Simulacre Prova Final UF2 Sistemes

PAS PREVI: En tots els exercicis s'haurà d'usar, per provar els "playbooks" demanats, una màquina virtual amb un sistema Fedora Workstation instal·lat (incloent els paquets "ansible-core" i "ansible-collection-community-general") i amb un usuari amb capacitat per executar *sudo* (que serà l'usuari que executarà els "playbooks")

NOTA: Tots els "playbooks" demanats als exercicis s'hauran d'executar contra la mateixa màquina virtual local (mitjançant el "connection plugin" "local" -o, el que és el mateix, indicant com a màquina "víctima" l'anomenada "localhost" o "127.0.0.1"-). Això implica que a aquesta màquina no caldrà que hi hagi el servei SSH encès ni tampoc que hi aparegui cap nom/IP concret al seu inventari. Tampoc caldrà indicar cap usuari amb el paràmetre "-u" d'*ansible* (o la directiva "remote_user:" d'*ansible-playbook*), ni tampoc el paràmetre "-k", perquè el connector "local" pren l'usuari que executa la comanda *ansible/ansible-playbook* (que assumirem que es diu "usuari") com l'usuari a emprar en l'execució dels playbooks. Però sí els paràmetre *-b* i *-K* si fos necessari.

1.-Escriu un playbook Ansible que, en executar-lo, faci les següents tasques (al final podràs comprovar si s'han realitzat correctament totes elles observant la sortida de les comandes *systemctl status nftables* i/o *sudo nft list ruleset*):

a) Instal.li el paquet "nftables" a la màquina "víctima" fent servir el mòdul *dnf*

b) Copïi en la carpeta "/etc/nftables" de la màquina "víctima" (que en aquest cas és la mateixa màquina que la màquina controladora) un fitxer anomenat "myrules.nft", ubicat a la carpeta "/opt" de la màquina controladora i amb el contingut indicat a la "Nota" següent (el qual consisteix en un conjunt de regles pròpies del tallafocs Nftables; el seu significat concret és irrellevant per l'exercici però han de ser sintàctiament vàlides) fent servir el mòdul *copy* (tingues en compte, d'altra banda, que el valor per defecte de l'opció *force=* d'aquest mòdul és "yes")

NOTA: El fitxer "/opt/myrules.nft" ha de tenir el següent contingut:

```
table ip filter {
    chain input { type filter hook input priority filter; policy accept; }
    chain output { type filter hook output priority filter; policy accept; }
}
```

c) Afegeixi la línia *include "/etc/nftables/myrules.nft"* al final de l'arxiu "/etc/sysconfig/nftables.conf" de la màquina "víctima (fent servir el mòdul *lineinfile*)

d) Inicïi el servei "nftables" a la màquina "víctima" i també l'habiliti (fent servir en els dos casos el mòdul *systemd* en una sola tasca)

2.-Escriu un playbook Ansible que, en executar-lo, faci les següents tasques (al final podràs comprovar si s'han realitzat correctament totes elles observant la sortida de les comandes *systemctl status pum.timer, systemctl list-timers* o *journalctl -u pum.service*):

a) Creï un fitxer (buit) anomenat "pum.timer" i un altre (també buit) anomenat "pum.service", ambdós ubicats a la carpeta "/etc/systemd/system" de la màquina "víctima" (fent servir el mòdul *file* i un bucle)

b) Ompli de contingut tant l'arxiu "pum.timer" com l'arxiu "pum.service" (fent servir en els dos casos el mòdul *blockinfile*, en tasques separades), concretament així:

Contingut "pum.timer"

```
[Unit]
Description=Timer Hola
[Timer]
OnBootSec=1m
OnUnitActiveSec=1m
[Install]
WantedBy=default.target
```

Contingut "pum.service"

```
[Unit]
Description=Service Hola
[Service]
ExecStart=date
```

c) Inicïi el timer "pum.timer" a la màquina "víctima" i també l'habiliti (fent servir en els dos casos el mòdul *systemd* en una sola tasca)

3.-Escriu un playbook Ansible que, en executar-lo, faci les següents tasques (al final podràs comprovar si s'han realitzat correctament totes elles observant què passa executes *ssh root@127.0.0.1* , on l'usuari "root" hauria de tenir assignada prèviament una contrasenya)

a) Instal.li el paquet "openssh-server" a la màquina "víctima" fent servir el mòdul *dnf*

b) Modifiqui la configuració establerta a l'arxiu "/etc/ssh/sshd_config" de la màquina "víctima" per a què la línia *PermitRootLogin* valgui *yes* , fent servir el mòdul *replace*

c) Reinicii el servei "sshd" de la màquina víctima només si la tasca anterior ha canviat efectivament la línia en qüestió, si no, no (és a dir, implementa aquesta tasca en forma de "handler").

4.-En aquest exercici se't demanarà crear un usuari "convidat" i modificar la configuració necessària al Gdm per tal d'habilitar l'accés al sistema a aquest usuari (tal com ja vam veure en un exercici del document sobre "display managers" de la UF anterior). Podràs comprovar si has realitzat correctament aquest exercici tancant sessió: hauràs de veure el nou usuari "convidat"; si llavors inicies sessió amb ell -sense contrasenya- i proves de fer algun canvi en el sistema, quan tanquis sessió i tornis a iniciar-la amb el mateix usuari "convidat" hauràs de comprovar que els canvis realitzats s'han perdut. Els passos a fer són, concretament, els següents:

a) Crea un usuari amb nom "guest", una contrasenya qualsevol i la carpeta "/tmp/guest" com a una carpeta personal fent servir el mòdul *user*

b) Per a què no aparegui a cada inici de sessió de l'usuari convidat l'assistent de benvinguda del Gnome, elimina el paquet "gnome-tour" fent servir el mòdul *dnf*

c) Fes que l'script "/etc/gdm/PostLogin/Default" tingui exactament el següent contingut (fent servir el mòdul *blockinfile*) i fes-lo executable (fent servir l'opció *mode* del mòdul *file*)

```
#!/bin/sh
guestuser="guest"
if [ "$USER" = "$guestuser" ]; then
        mkdir /tmp/"$guestuser"
        cp /etc/skel/.* /tmp/"$guestuser"
        chown -R "$guestuser":"$guestuser" /tmp/"$guestuser"
fi
exit 0
```

cII) Fes que l'script "/etc/gdm/PostSession/Default" tingui exactament el següent contingut (fent servir el mòdul *blockinfile*) i fes-lo executable (fent servir l'opció *mode* del mòdul *file*)

```
#!/bin/sh
guestuser="guest"
if [ "$USER" = "$guestuser" ]; then
        rm -rf /tmp/"$guestuser"
fi
exit 0
```

d) Cal fer que Gdm permeti els inicis de sessió sense contrasenya (com a mínim per l'usuari "guest"). Això es pot fer afegint la línia *auth sufficient pam_succeed_if.so user = guest* al principi de l'arxiu "/etc/pam.d/gdm-password". Fes-ho emprant el mòdul *lineinfile*

5.-Instal·la a la màquina virtual de treball el paquet "httpd" i posa en marxa el servei corresponent amb *sudo systemctl start httpd*. Des de la teva màquina real, accedeix varis cops a la pàgina principal d'aquest servidor HTTP (mitjançant la comanda *curl* o amb el navegador, és igual). Un cop fet això, executa la comanda *ansible-playbook miplay.yml -K -e log_file=/var/log/httpd/access_log* (on el playbook utilitzat és el següent) i contesta les següents preguntes:

```
- name: Un playbook
  hosts: localhost
  vars:
  - carpeta: "{{ lookup('env', 'HOME') }}"
  tasks:
  - shell: date +"%Y%m%d%H%M%S"
    register: current_date_time
  - shell: grep -i "$(date +%d/%b/%Y)" {{ log_file }}
    become: yes
    ignore_errors: true
    environment:
        LANG: en_US.UTF-8
    register: log_messages
  - file: path={{ ansible_env.HOME}}/pepe state=directory mode=0755
  - copy: content={{ log_messages.stdout_lines }} dest={{ ansible_env.HOME}}/pepe/logs-
{{ ansible_hostname }}-{{ current_date_time.stdout }}.txt
  - fetch: src={{ ansible_env.HOME}}/pepe/logs-{{ ansible_hostname }}-{{ current_date_time.stdout }}.txt
dest={{ carpeta }}/logs-{{ ansible_hostname }}-{{ current_date_time.stdout }}.txt flat=yes
```

a) ¿Quin és el valor de la variable "current_date_time" i per a què es fa servir en el playbook anterior?

b) ¿Quin és el valor de la variable "log_messages" i per a què es fa servir en el playbook anterior?

c) ¿Quina cadena busca la comanda *grep* indicada a la segona tasca, i on?

cII) ¿Per què és important la línia *ignore_errors: true* en executar aquesta comanda *grep*? ¿I l'establiment de la variable LANG a "en_US.UTF-8"? (en aquest sentit, compara el format que té la data en la sortida de la comanda *date +%d/%b/%Y* en el teu idioma actual i en l'arxiu "/var/log/httpd/access_log")

d) ¿Per a què serveix la tercera tasca (la que fa servir el mòdul "file") i, en concret, què representa el valor de la variable *{{ ansible_env.HOME }}* ?

e) Descriu amb les teves paraules què fa la tasca que fa servir el mòdul "copy", paràmetre a paràmetre (fixa't que no es fan servir els més habituals)

f) Descriu amb les teves paraules què fa la tasca que fa servir el mòdul "fetch", paràmetre a paràmetre. A partir d'aquí, dedueix i explica quina diferència de significat hi ha entre l'expressió *{{ ansible_env.HOME }}* i *{{ lookup('env', 'HOME') }}*

g) ¿Què passaria si a la comanda *ansible-playbook* d'aquest exercici s'hi afegís el paràmetre *-b*, i per què? Pista: observa quin és el valor de *{{ ansible_env.HOME }}* en aquest cas

h) Resumeix en una frase la possible utilitat d'aquest playbook (és a dir, amb quin objectiu el faries servir)

6.-a) ¿Què fa el següent playbook i quin missatge final mostra per pantalla (i quan)?

```
- hosts: localhost
  tasks:
  - wait_for:
        path: /opt/hola.txt
    register: arxiu
- debug: msg="Sí, {{ arxiu.path }} "
```

aII) Escriu (i adjunta com a resposta) un playbook que utilitzi el mateix mòdul *wait_for* però ara per comprovar si el port 443 del servidor "www.github.com" respon (mostrant-se, en cas que sí, un missatge qualsevol a pantalla).

b) ¿Què fan els següents playbooks i per què? ¿Quina comanda *ansible-playbook* faries servir per executar-los?

Contingut "loop.yml"

```
- hosts: localhost
  tasks:
  - include_tasks: "{{ ansible_lo.ipv4.address }}.yml"
```

Contingut "127.0.0.1.yml"

```
- debug: msg="Hola"
- include_tasks: 127.0.0.1.yml
```

bII) ¿I aquests? ¿Per a què serveix l'expressió "| default(-1)" ? ¿I l'expressió "| int"?

Contingut "loop.yml"

```
- hosts: localhost
  tasks:
  - include_tasks: "{{ ansible_lo.ipv4.address }}.yml"
```

Contingut "127.0.0.1.yml"

```
- set_fact:
    counter: "{{ counter | default(-1) | int + 1 }}"
- debug: msg="{{ counter | int }}"
- include_tasks: 127.0.0.1.yml
  when: "counter | int < 5"
```

c) ¿Què fa el següent playbook? Interpreta la seva sortida

```
- hosts: localhost
  gather_facts: false
  vars:
    format: "ext4"
  tasks:
  - shell: "cat /etc/fstab"
    register: contingut
  - debug: msg="{{ item }}"
    loop: "{{ contingut.stdout_lines }}"
    when: format in item
```

d) ¿Què fa el següent playbook (el qual fa servir el filtre "split", no vist fins ara)? Raona el que fa cada tasca

```
- hosts: localhost
  vars:
    emails: ["hello@gritfy.com", "steve@google.com", "mark@ibm.com", "dave@meta.com"]
    usernames: []
  tasks:
- debug: msg="{{emails[0] | split('@')}}"
- debug: msg="L'usuari {{ item | split('@') | first }} té el domini {{ item | split('@') | last }}"
  loop: "{{emails}}"
- set_fact: usernames="{{ usernames + [ item | split(':') | first ] }}"
  loop: "{{ lookup('file', '/etc/passwd').splitlines()}}"
- debug: var=usernames
```

e) ¿Què fa el següent playbook? Interpreta la seva sortida

```
- hosts: localhost
  become: true
  tasks:
  - find: paths="/etc/ansible" patterns="*.txt"
    register: llista
  - file: path="{{ item.path }}" state=absent
    loop: "{{ llista.files }}"
```

eII) ¿Quina diferència hi ha entre el playbook anterior i el següent?

```
- hosts: localhost
  become: true
  tasks:
  - find: paths="/etc/ansible" patterns="^.*\.txt" use_regex=true
    register: llista
  - file: path="{{ item.path }}" state=absent
    loop: "{{ llista.files }}"
```

7.-Llegeix la documentació del mòdul "stat" oficial d'Ansible (i en especial els exemples i l'apartat "Return Values") per tal d'escriure un playbook que comprovi la data d'última modificació ("mtime") del fitxer "/etc/shadow"; i si aquesta és major que el valor del fact "ansible_date_time.epoch" menys un dia (és a dir, si algun usuari va actualitzar la seva contrasenya ahir), el playbook haurà de generar un missatge d'error (mitjançant el mòdul "fail").

> NOTA: Tingues present que hauràs de convertir el valor "ansible_date_time.epoch" (que és cadena) a tipus numèric per poder fer la comparació demanada

8 . - Escriu un playbook Ansible que, en executar-lo, comprovi dues dades (de la manera que vulguis, ja sigui mitjançant "facts", o mòduls Ansible específics o bé a partir de la sortida de comandes de shell...):

a) La quantitat de memòria RAM lliure que té la/les màquina/es "víctima" inspeccionada/es (en el nostre cas, només serà "localhost"): si aquesta és major de 2GB, haurà d'iniciar el servei "httpd" però si és menor caldrà que mostri un missatge per pantalla com aquest: "HTTP Server not started: free memorysize is less than 2G"

b) La quantitat d'espai de disc dur lliure que té el sistema de fitxers muntat a "/" de la/les màquina/es "víctima" inspeccionada/es (en el nostre cas, només serà "localhost"): si aquesta és major de 20GB, haurà d'instal·lar el paquet "nmap" però si és menor caldrà que mostri un missatge per pantalla com aquest: "Nmap not installed: free disk space in root filesystem is less than 20G"

> NOTA: Existeixen solucions molt més sofisticades i especialitzades que Ansible per portar un control exhaustiu de les característiques hardware de tot un parc d'ordinador. Exemples són **Glpi** (https://glpi-project.org), **Fusion-Inventory** (http://fusioninventory.org) o **OCS-Inventory** (https://ocsinventory-ng.org), entre d'altres. No obstant, tots ells necessiten per recopilar la informació la instal·lació d'un agent específic a les màquines a inspeccionar