BSIT 4D IAS 102 Finals

Using any video recording software(Zoom, Obs etc.), demonstrate the use of the following Linux security commands. Make sure your face will appear in the recorded video. Create a folder(your fullname - lastname_firstname_mid) in this google drive and upload your recorded video(https://drive.google.com/drive/folders/1npq81kfrmfpk1el84RmOZBXAeiUtq3h3?usp=sharing) 1 video recording each item(7 video recording).

1. File System and Navigation

pwd: Print working directory.
ls: List directory contents.
    -l: Detailed listing.
    -a: Show hidden files.
cd: Change directory.
    cd ..: Move up one directory.
    cd ~: Go to the home directory.
mkdir: Create a new directory.
touch: Create a new file.
rm: Remove files or directories.
    -r: Recursive (for directories).
    -f: Force (without confirmation).
cp: Copy files or directories.
mv: Move or rename files or directories.

2. Networking

ifconfig: Display network interface information.
ping: Test network connectivity.
netstat
    -a: Show both listening and non-listening sockets.
    -t: Show only TCP connections.
    -u: Show only UDP connections.
    -n: Display numerical addresses instead of resolving hostnames.
    -l: Show only listening sockets.
    -p: Display the PID and program name.
    -r: Display the kernel routing table.
ss: Display socket statistics.

Network: using nano text editor open the file below

```
#nano /etc/netplan/00-installer-config.yaml
```

```
# Setting dynamic IP Address
```

```
network:
  version: 2
  ethernets:
    enp0s3:
      dhcp4: yes

# Setting static IP Addess
network:
   ethernets:
        enp0s3:
            dhcp4: no
            addresses: [192.168.2.150/24]
            gateway4: 192.168.2.1
            nameservers:
                              addresses: [8.8.8.8, 192.168.2.1]
```

Execute the netplan:

```
#sudo netplan apply
```

3. Securing SSH is crucial for protecting your server from unauthorized access.

Change Default SSH Port (Optional, but Recommended):
Open the SSH configuration file:

```
sudo nano /etc/ssh/sshd_config
```

Find the line #Port 22 and change it to a different port (e.g., Port 2222). Remove the # to uncomment the line.
Save the file and exit the editor.

Restart the SSH service:

```
sudo systemctl restart ssh
```

Disable Root Login:

In the same sshd_config file, find the line PermitRootLogin and set it to no:

```
PermitRootLogin no
```

Save and exit the file.

Restart the SSH service:

```
sudo systemctl restart ssh
```

Set Up SSH Key Authentication:

Generate an SSH key pair on your local machine (if you haven't already):

```
ssh-keygen -t rsa
```

Copy the public key to your server:

```
ssh-copy-id username@your_server_ip
```

Disable password-based authentication (optional but recommended):

Open /etc/ssh/sshd_config and set PasswordAuthentication to no. Restart the SSH service.

Limit User Access:
Only allow specific users to SSH into the server by editing the sshd_config file:

```
sudo nano /etc/ssh/sshd_config
```

Add or modify the line:

```
AllowUsers username
```

Save and exit the file.

Restart the SSH service:

```
sudo systemctl restart ssh
```

Enable Two-Factor Authentication (Optional but Recommended):

4. Setting up Uncomplicated Firewall (UFW) on Ubuntu is a straightforward process. UFW provides an easy way to manage iptables rules for basic server security.

Here's a step-by-step guide with examples:

Install UFW:

```
$ sudo apt update
$ sudo apt install ufw
```

To check the status of UFW, use the following command:

```
$ sudo ufw status
```

It should be inactive by default, and you'll see a message like Status: inactive.

Allow SSH (if using a different port, replace 22 with your custom port):
Allow SSH traffic to your server:

```
$ sudo ufw allow 22/tcp
```

If you've changed the default SSH port (not recommended unless necessary for security reasons), replace 22 with your custom port.

Allow Other Necessary Services:
Allow any other services you want to be accessible. For example, if you have a web server:

```
$ sudo ufw allow 80/tcp  # Allow HTTP
$ sudo ufw allow 443/tcp # Allow HTTPS
```

Enable UFW:
Once you've configured the rules, enable UFW:

```
$ sudo ufw enable
```

You'll see a warning that enabling UFW may disrupt existing SSH connections. Type y and press Enter to proceed.

Check Status and Rules:
Verify that UFW is active and that the rules are set correctly:

```
$ sudo ufw status
```

You should see a list of allowed services along with their corresponding ports.

Additional Rules (Optional):
Depending on your specific setup, you may need to allow other services like FTP, mail, or custom applications. Use similar commands to allow them.
For example, to allow FTP:

```
$ sudo ufw allow 21/tcp
```

Deny or Limit Access from Specific IPs (Optional):
You can also limit access to specific IP addresses or ranges. For example, to allow SSH access only from a specific IP:

```
$ sudo ufw allow from 192.168.1.100 to any port 22
```

To deny access:

```
$ sudo ufw deny from 192.168.1.101 to any
```

Disable UFW (if necessary):
If you need to temporarily disable UFW, you can do so with the command:

```
$ sudo ufw disable
```

To re-enable it later, use:

```
$ sudo ufw enable
```

To deny port

```
sudo ufw deny 80
```

Remember, UFW provides a basic level of firewall protection. For advanced configurations or specific needs,
you may want to look into iptables or other firewall solutions. Always test your rules to ensure they don't lock you out of your server.

5. Fail2ban is a tool that helps protect against brute-force attacks by monitoring logs for repeated login failures and blocking suspicious IP addresses. Here are step-by-step instructions with examples for setting up Fail2ban on Ubuntu:

Install Fail2ban:

```
sudo apt update
sudo apt install fail2ban
```

Copy the Configuration File:
Create a local copy of the Fail2ban configuration file:

```
sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
```
This allows you to make custom configurations without affecting the default settings.
Edit the Configuration:
Open the jail.local file in a text editor:

```
sudo nano /etc/fail2ban/jail.local
```
Find the [sshd] section. This is the jail for SSH.

Example Configuration for SSH:

ini
Copy code
```
[sshd]
enabled = true
port = ssh
filter = sshd
logpath = /var/log/auth.log
maxretry = 3
banaction = ufw
```
enabled = true: This enables the jail.

port = ssh: Specifies the service to monitor (in this case, SSH).
filter = sshd: Refers to the filter used to parse log entries (predefined filter).
logpath = /var/log/auth.log: Path to the log file where authentication attempts are recorded.
maxretry = 3: Sets the number of login attempts before an IP is banned.
banaction = ufw: Specifies that Fail2ban should use UFW to ban IPs.

You can adjust these values to suit your specific needs.

Create Custom Filters (Optional):
If you need to create custom filters to match specific log patterns, you can create them in /etc/fail2ban/filter.d/.
Start Fail2ban:
```
sudo systemctl start fail2ban
sudo systemctl enable fail2ban
```
This starts Fail2ban and enables it to start on boot.

Check Status:
You can check the status of Fail2ban to ensure it's running:
```
sudo fail2ban-client status
```
This should show you information about the jails and their status.

Check Banned IPs:
To view currently banned IPs, you can use:
```
sudo fail2ban-client status sshd
```

This will show you the list of banned IP addresses for the SSH jail.

Test Fail2ban (Optional):
You can test Fail2ban by intentionally causing login failures. After a few failed attempts, you should see the IP being banned.

Remember, Fail2ban is a powerful tool, and misconfigurations could potentially lock you out of your server. Always test your configurations and ensure you have an alternative way to access your server in case of any issues.

6. Setting up Two-Factor Authentication (2FA) with Google Authenticator on Ubuntu involves several steps. Here's a step-by-step guide with examples:

Install Google Authenticator:
Install the Google Authenticator PAM module:
```
sudo apt update
sudo apt install libpam-google-authenticator
```
Configure SSH to Use 2FA:
Open the SSH daemon configuration file:
```
sudo nano /etc/ssh/sshd_config
```

Add the following line at the end:
```
ChallengeResponseAuthentication yes
```
Save and close the file.

Restart the SSH service:
```
sudo systemctl restart ssh
```

Set Up Google Authenticator for a User:
Switch to the user for whom you want to enable 2FA:
```
su - username
```
Run the google-authenticator command:
```
google-authenticator
```

Follow the prompts to set up 2FA. You'll be prompted to scan a QR code using the Google Authenticator app on your mobile device. This will link the app to your user account.

After scanning the QR code, you'll receive a set of emergency scratch codes. Keep these in a safe place.

Configure PAM for Google Authenticator:

Open the PAM configuration file for SSH:
```
sudo nano /etc/pam.d/sshd
```

Add the following line at the top:
auth required pam_google_authenticator.so
Save and close the file.

Test 2FA:
Log out of your SSH session and try to log in again. You should now be prompted for both your password and the verification code from the Google Authenticator app.
Backup and Recovery (Important):

7. Securing SSH is crucial for protecting your server from unauthorized access.

Change Default SSH Port (Optional, but Recommended):
Open the SSH configuration file:
sudo nano /etc/ssh/sshd_config
Find the line #Port 22 and change it to a different port (e.g., Port 2222). Remove the # to uncomment the line.
Save the file and exit the editor.

Restart the SSH service:
sudo systemctl restart ssh
Disable Root Login:

In the same sshd_config file, find the line PermitRootLogin and set it to no:
PermitRootLogin no
Save and exit the file.

Restart the SSH service:
sudo systemctl restart ssh
Set Up SSH Key Authentication:

Generate an SSH key pair on your local machine (if you haven't already):
ssh-keygen -t rsa
Copy the public key to your server:

ssh-copy-id username@your_server_ip
Disable password-based authentication (optional but recommended):

Open /etc/ssh/sshd_config and set PasswordAuthentication to no. Restart the SSH service.

Limit User Access:
Only allow specific users to SSH into the server by editing the sshd_config file:
sudo nano /etc/ssh/sshd_config
Add or modify the line:

AllowUsers username

Save and exit the file.

Restart the SSH service:

```
sudo systemctl restart ssh
```

Enable Two-Factor Authentication (Optional but Recommended):