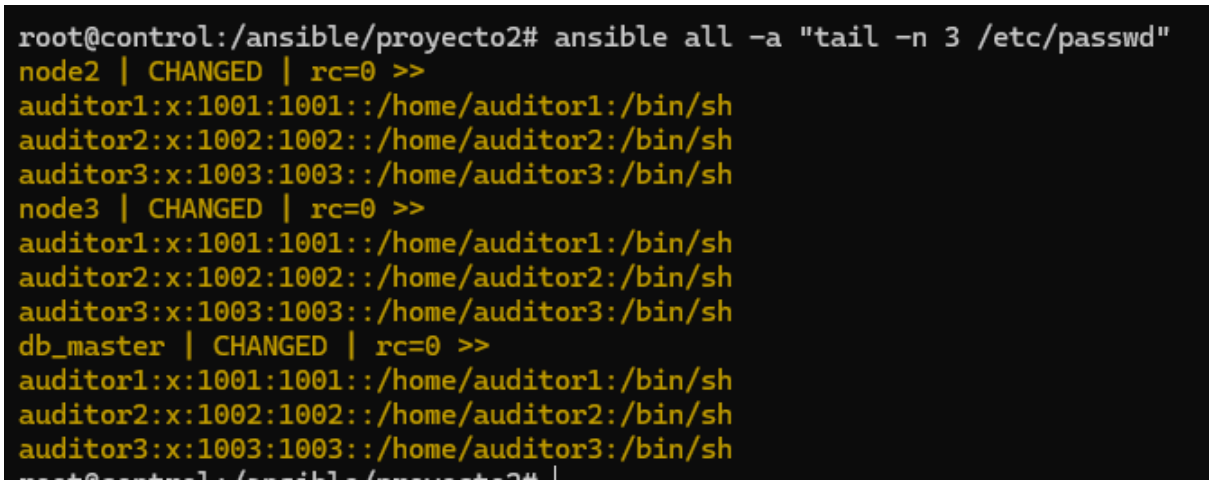
```
TASK [Verificar existencia de sshd_config] ********************
ok: [db_master]
ok: [node2]
ok: [node3]

TASK [Deshabilitar login de root en SSH] **********************
ok: [node2]
ok: [node3]
ok: [db_master]

TASK [Instalar fail2ban en servidores de produccion] **********
skipping: [node2]
ok: [db_master]
ok: [node3]

TASK [Asegurar que nginx este detenido en db_servers] *********
skipping: [node2]
skipping: [node3]
fatal: [db_master]: FAILED! => changed=false
  msg: 'Could not find the requested service nginx: host'
...ignoring

TASK [Escribir resultado de uptime en log] ********************
changed: [db_master]
changed: [node2]
changed: [node3]
```

```
PLAY RECAP *********************************************************************************************************
db_master                  : ok=8    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=1
node2                      : ok=6    changed=1    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
node3                      : ok=7    changed=1    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
```

5. Comprobaciones extras.

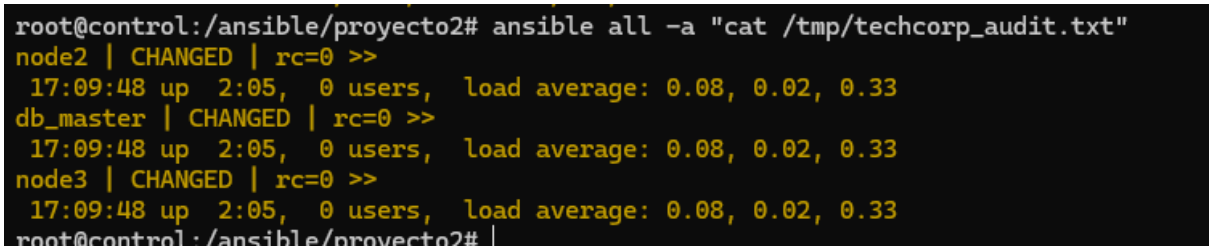
- Comprobación de que los usuarios del “audit_team” han sido creados:

ansible all -a "tail -n 3 /etc/passwd"



- Verificar el log de “auditoría”:

ansible all -a "cat /tmp/techcorp_audit.txt"



- Verificar instalación de “fail2ban”: Ten en cuenta que el node2 no debe tenerlo instalado.

ansible prod -a "which fail2ban-server"