**Laboratory Exercise X – Host Based Network Security Basics - Part-2**

Due Date: Date

Points Possible: Number of points out of total course points or recommended percent of course grade.

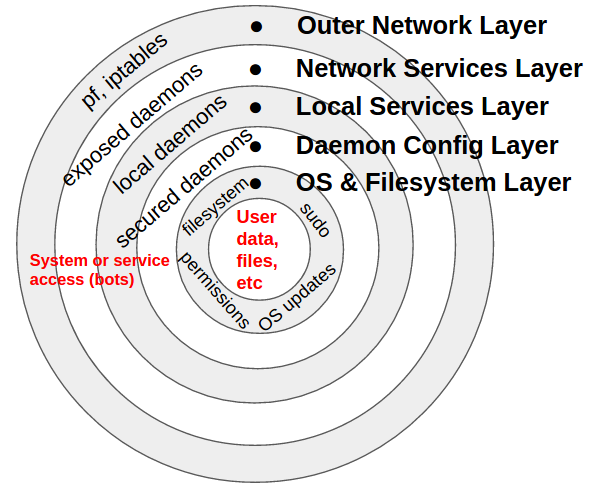
**1. Overview**

This laboratory exercise will provide some hands-on *layered defense* experience with hardening a *LAMP* (Linux, Apache, MySQL, PHP) server by examining what ports, IPs and services are exposed to the network, and work on addressing and securing the outstanding network security issues layer by layer.

[Note to instructors: This lab exercise is Part-2 of a two-part series. See explanation of what is covered in this Part 2 lab below.]

**Background**

As discussed in the last exercise (Part 1), the *network profile* of a system or host is how it appears to the outside networks of the world, and a system’s overall hardened network profile is a function of its *layered defense*. These defensive layers can be generalized into five or so defensive layer categories that we are splitting into two exercises for you to experience.

Part-2 of this lab will examine the final three layer categories:

* Part 1: External Layers
  + Outer Network & Access Layer
  + Network Services Layer
* Part 2: Internal Layers
  + Local Services Layer
  + Daemon Config Layer
  + OS & Filesystem Layer

Layers of a Network Host

In this second part of the host hardening lab, we’re going to be looking at any services that are not offering a network-based service (e.g. unused print, database or local SMTP (mail) services). We will either shut them down or configure them to run locally only (only visible on loopback or lo / 127.0.0.1 / ::1) so as not to be seen if there’s ever a failure or exploit at the outer kernel and access layer.

After locking down our local services, for the next layer in we’re going to look at securing the host’s running daemon configurations and make them a bit more secure by either restricting service access to specific users or groups or minimizing things like system-information leakage (e.g. daemon versions, patch levels, etc).

Finally, down at the OS and filesystem layer, one must always ensure to fully patch and use system-based OS auto-patching to keep and maintain a *software baseline\** that also keeps the system software current, as well as locking down directories, files, user permissions, etc.

**\*NOTE:** Relying on the security of *software baselines* assumes that:

1. the system is using *only* an approved OS standard packages. This means that packages and binaries only use official yum/RPM or apt-get/dpkg repository/approved sources (i.e. *not* allowing system files manually installed or being installed by “tar-ball” or other archive source), and
2. the system is not being treated as an ephemeral, disposable “image” (e.g. static ISO or virtual disk images without updates).

There are many other defenses that can be included in these layers such as host-based monitoring apps and tools, *jailing*/*chrooting*, *containers*, *software baseline reporting*, *FAM* (File Altercation Monitoring), *HIDS* (Host Intrusion Detection System), *log monitoring*, etc. However, this exercise will only look at a subset of these as outlined below.

**2. Resources required**

[Note to instructors: This lab exercise requires an account on the Virginia Cyber Range. To sign up for an account on The Range, please visit our Sign-Up page. Your students will also require an account on the Virginia Cyber Range; this will be explained in the setup of your course.]

An Internet-connected web browser and student login to the Virginia Cyber Range are required for this lab exercise. This lab exercise uses two Virginia Cyber Range virtual machines: a networking server (networking.example.com) you are tasked with locking down on the network and an auditing server (audit.example.com) from which you can scan your networking server. Your main Virginia Cyber Range login will provide a GUI desktop session to the networking.example.com virtual machine (VM). From there, you will open a local root terminal and a second root terminal with an ssh session to the audit server.

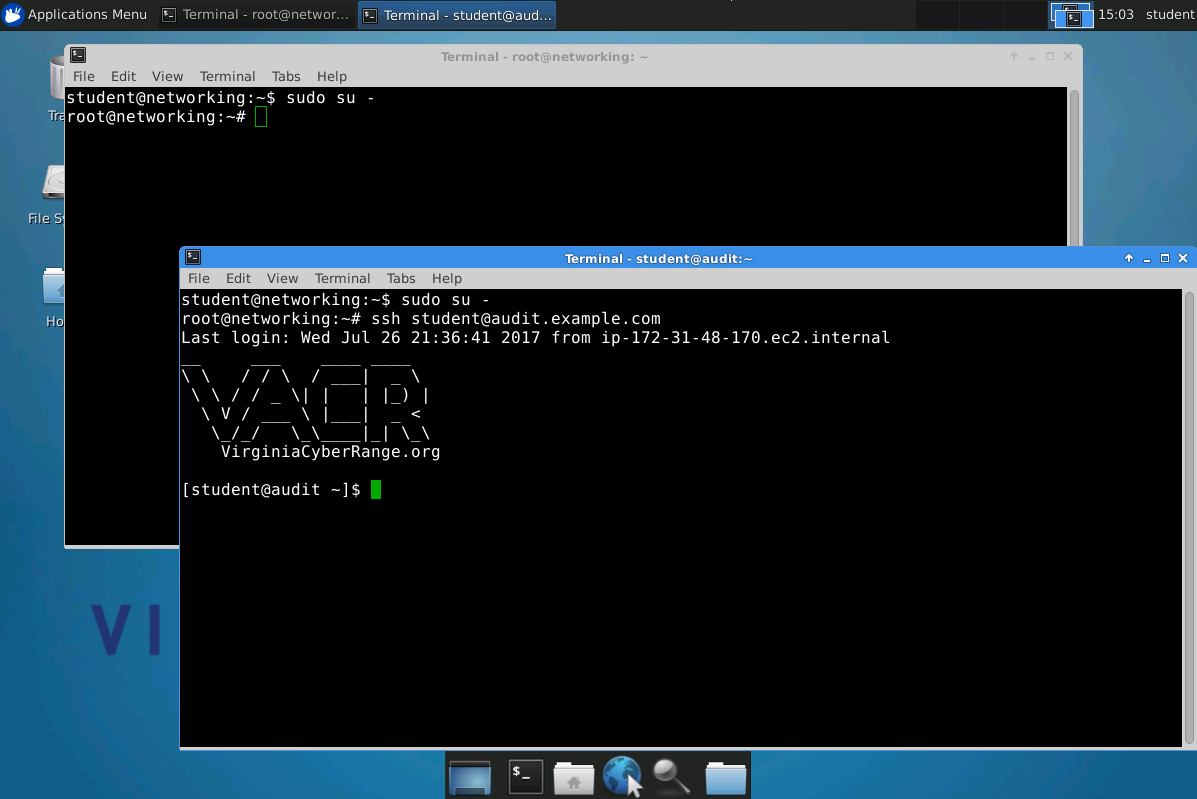
**3. Initial Setup**

Log into the Virginia Cyber Range (<https://portal.virginiacyberrange.net)>. Once logged in, select the Host Based Network Security Basics lab and the click "Join Exercise" button. Within your browser, you will be presented with a ssh terminal Linux login screen. Log in using these credentials:

Username: **student**

Password: **V4CR-n3t53cB451c5**

Next, open two terminals. In the first simply become root with **sudo su -** . This will be your network server (networking.example.com) terminal within the range where most of your work will take place. In the *second terminal*, also become root with **sudo su -** , and then ssh in to audit with **ssh student@audit.example.com** and become root there also.



This audit ssh session is where you will scan and audit your networking server from. Before continuing, ensure you have a setup that looks something like the figure above. You should have two root terminals up, one locally on the networking server and one on the audit server.

**4. Tasks**

**Challenge:**

The system you are being tasked with hardening (networking.example.com) is a vanilla, freshly provisioned Ubuntu Linux web/ssh server with a few vulnerable or network exposed services. In Part 1 of this lab, we already secured our ports, services that were exposed to the network, and locked them down on our *LAMP* server (Linux, Apache, MySQL, PHP). Next, we will work on addressing and securing the remaining *layered security* issues one by one. In this lab, we’ll be securing the remaining Layers 3 – 5 as listed here:

1. **Outer Layer:** Kernel level network and access controls
2. **Running Services Layer:** Network exposed services
3. **Local Services Layer:** Restrict local services to localhost
4. **Daemon Config Layer:** Harden exposed daemons
5. **OS & Filesystem Layer**

Let’s take a look at how to harden these remaining inner layers. You may find the Useful Commands and Config Files Cheat Sheet Handout helpful as you go through the tasks.

**Task 1: Local Services Layer: Restrict local services to localhost only**

On your networking.example.com server you want to configure risky, Internet exposed services or *daemons* to only run on the non-public localhost/127.0.0.1 addresses only instead of 0.0.0.0\* (any/all IP addresses) on the server.   
  
**\*NOTE:** On a running system, when the **netstat** command shows the 0.0.0.0 IP or 0.0.0.0/0, this is shorthand notation for *any* IPv4 address. A service bound to 0.0.0.0:25 (for example) means that the service (SMTP email in this case) will answer on any IPv4 IP address, public or private. Compare this to 127.0.0.1 or localhost, which is only the local loopback (non-public facing) IP and related interface lo.  
  
Locking down risky services (you don’t want to expose) to 127.0.0.1 will ensure that those services are not exposed to other computers on the network (even if iptables or your outer layers fail), thereby reducing the attack surface of your server when under attack:

1. **MySQL:** Configure mysqld/3306 to run on the 127.0.0.1:3306 (localhost-only)  
   Before the service is secured to only localhost, you can see that it is exposed to the world with the **netstat** command:

# **netstat -antp|grep -e ^Proto -e mysql**

Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name

tcp 0 0 0.0.0.0:3306 0.0.0.0:\* LISTEN 1219/mysqld

This shows the mysqld service is bound to all IP addresses (0.0.0.0/0), which can be accessed by outsiders. To lock this down at the service layer, edit the /etc/mysql/my.cnf service binding address from 0.0.0.0 to 127.0.0.1 and then restart the mysql service.

**Hint:** Find and change bind-address to 127.0.0.1 (in the my.cnf config file), and then restart the service with **service mysql restart** and rerun the **netstat** command above.

Q1: Once you have reconfigured and restarted MySQL, what does the above **netstat** command show as the new IP:port binding, and what does this effectively do for your server’s security profile?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **SMTP Email:** Configure smtp-mail/25 to localhost-only

Looking at the way this system was provisioned, the **netstat** command shows that the postfix email daemon is bound to public facing IP addresses:

# **netstat -antp|grep -e ^Proto -e master**  
Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name  
tcp 0 0 0.0.0.0:25 0.0.0.0:\* LISTEN 1343/master   
tcp6 0 0 :::25 :::\* LISTEN 1343/master

This shows that the postfix master, port 25 process (that accepts incoming email) is bound to both IPv4 (0.0.0.0:25) and IPv6 (:::25) IPs. Find the postfix daemon’s config for binding internet interfaces, change it to bind to only IPv4 127.0.0.1 and restart the postfix service (called “postfix”).

**Hint:** The config specific line is not in the master.cf file, and does not have anything to do with

the mynetworks line. Think physical.

Q2: As with the MySQL port binding step, what is the output of the above **netstat** command, and

what does this do for your server’s security profile if, for example, iptables were to be wiped out or

flushed?  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Q3: Speaking of iptables, what is the current output of **iptables -L**? Is your firewall running? If not, this is because your firewall is not persisting across reboots. Don’t do it, but do you know how you could make it come up at boot time *before* the server brings up its public IP? If so, jot it down here:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **Reboot & Check All *Local* Port Bindings:** By running a local **netstat,** reboot the networking server and verify your local (127.0.0.1) port bindings configured in tasks a & b still looks okay.  
     
   Q4: Briefly, what port bindings do you see from a local netstat command on your networking server?  
   Locally on networking.example.com:

**# netstat -antp**  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **Check All *External* Port Bindings:** By **nmap** scan from audit server

After the networking server reboot in c, (as root) type **iptables -F** to flush any firewall settings (removing that layer of defense) and then log back in to audit.example.com, do a **sudo su -** to root and use **nmap** to scan the networking server system to see that it none of your newly locked down daemon services are exposed (with the firewall in place). It should only be exposing ports of 22, 80, 443(if configured), and 3386 at most:

# **nmap -sS networking.example.com**  
PORT STATE SERVICE  
22/tcp open ssh  
80/tcp open http  
3389/tcp open ms-wbt-server

Q5: Briefly, what exposed ports do you see from the **nmap** scan   
of **networking.example.com**? Do you need to lock any services down? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Now re-run the **/etc/firewall.sh** script from the Part 1 lab exercise, and re-run the scan. It should look roughly the same (indicating you have the same settings at the outer (firewall) and inner (service) layers.

Q6: Did any of your **networking** server hardening steps revert after the reboot? If so, explain which ones and what you had to do to fix this: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Task 2: Daemon Config Layer: Harden exposed daemons**

1. **Lockdown SSHD:** Configure sshd to allow *only* the user “student” to log in  
     
   From the audit server session, log in to networking.example.com as the hacker user:  
   [root@audit ~]# **ssh hacker@networking.example.com**  
   \_\_ \_\_\_ \_\_\_\_ \_\_\_\_   
   \ \ / / \ / \_\_\_| \_ \   
    \ \ / / \_ \| | | |\_) |  
    \ V / \_\_\_ \ |\_\_\_| \_ <   
    \\_/\_/ \\_\\_\_\_\_|\_| \\_\  
    VirginiaCyberRange.org  
      
   System: networking.example.com   
     
   hacker@networking:~$  
     
   **Tip:** All logins between these systems is configured to be ssh-key based and does not  
   require passwords (making things much easier). Can you fine where in the sshd config  
   this is controlled?  
     
   Restricting *who* can and cannot log in to a server is critical. Many distros no longer allow root logins by default, but at the sshd service level you can easily control users or groups of users who can get in.  
     
   The sshd service is controlled via the /etc/ssh/sshd\_config file. See the sshd\_config man page to see how to restrict user logins. Don’t forget to restart the sshd service once done.

**Hint:** Typically, you don’t want to allow root logins to production, especially on external facing systems. The two lines that can be added/changed to meet this requirement are:

PermitRootLogin no  
...  
AllowUsers student

and then restart the service with **service sshd restart** . On most sshd versions, this config is redundant in that AllowUsers will block all users (including root) if not listed in the AllowUsers directive.

Q1: After making the user login restriction, can student still log in? What about the hacker user? root?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Tip:** User logins can be controlled at multiple levels. The sshd\_config, the /etc/passwd file (see shell named “nologin”), as well as at the pam level (for experts only) are just three such layers. If you’re going to control user logins, it’s best to only do so at one “layer” so troubleshooting doesn’t become problematic. Ideally, most IT experts will not even use on-box access controls or user listings, but prefer more centrally controlled authentication systems such as ldap, kerberos or other dedicated system (such as with dual factor services) which utilize pam plugins.

1. **SSHD / HIPS:** Install/enable a *Host Intrusion Prevention System (HIPS)* Fail2Ban. This watches your ssh log files and block attackers performing ssh brute attacks on your system for a specific amount of time. To see what’s currently configured to “+” start at your default runlevel (2 in this system), run the service --status-all |less (for non-systemd based systems).   
     
   Q2: Does the fail2ban service appear?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**Tip:** When installing packages, don’t forget that for apt and rpm package installs, the package management system needs a current list of all available software. Before running apt-get or aptitude, if you have not done so recently, be sure to update the package management repository listings with the update command:

# **aptitude update** # update all repository meta data  
  
Now you are ready to download and install packages on the system.

Once you locate the fail2ban package in the apt repositories, go ahead and install it:

# **aptitude search fail2ban** # find fail2ban

p fail2ban - ban hosts that cause multiple authentication errors

# **aptitude install -y fail2ban** # install the package

The following NEW packages will be installed:  
 fail2ban python-pyinotify{a} whois{a}   
…  
  
Now once again, run the service --status-all |less command and record what it shows you about fail2ban:  
  
Q3: What does the service command show you about the configured state of fail2ban?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

One way to check the service or even force a restart is with the service command:

# **service fail2ban status**  
 \* Status of authentication failure monitor \* fail2ban is running  
# **service fail2ban restart**  
 \* Restarting authentication failure monitor fail2ban

Another lower level ways to see if it’s running is to grep for the running processes:

# **ps auxw|grep [f]ail2**

Q4: What’s the process ID of your running fail2ban service?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

To see all the clients being iptables blocked from ssh logins to your system, keep an eye on the log file: /var/log/fail2ban.log

# **tail -f /var/log/fail2ban.log**

If your sshd config is still set to only allow the user student to login, then from your root audit session, try to log back into hacker@networking.example.com a bunch of times.   
  
Q5: Of course, you won’t be able to log in (because from previously, we’re only allowing in the user student), but what do you see from the client side after attempting the same login attempt between 5-10 times?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Q6: What do you see in /var/log/fail2ban.log?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Q7: How can this make your system more secure, and what type of attack can this inner layer of defense alone protect your systems from? (see the man page for fail2ban if needed) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Q8: What appears to be your system’s default “bantime” (duration)?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **Apache2 Default Index Page:** Secure Apache From Leaking System Info  
     
   Apache2 is most Linux system’s default web server daemon. Most distros, however, allow it to disclose, or “leak” critical system info (through web pages) about your system that can be used by would be attackers.   
     
   Before disabling the server’s base index.html page, examine what it looks like by pointing your system’s web browser or the elinks text based browser to your server’s FQDN (fully qualified domain name) or the localhost hostname:

# **elinks -dump http://localhost** NOTE: If you don’t have elinks, then try installing it.  
Q9: Looking through this default web page, what all does it disclose or “leak” about your system? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Now delete or rename the base index.html file on the networking system:

# **mv /var/vvv/html/index.html /var/www/html/index.html\_ORIG**  
  
and reload the base web page.  
  
Q10: What do you see after this change? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Q11: Why could the resulting behavior alone be even worse than the default index.html file? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **Apache2:** Disable root web Indexes  
     
   Now that the base page is gone, we will want to ensure that the server won’t simply allow clients to surf the filesystem using file indexes (filesystem browsing). This feature can be turned on and off for each web site, but the main server’s “indexes” feature needs to be shut off in the main /etc/apache2/apach2.conf file like this:

<Directory /var/www/>  
**Options -Indexes**  
Options FollowSymLinks  
...  
  
and of course restarting apache2:

# **service apache2 restart**

Q12: Now what do you get when hitting the main networking.example.com web page before and after this change, and what info are we still leaking?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **Apache2 Info Leaking:** Lock down the server-info setting  
     
   Before locking down this setting, from the networking server itself, record what info the default configuration is leaking using the curl, command line web client utility like this:

# curl http://networking.example.com

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Now that you have captured the before, make the /etc/apache2/conf-enabled/security.conf config change below

#ServerTokens Full  
ServerTokens Minimal  
...  
#ServerSignature On  
ServerSignature Off  
  
then restart apache2, and record the change in the curl output:  
  
Q13: What info did you prevent from being leaked?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **Apache2 / WAF:**   
   Here are other more active actions you can takein the installation of Web Application Firewalls (WAFs) packages such as libapache2-mod-security2 or libapache2-mod-evasive.

Q14: Using the package manager’s “show” option, report the short one-line description of what libapache2-mod-security2 and libapache2-mod-evasive each does for your apache2 server’s security, if you were to employ them:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Task 3: OS & Filesystem Layer**

* 1. **OS Package Updates:** Update to latest distro package versions  
     Always make sure to update the package manager repository info, and then apply all the latest software package updates. On Debian/Ubuntu-based systems, this is performed with the apt-get or aptitude package manager (other RPM based systems are similar, but using the yum package manager). Either perform this using **sudo** or as root:

# **aptitude update** *# update all package repository data*

…

# **aptitude -y -q full-upgrade** *# download & install updates*  
[100%] Building dependency tree  
…

Now all the apt repository’s meta data are current and all the latest package updates have been applied. Next, to *stay secure* (package wise) you must ensure that future security updates are automatically applied.

* 1. **Auto Package Updates:** Enable auto updates (if applicable)  
     On Debian/Ubuntu-based systems, you enable automatic package updates by first installing the unattended-upgrades package, and then configuring your system to perform daily security software updates. Can you now use aptitude to install unattended-upgrades?

After installing the unattended-upgrades package, then copy the automatic upgrades file into place (for Ubuntu Linux systems):

# **cp -a /usr/share/unattended-upgrades/20auto-upgrades /etc/apt/apt.conf.d/**

* 1. **Secure /home and /home/$USER folders:**  
     To prevent users from seeing the contents of other people’s home directories, or enumerating your system’s home directories, one can also change the mode of each user’s home directory (/home/$USER/) or the whole /home/ folder.  
       
     As the student user, look to see what other users have /home folders:

$ **pwd** *# print (your current) working directory*  
/home/student  
$ **ls -la /home/**  
total 20  
drwxr-xr-x 5 root root 4096 Jul 26 16:36 .  
drwxr-xr-x 24 root root 4096 Jul 27 17:50 ..  
drwxr-xr-x 4 hacker hacker 4096 Jul 27 19:08 hacker  
drwxr-xr-x 22 student student 4096 Jul 27 19:36 student  
drwxr-xr-x 4 ubuntu ubuntu 4096 Jan 9 2017 ubuntu  
  
  
$ **ls -la /home/ubuntu/** *# Take a peek inside this person’s dir*  
total 44  
drwxr-xr-x 4 ubuntu ubuntu 4096 May 9 21:30 .  
drwxr-xr-x 4 root root 4096 Jan 9 15:48 ..  
-rw------- 1 ubuntu ubuntu 939 May 9 21:31 .bash\_history  
-rw-r--r-- 1 ubuntu ubuntu 220 Apr 9 2014 .bash\_logout  
-rw-r--r-- 1 ubuntu ubuntu 3637 Apr 9 2014 .bashrc  
drwx------ 2 ubuntu ubuntu 4096 Jan 9 15:14 .cache  
-rw-rw-r-- 1 ubuntu ubuntu 11 May 9 21:31 PRIVATE-STUFF  
-rw-r--r-- 1 ubuntu ubuntu 675 Apr 9 2014 .profile  
drwx------ 2 ubuntu ubuntu 4096 Jan 9 15:11 .ssh  
-rw------- 1 root root 636 Jan 9 18:35 .viminfo  
-rw-rw-r-- 1 ubuntu ubuntu 14 Jan 9 15:21 .xsession

$ **cat /home/ubuntu/PRIVATE-STUFF** *# Peek inside a private file*  
Top Secret

Still as the student user, but using **sudo**, lock down the /home and /home/$USER folders:

$ **sudo chmod 711 /home/\***

$ **sudo chmod 711 /home/**

and then try to access private content:

Q1: Now what do you see now when you do **ls -la** of **/home**?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**5. References**

* Useful Commands and Config Files Cheat Sheet

[This portion of the lab exercise template is provided for instructors that will be using this lab in a class they are teaching.]

**Answer Key (Coming soon! Please check with the author for solutions.)**

**Task 1: Local Services Layer: Restrict local services to localhost only**

Q1

Q2

Q3

Q4

Q5

Q6

**Task 2: Daemon Config Layer: Harden exposed daemons**

Q1

Q2

Q3

Q4

Q5

Q6

Q7

Q8

Q9

Q10

Q11

Q12

Q13

Q14

**Task 3: OS & Filesystem Layers**

Q1

**KSAs Addressed**

From (<http://csrc.nist.gov/publications/drafts/800-181/sp800_181_draft.pdf>)

**Knowledge:**

* K0033: Knowledge of host/network access control mechanisms (e.g., access control list).
* K0224: Knowledge of system administration concepts for Unix/Linux and/or Windows operating systems.
* K0537: Knowledge of system administration concepts for the Unix/Linux and Windows operating systems (e.g., process management, directory structure, installed applications, Access Controls).
* K0608: Knowledge of Unix/Linux and Windows operating systems structures and internals (e.g., process management, directory structure, installed applications).

**Skills:**

* S0007: Skill in applying host/network access controls (e.g., access control list).

**Knowledge Units (KUs) Addressed:**

From (<https://www.iad.gov/NIETP/documents/Requirements/CAE-CD_Knowledge_Units.pdf)>

* Cyber Defense