ssladmin_tasks_2025

Induction 2025

Instructions

- No extra points for completing the tasks faster. So take your time.
- (But keep in mind that you will be burning through your credits for the `VM`. So don't take forever ;P)
- You can **search for things online** if you have doubts If you cant find a solution online, Ask the others from your batch
- If you still cant find a solution or have doubts, Ask a **senior admin**
- **Please spend atleast 5-10 minutes** searching for a solution yourself before asking for help from others
- You can ask others for help but **don't ask them to do it for you.**
- You **need not** do the Optional tasks, they are just for helping you learn more.
- There is no order, but do tasks 1,2 and 3 before the rest. This is to first setup a firewall and an ssh connection so that you can work in secure environment. You need to do 1,2,3 to pass

Tasks

1. Initial Setup (4 pts)

1. **Create an Ubuntu VM in Azure:** (2pts)
    - Login with NITC Mail id and claim student benefits (either through github pro pack or azure student id), or debit card connect in AWS (they have free tier VMs).
    - Use any of the following Ubuntu versions: 20.04, 22.04.

- Update the IP of your VM in the shared document, you will be assigned a domain for that IP.

2. **System Updates and Security:** (2pts)
    - Update the system packages.
    - Set up unattended upgrades to ensure the system always receives the latest security updates.

2. Enhanced SSH Security (6pts)

1. **SSH Configuration:** (1+1+2+1+1pts)
    - Disable root login.
    - Disable password-based authentication.
    - Enable and configure public key authentication.
    - Restrict SSH access to specific IP addresses (e.g., your local IP or a specific range).
    - Set up fail2ban to protect against brute-force attacks. **(NEW)**
    - Add our key so that we can ssh in as a user with superuser privileges(will be shared later)
2. **Multi-Factor Authentication (MFA):** *[Optional]*
    - Install and configure Google Authenticator or any other MFA tool to secure SSH login. **(NEW)**

3. Firewall and Network Security (4pts)

Warning, make sure you dont lock yourself out by blocking ssh. You would most likely have to reset and start over.

1. **Firewall Configuration:**
    - Configure UFW to deny all incoming traffic except for SSH (on a non-default port, e.g., 2222), HTTP, and HTTPS. **(NEW)**
    - Ensure that the UFW logs are enabled for auditing purposes. **(NEW)**
    - Readup more about iptables and how ufw works

2. **Intrusion Detection System (IDS):** *[Optional]*
    - Install and configure an IDS like Snort or Suricata. **(NEW)**
    - Set up rules to monitor and alert on suspicious activities.**(NEW)**

4. User and Permission Management (7 Pts)

1. **User Setup:** (3pts)
    - Create the following users with the specified permissions:
        - `exam_1, exam_2, exam_3`: Regular users with access only to their home directories.
        - `examadmin`: User with root privileges.
        - `examaudit`: User with read-only access to all user home directories.
2. **Home Directory Security:** (2pts)
    - Ensure each user's home directory is only accessible by that user.
    - Set up quotas to limit the disk space each user can use. **(NEW)**
3. **Backup Script:** (2pts)
    - Create a script to back up the home directories of all `exam_*` users.
    - The script should run daily and store backups in a secure, compressed format.
    - Ensure that only `examadmin` can run the script.

5. Web Server Deployment and Secure Configuration (10pts)

Install NginX.

1. **Reverse Proxy Configuration:** (7pts)
    - Set up NginX as a reverse proxy for applications running on the VM.
    - Ensure that the applications is not directly accessible from the internet. These apps must run from a non-privileged user. Create a non privileged user before you proceed.

    - Get `app1` from here. Verify the sha256 signature from here. `app1` runs (`chmod +x app1; ./app1`) on port 8008 and it prints the path it gets from the http request
    - Get `app2` from here. This is the IsSSLOpen app, it runs on port 3000.
- The task is to setup an `nginx` reverse proxy such that: (1.5+1.5+4pts)
    - https://x.ssl.airno.de/server1/ should print

      `SSL Onboarding. path: /server1/`
    - https://x.ssl.airno.de/server2/ should print

      `SSL Onboarding. path: /`
    - https://x.ssl.airno.de/sslopen should open *IsSSLOpen* app. Create a token for us to edit the page. (Read the docs for that issslopen)
- Make sure these apps are not exposed to the internet directly, they should only be accessed through https (443 port) via NginX

2. **Content Security Policy (CSP):** (3pts)
    - Configure a robust CSP to mitigate cross-site scripting (XSS) attacks. **(NEW)**

6. Database Security (5pts)

1. **Database Setup:** (2pts)
    - Install MariaDB.
    - Create a database named `secure_onboarding`.
    - Create a user with minimal privileges required to interact with the `secure_onboarding` database.
2. **Database Security: (NEW)** (3pts)
    - Disable remote root login.
    - Ensure that MariaDB is only accessible from localhost and not even from a LAN machine.

- Set up regular automated backups of the database.

7. VPN Configuration (6pts)

1. **VPN Setup:**
    - Install and configure WireGuard as a VPN server.
    - Create VPN credentials for at least two users.
    - Ensure that the VPN allows access to the local network and the internet.
    - Share one credential to the admins for testing

8. Docker Fundamentals and Personal Website Deployment (5pts)

1. **Basic Docker Setup** (2pts)
    - Install Docker on your VM.
    - Configure Docker to start on system boot.
    - Add your user to the `docker` group for non-root access.
    - Test Docker installation by running a simple container (e.g., `hello-world`).
2. **Deploying a Portfolio Website via Docker and Nginx** (3pts)
    - Create a basic portfolio website (can be static HTML/CSS, no points for design).
    - Use Docker to containerize the website.
    - Set up a systemd service or run the container as a **detached daemon**, ensuring it starts automatically on boot.
    - Mount your website content using Docker **volumes** for persistence and easy updates.
    - Apply Linux capabilities like `-cap-add=NET_ADMIN` or permissions (if required for advanced setups).
    - Set up an **Nginx reverse proxy on the host VM (not in Docker)**:
        - Route HTTP traffic from the VM's public IP (port 80) to the website container.

    - Ensure it’s accessible from the browser using the VM’s IP.
- Bonus: Set proper file permissions and ownership in the container and mounted volumes for secure access.

9. Ansible Automation in Dockerized Lab Environment (8pts)

1. **Ansible Setup and Basics** (2pts)

- Create a custom Docker network (`ansible-net`).
- Run 2–3 containers:
    - One **control** container with Ansible installed.
    - One or two **target** containers with OpenSSH server installed.
- Enable SSH access to the target containers from the control container.
- Configure SSH key-based authentication from control to targets.
- Inside the control container:
    - Create an inventory file listing target containers.
    - Write a basic playbook to:
        - Ping the target(s).
        - Check disk space (`df -h`).
        - Show system uptime.

1. **Lab Configuration Management:** (3pts)
    - Create a comprehensive playbook that automates:
        - Installation of specific software packages (e.g., Python 3, git, vim, htop, curl, wget, tmux)
        - Configuration of bash aliases and vim settings for all users
        - Implementation of security policies (configuring UFW rules, SSH hardening, setting up proper file permissions)
    - Use variables, templates, and conditionals to make the playbook adaptable to different lab environments

- Document your playbook with comments explaining each major section

2. **Ansible Roles for Lab Management:** (3pts)
   - Develop at least two reusable roles:
     - A “lab-base” role that sets up the standard lab environment (install updates, configure timezone, set hostname, install monitoring tools like netdata)
     - A “student-workstation” role that configures systems for student use (create student account, install development tools like VSCode, set up programming language environments)
   - Include proper role structure (tasks, handlers, defaults, vars, templates)
   - Add idempotence checks to ensure roles can be run repeatedly without issues
   - Demonstrate the roles by applying them to your VM

Optional Task: Advanced Security Practices (NEW)

1. **AppArmor or SELinux:**
   - Enable and configure AppArmor or SELinux to enforce mandatory access controls on the system.
2. **Logging and Auditing:**
   - Install and configure an auditing tool like auditd.
   - Ensure that all security-related events are logged and can be reviewed.
3. **Log Analysis:**
   - Set up a log analysis tool like ELK Stack (Elasticsearch, Logstash, Kibana) to aggregate and analyze logs for security incidents.

Optional Task: Load Balancer Setup

> Note: This task is just for people who want to try new things. This is more easier if you are familiar with basic concepts of Operating Systems (Scheduling). However it requires much

> less work than you think and is here for getting better perspective of real world web applications.

You are given a simple app which listens for `http` requests. Using your domain, setup an `nginx` reverse proxy for https://x.ssl.airno.de/loadBalancer/ to this app. (Save it as a python file, run it with `python3 app.py`)

- When you type your domain in the browser, something like `8080:y` should be shown where `y` represents the number of times you refreshed/accessed the page. `y` is reset each time you restart the app.
- Make 3 extra copies of the app and rename them according to your wish. Change the port number in each of the programs to `8081,8082,8083`. Now you should have 4 programs with port numbers `8080,8081,8082,8083`. Run all of these programs in parallel (you could use a script for that).
- Modify the reverse proxy to cater the requests to those 4 programs in a **round robin fashion**. This is called load balancing, that is you divide the requests to 4 identical programs running concurrently.

On your first request,

`8080:1` should be received

on your next request,

```
8081:18082:18083:18080:28081:28082:28083:2...
```

This should be the sequence of outputs, each time you refresh the page.

Optional Task: Advanced Docker Containerization (8 pts)

1. **Nextcloud Containerization Project:** (8pts)

   For this task, you will containerize Nextcloud (an open-source file sharing and collaboration platform):

   - **Application Setup:** (3pts)
     - Create a Docker Compose file that defines:

        - Nextcloud container (use the official Nextcloud image)
        - MariaDB container for the database
        - Redis container for caching
        - Nginx container as a reverse proxy in front of Nextcloud
    - Configure proper networking between containers
    - Set up environment variables for database credentials
    - Ensure data persistence with named volumes for both Nextcloud files and database data
- **Configuration and Customization:** (2pts)
    - Enable and configure at least two Nextcloud apps (e.g., Calendar, Contacts, Talk)
    - Configure Nginx with proper caching headers and optimizations
    - Set up proper SSL termination in the Nginx container
- **Security Hardening:** (2pts)
    - Implement security best practices:
        - Use non-root users inside containers
        - Implement resource limits (CPU, memory)
        - Configure read-only file systems where possible
        - Set up hardening options in Nextcloud's config.php
- **Documentation:** (1pt)
    - Create a README.md file documenting:
        - How to deploy your containerized Nextcloud
        - Security measures implemented
        - Backup and restore procedures
        - Potential scaling considerations

2. **Container Orchestration:** *[Optional]*

- Convert your Docker Compose setup to Kubernetes manifests (deployments, services, configmaps)
- Deploy Nextcloud on a local Kubernetes cluster (minikube or k3s)
- Implement horizontal pod autoscaling for the Nextcloud deployment