Name: Brindyn Schultz

Connect VM to VLAN (if applicable)

1. Run ‘`ip link show`’ to retrieve the current network adapter. It should look like ‘`enp0s1`’.
2. Edit the netplan configuration file using the command ‘`sudo nano /etc/netplan/00-installer-config.yaml`’, then duplicate the configuration for the old adapter which should look like ‘`enp0s3`’. Change the duplicate to the current adapter. For example the config file could look like this when finished:

```
network:
  ethernets:
    enp0s3:
      dhcp4: true
    enp0s1:
      dhcp4: true
  version: 2
```

3. Save the modified configuration file using ^X and typing ‘`y`’ to confirm. Restart the system using ‘`sudo reboot`’.
4. Test the connection to the virtual LAN by running ‘`ping 8.8.8.8`’ or ‘`ping google.com`’.

Update the OS (recommended)

1. First retrieve the latest packages using the command ‘`sudo apt update`’.
2. After the latest packages have been retrieved, upgrade the current ones to the new versions using ‘`sudo apt upgrade`’.
3. After that is completed, reboot the system if needed with ‘`sudo reboot`’.
4. Next, upgrade packages with changing dependencies using ‘`sudo apt dist-upgrade`’. This may also update the system's kernel.
5. Lastly, reboot the system using ‘`sudo reboot`’.

Configure Automatic Security Updates (step 1)

1. First explicitly check that the most recent security updates have been installed using ‘`sudo apt install unattended-upgrades`’.
2. Remove any unneeded packages that were automatically installed during the security update by running ‘`sudo apt autoremove`’.

3. Currently the machine is set to only install security updates manually. Using Unattended Upgrades, you can configure the package to install security updates on the machine automatically using ‘`sudo dpkg-reconfigure --priority=low unattended-upgrades`’. Follow the steps on screen to configure it.
4. Enable automatic security updates and set the interval for checking for updates and cleaning cache to be frequent. You can edit the configuration file for this by running ‘`sudo nano /etc/apt/apt.conf.d/10periodic`’.
5. Modify the config file so that it says:
   ```
   APT::Periodic::Update-Package-Lists "1";
   APT::Periodic::Download-Upgradeable-Packages "1";
   APT::Periodic::AutocleanInterval "7";
   ```
   a. This means that the machine checks for updates daily using the parameter `Update-Package-Lists` set to `"1"` day, downloads and installs upgradeable packages using the parameter `Download-Upgradeable-Packages` set to True (`"1"`), and cleans up package cache weekly using the parameter `AutocleanInterval` set to `"7"` days.
6. Save the modified configuration file using ^X and typing ‘`y`’ to confirm.
7. Enable unattended upgrades. You can edit the configuration file for this by running ‘`sudo nano /etc/apt/apt.conf.d/20auto-upgrades`’. Modify the config file so that it says:
   ```
   APT::Periodic::Update-Package-Lists "1";
   APT::Periodic::Unattended-Upgrade "1";
   ```
   a. This means that the machine checks for updates daily using the parameter `Update-Package-Lists` set to `"1"` day and unattended upgrades is enabled using the parameter `Unattended-Upgrades` set to True (`"1"`).
8. Save the modified configuration file using ^X and typing ‘`y`’ to confirm.

Remove Remote Root SSH Login (step 2)

1. First open the SSH server configuration file in an editor using the command ‘`sudo nano /etc/ssh/sshd_config`’.
2. Find the line that says `PermitRootLogin`, and change its value from ‘`yes`’ to ‘`no`’. After making the edit, the line will look like:
   ```
   PermitRootLogin no
   ```
3. Save the modified configuration file using ^X and typing ‘`y`’ to confirm.

4. After saving your changes to the SSH config file, restart the SSH service using '`sudo service ssh restart`'.

Remove Vulnerable Services (step 3)

1. First retrieve a list of all packages installed on the system using 'dpkg -l'.
2. Search through the list for packages that can pose security threats that are not needed for system functionality. I found telnet, ftp, tnftp, cups, avahi, ipp-usb, libsane, and sane. Remove them using '`sudo apt-get purge telnet ftp tnftp cups-browsed cups-core-drivers cups-daemon cups-filters avahi-daemon ipp-usb libsane-common sane-airscan sane-utils`'.
3. After removing these services, remove any unnecessary packages within the system that were using those services with '`sudo apt autoremove`'.
4. Reboot the system using '`sudo reboot`'.

Prevent Listening on Other Ports (step 4)

1. First view all services running that are associated with the system's ports using the command '`sudo ss -tulnp`'.
    a. In my case, the services that were listening were: `mysqld` on ports 33060 and 3306, `systemd-resolve` on port 53, `sshd` on port 22, `cupsd` on port 631, `apache2` on port 80, and both `xinetd` and `inetd` on port 23.
2. The services that are unnecessary are `cups` (`cupsd`) which is a service for managing printing services, and `xinetd` (`xinetd`) which is a daemon for managing `extended inet` services such as insecure legacy file transfer protocols and legacy troubleshooting services. These packages and their configuration files can be removed using '`sudo apt-get purge xinetd telnetd cups`'.
    a. There is one additional insecure service that cannot be removed from the system, which is `inet` (`inetd`). This service manages legacy file transfer protocols such as `telnet` and `ftp`, as well as, legacy troubleshooting services such as `chargen`, `discard`, and `echo`. The reason you cannot remove `inet`, however, is that it manages the system's date and time, making it a dependency for most packages including ones we need.
3. Additionally, `mysql` (`mysqld`) provides mysql services which can be risky for a system. Many web servers no longer use mysql due to cyber risks, so this service and its configurations can be removed, using '`sudo apt-get purge mysql-server mysql-server-8.0 mysql-client mysql-common`'.

    a. If the company using this web server needs mysql for their operations, do not remove it.

4. After removing these services, remove any unnecessary packages within the system that were using those services with ‘`sudo apt autoremove`’.
5. Even though `inet` cannot be removed, it should not communicate with the system’s ports. To do this, we can disable it by using the commands ‘`sudo systemctl stop inetd`’, followed by ‘`sudo systemctl disable inetd`’.
6. Reboot the system using the command ‘`sudo reboot`’, then check the system’s ports using ‘`sudo ss -tulnp`’. You should only see services related to systemd, SSH, and Apache listening.

Strengthen the Password Policy (step 5)

1. This system comes with a password managing package called `cracklib`. We want to configure it. First open the `common-password` configuration file using ‘`sudo nano /etc/pam.d/common-password`’.
2. Remove the line that says ‘`# password requisite pam_deny.so`’. This is a vulnerability, because it means that if any process, including `cracklib`, denies access, the authentication is required to stop.
3. Save the modified configuration file using ^X and typing ‘`y`’ to confirm.
4. Reconfigure the password manager using ‘`sudo pam-auth-update`’.
5. Check the options for ’`cracklib`’ ,’`unix authentication`‘ ,’`register user sessions in the systemd control group hierarchy`’, and ’`create home directory on login`’.
6. Press tab and enter to submit these settings and continue.
7. Now, open the `common-password` configuration file again using ‘`sudo nano /etc/pam.d/common-password`’.
8. Edit the line that says ‘`password requisite pam_cracklib.so retry=3 minlen=8 difok=3`’, and change it to ‘`password requisite pam_cracklib.so retry=3 minlen=16 ucredit=-1 lcredit=-1 dcredit=-1 ocredit=-1 difok=3`’.
    a. This means the system will allow 3 retries (`retry`), the password has a minimum length of 16 characters (`minlen`), the password must have at least 1 uppercase letter (`ucredit`), the password must have at least 1 lowercase letter (`lcredit`), the password must have at least 1 digit (`dcredit`), the password must have at least 1 special character

(`ocredit`), and the password must have at least 3 characters that are different from the old password (`difok`).

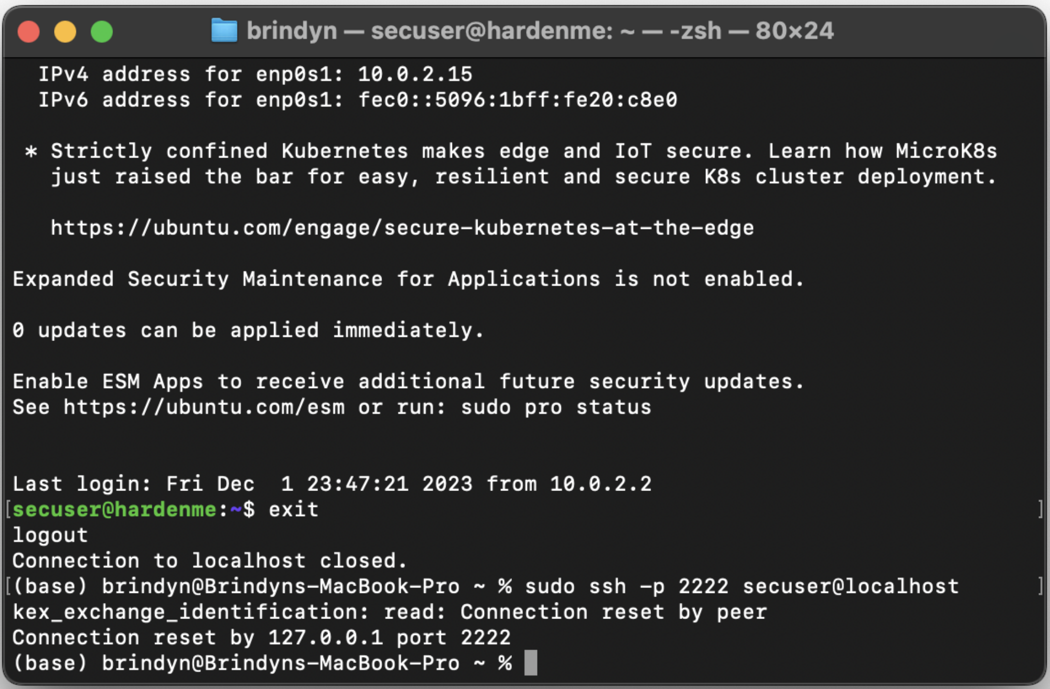
9. Save the modified configuration file using ^X and typing ‘`y`’ to confirm.
10. Edit the file related to login parameter definitions using the command ‘`sudo nano /etc/login.defs`’.
11. Locate the line that says ’`PASS_MAX_DAYS    99999`’ and change it to ‘`PASS_MAX_DAYS    180`’. This will change the maximum number of days a password is valid from 99,999 days, which is 273 years, to 180 days, which is 6 months.
12. Save the modified parameter definition file using ^X and typing ‘`y`’ to confirm.
13. Next we want to list all user accounts to change the passwords of current accounts and remove old accounts. We can list these accounts using the command ‘`awk -F: '$3 >= 1000 {print $1}' /etc/passwd`’.
    a. The parameter ‘`$3 >= 1000`’ checks the column containing UIDs of the accounts (`$3`) to see if they are greater than or equal to 1000. This is true for all user accounts. System accounts have UIDs less than 1000, and we don’t want to remove those (they remove themselves when you purge the package they are related to). For accounts with UIDs greater than or equal to 1000, it will print the column containing their usernames (`$1`).
14. There are three unused user accounts on the system: nobody, employee1, and employee2. Remove them using ‘`sudo userdel -r nobody`‘ ,’`sudo userdel -r employee1`’, and ‘`sudo userdel -r employee2`’.
15. We also want to change the password of secuser, which we can do using ‘`echo 'secuser:Project-Project23' | sudo chpasswd`’.

Configure Firewall (step 6)

1. First install the firewall using the command ‘`sudo apt install ufw`’. This is a high level interface that manages `iptables`.
2. To make firewall rules persistent across reboots install the `iptables-persistent` package using the command ‘`sudo apt-get install iptables-persistent`’.
3. Reset all current `iptables` rules to defaults to get rid of any bad rules. This is done using the command ‘`sudo ufw reset`’
4. Configure basic firewall rules for allowing input SSH, HTTP, and HTTPS activity using the commands ‘`sudo ufw allow ssh`‘ ,’`sudo ufw allow http`’, and ‘`sudo ufw allow https`’.

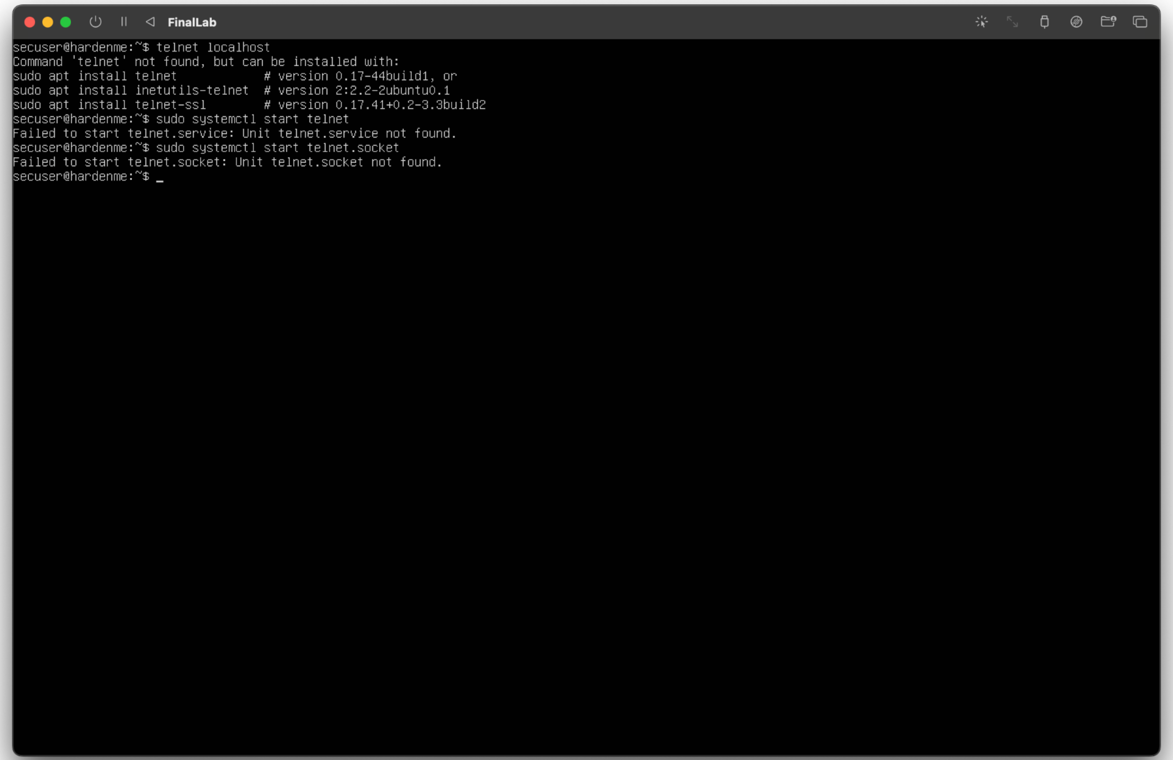
5. Enable the firewall using ‘`sudo ufw enable`’
6. Now you can view all of the `iptables` rules that `ufw` created by using ‘`sudo iptables -L -n -v`’ or a specific set of rules such as ‘`sudo iptables -L INPUT -n -v`’ for all input rules.
    a. To test if the firewall rules are working you can try a dropped input and an accepted input. An example is the command ‘`ping 8.8.8.8`’, which should send packets but not receive them, since we did not make a firewall rule to allow ICMP input. You can test an accepted input using the command ‘`curl http://example.com`’ which should successfully output the html code for the website example.com, since we made a rule to allow http. If either of these do not work as expected, you may have an issue with the output or input rules.
7. Lastly, run ‘`sudo reboot`’ to restart the machine, then run ‘`sudo iptables -L -n -v`’ to ensure the firewall rules are persistent. You can also rerun the firewall rules test again to be extra confident.

Testing



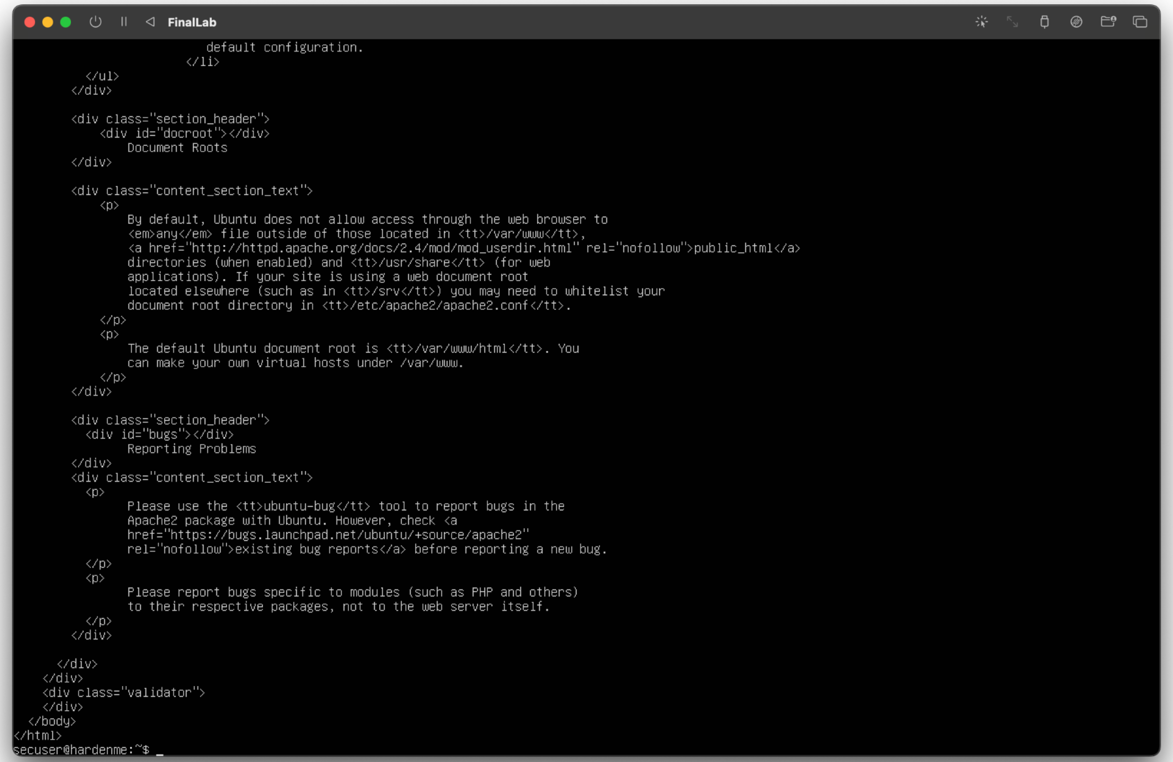
```
  IPv4 address for enp0s1: 10.0.2.15
  IPv6 address for enp0s1: fec0::5096:1bff:fe20:c8e0

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


Last login: Fri Dec  1 23:47:21 2023 from 10.0.2.2
secuser@hardenme:~$ exit
logout
Connection to localhost closed.
(base) brindyn@Brindyns-MacBook-Pro ~ % sudo ssh -p 2222 secuser@localhost
kex_exchange_identification: read: Connection reset by peer
Connection reset by 127.0.0.1 port 2222
(base) brindyn@Brindyns-MacBook-Pro ~ %
```

In this screenshot, I allowed remote root login on the VM, then connected to the machine using ssh, then I disabled remote root login on the VM and tried to connect again, but was no longer able to connect.



```
secuser@hardenme:~$ telnet localhost
Command 'telnet' not found, but can be installed with:
sudo apt install telnet            # version 0.17-44build1, or
sudo apt install inetutils-telnet  # version 2:2.2-2ubuntu0.1
sudo apt install telnet-ssl        # version 0.17.41+0.2-3.3build2
secuser@hardenme:~$ sudo systemctl start telnet
Failed to start telnet.service: Unit telnet.service not found.
secuser@hardenme:~$ sudo systemctl start telnet.socket
Failed to start telnet.socket: Unit telnet.socket not found.
secuser@hardenme:~$ _
```

In this screenshot, I couldn’t connect over telnet, so I tried to start it on the VM side, but since I completely removed it from the machine, it couldn’t start.



```
                    default configuration.
                  </li>
          </ul>
        </div>

        <div class="section_header">
            <div id="docroot"></div>
                Document Roots
        </div>

        <div class="content_section_text">
            <p>
                By default, Ubuntu does not allow access through the web browser to
                <em>any</em> file outside of those located in <tt>/var/www</tt>,
                <a href="http://httpd.apache.org/docs/2.4/mod/mod_userdir.html" rel="nofollow">public_html</a>
                directories (when enabled) and <tt>/usr/share</tt> (for web
                applications). If your site is using a web document root
                located elsewhere (such as in <tt>/srv</tt>) you may need to whitelist your
                document root directory in <tt>/etc/apache2/apache2.conf</tt>.
            </p>
            <p>
                The default Ubuntu document root is <tt>/var/www/html</tt>. You
                can make your own virtual hosts under /var/www.
            </p>
        </div>

        <div class="section_header">
          <div id="bugs"></div>
                Reporting Problems
        </div>
        <div class="content_section_text">
          <p>
                Please use the <tt>ubuntu-bug</tt> tool to report bugs in the
                Apache2 package with Ubuntu. However, check <a
                href="https://bugs.launchpad.net/ubuntu/+source/apache2"
                rel="nofollow">existing bug reports</a> before reporting a new bug.
          </p>
          <p>
                Please report bugs specific to modules (such as PHP and others)
                to their respective packages, not to the web server itself.
          </p>
        </div>

      </div>
    </div>
    <div class="validator">
    </div>
  </body>
</html>
secuser@hardenme:~$ _
```

In this screenshot, I tested port 80 with the command ‘`curl http://localhost:80`’, which worked, so port 80 is open and functioning.

Discussion

During the process of hardening this machine, I had a hard time removing vulnerable services preventing unnecessary services from listening on ports. This was because I am not familiar with many of the services, so I had to spend a lot of time researching the services and running down rabbit holes of dependencies to figure out what could be removed. I also got tripped up by the inet package and how it couldn’t be removed but had services within it that could. Things I think I did really well are configuring automatic updates, patching the machine, and configuring firewall rules. I found that the ufw package made handling firewall rules so much easier, as well as, there were a lot of online resources for updating the machine and patching. When it came to the hardening process, the main area that was really up to my choice was in what packages to remove. I did a lot of research on vulnerabilities and found that certain services like telnet, ftp, cups, and avahi were on the system but are notorious for being insecure. Future steps to take in securing the system would be to include an IDS and antivirus software, as well as many of the security tools used by OpenBSD to mitigate attacks such as DoS attacks and SYN flood attacks. It may also be a good idea to force multi factor authentication for logging into the system such as a fingerprint scanner with remote alternatives like one-time codes. Another good security feature would be entire disk encryption to protect the file system in the case of a cyber attack.