[Any blue text should be replaced by instructor using material and font color changed to black.]

Course Title
Term: (Fall, Spring, Summer, Winter) 20XX

Laboratory Exercise X – Host Based Network Security Basics - Part-1

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

1. Overview

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-1 of a two-part series. See explanation of what is covered in this Part 1 lab below.]

Background

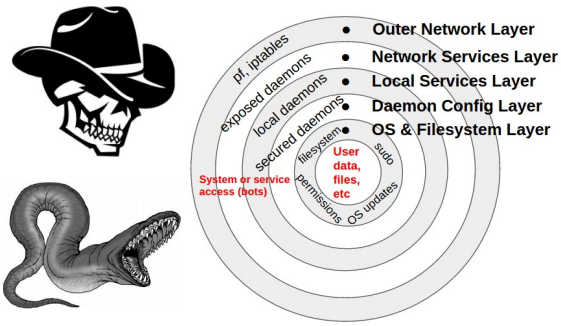
The *network profile* of a system or host is how it appears to the outside networks of the world, or put another way, what aspects of a system can be seen, probed and exploited by those on the network. The less services you run, the less you expose to the outside world. Further in on the server, the more you harden a host's various *layers,* the better the system's overall hardened network profile or *layered defense* becomes.

While no system should ever be considered 100% secure, one can imagine a system's network security profile as consisting of various *layers* from the outside in. Each layer in builds upon the outer layers, ultimately protecting system access and user data at the center. The more layers and controls that are in place, the more secure the host's overall network security profile.

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 1 of this lab will only examine our first two 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 Layers



Layers of a Network Host

2. Resources required

[Any blue text should be replaced by instructor using material and font color changed to black.]

Course Title
Term: (Fall, Spring, Summer, Winter) 20XX

[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

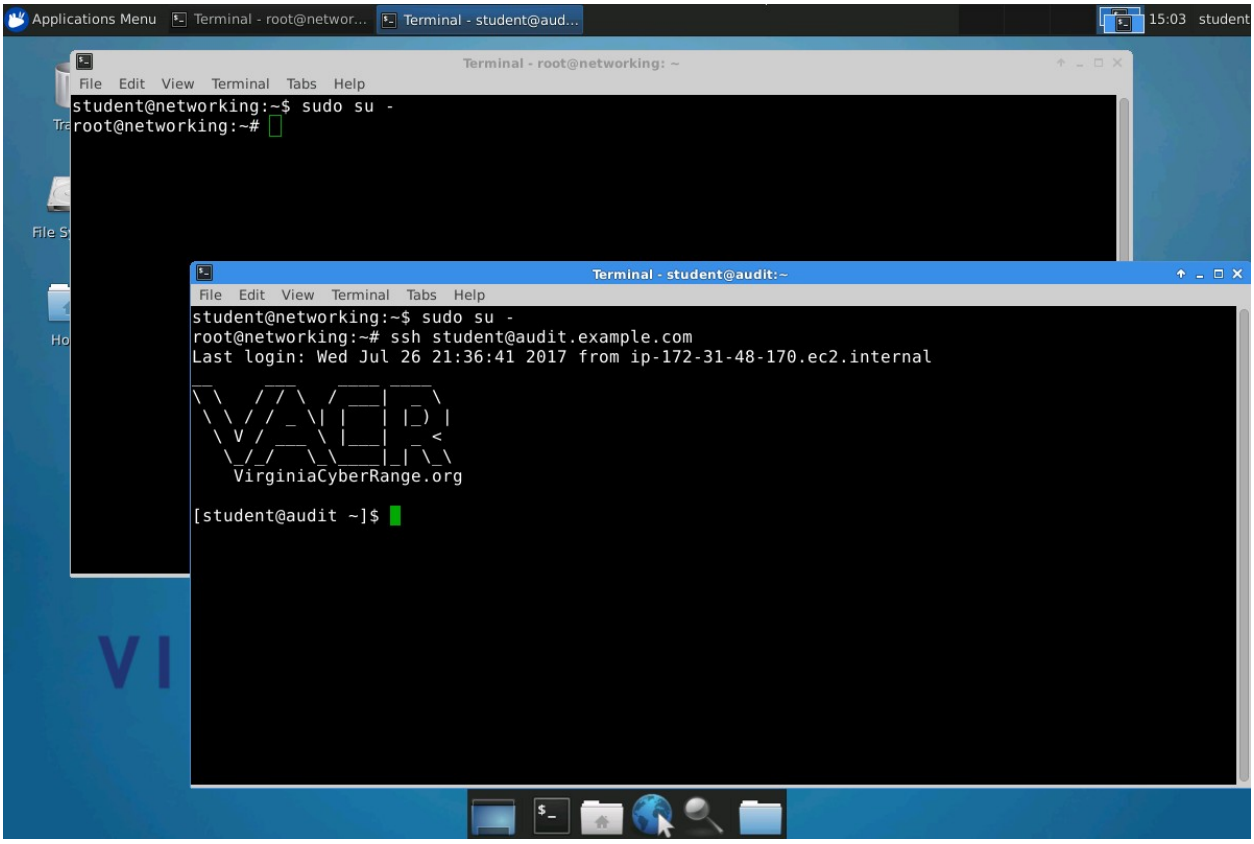
Log into the Virginia Cyber Range (https://portal.virginiacyberrange.net). Once logged in, select the Host Based Network Security Basics lab and the click "Join Exercise" button. Within your browser, you will be presented with a ssh terminal Linux login screen. Log in using these credentials:

> 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.

[Any blue text should be replaced by instructor using material and font color changed to black.]

Course Title
Term: (Fall, Spring, Summer, Winter) 20XX



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. You need to examine what ports, IPs and services are exposed to the network, and work on addressing and securing the outstanding network security issues layer by layer.

Hardening a *LAMP* server (Linux, Apache, MySQL, PHP) is typically done by securing the following layers of the system:

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**

[Any blue text should be replaced by instructor using material and font color changed to black.]

Course Title
Term: (Fall, Spring, Summer, Winter) 20XX

Let's take an in depth, hands on look at how to harden each of these layers. You may find the Useful Commands and Config Files Cheat Sheet Handout helpful as you go through the tasks.

Task 1: Outer Layer: Kernel level network and access controls
On the `networking` server, configure the following network and system *Mandatory Access Controls* to limit network and system access:

a. **Network:** Looking at your iptables firewall (in /etc/firewall.sh):

These typical firewall *access control lists* (ACLs) rules, as well a few others, are already set up in the top of your `firewall.sh` script:

This ACL entry allows all `lo` interface (localhost) traffic in:
`iptables -A INPUT -i lo -j ACCEPT`

This entry allows inbound TCP ssh port 22 sessions in on `eth0`:
`-A INPUT -i eth0 -p tcp --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT`

And this entry allows in RDP (remote desktop) port 3389 sessions:
`-A INPUT -i eth0 -p tcp --dport 3389 -m state --state NEW,ESTABLISHED -j ACCEPT`

WARNING: Don't edit these two rules in the top half of the **`/etc/firewall`** script! They maintain your connectivity to the cyber range system.

i. From the `audit` server's root login terminal, run this **`nmap`** stealth scan (**`-sS`**) against the `networking` server:

```
# nmap -sS networking.example.com
Not shown: 994 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
25/tcp   open  smtp
80/tcp   open  http
631/tcp  open  ipp
3306/tcp open  mysql
3389/tcp open  ms-wbt-server
```

This is what the stock networking server looks like from the outside (without any firewall in place).

ii. Back on the `networking` server's root prompt, run the *stock* firewall script as is:

iii.

```
# /etc/firewall.sh
APPLYING FIREWALL RULES:  Use "testing" option to have it auto-flush
in 2 minutes.
Done!
```

iv. Next switch back to the `audit` server's terminal and run the same **`nmap`** scan of the `networking` server and compare the new scan's output (like below) to what

you saw before.

Q1: What ports are now open from the outside with this stock firewall script in place?

_____.

Q2: What services do those ports represent?

_____.

Q3: Given the requirements of this as a web server with access ports 22 and 3389 open, what are all of the ports should one should be seeing as "open"?

_____.

NOTE: Ports 22 and 3389, are the SSH and RDP ports, respectively, that the Cyber Range uses to provide you the remote login and "Remote Desktop" service.

v. Switch back on the `networking` server, open the firewall.sh script and go to the bottom where there is the space for additional firewall ACL "ALLOW" entries.

One example rule allowing in port 22 /ssh sessions (just as a template) can be seen, followed by a line REJECTing all other incoming traffic (that has not been previously allowed).

NOTE: With firewall or ACL rules, **order matters**. First you let in specific ports & protocols, then you block everything else. Get your ACLs out of order and nothing will work correctly.

Use the `--dport 22` line to poke similar "holes" in the firewall to allow all web traffic through the firewall:

```
iptables -A INPUT -p tcp -m tcp --dport 22 -j ACCEPT
# <-- enter your rules here
iptables -t filter -A INPUT -j REJECT
```

Q4: Which ports did you add to your firewall script?

_____.

WARNING: Do **not** modify the ACCEPT lines for ports 33xx (for RDP) or port 22 (ssh) in the upper section of the firewall script. Otherwise you could lock yourself out of the system.

TIP: When testing the firewall script the first few times, invoke it with the

`testing` option which will disregard (flush) you firewall changes 2 minutes after running it:

```
# /etc/firewall.sh testing
TESTING: Will auto-flush in 2min...
Done!
```

If you do lock yourself out of the system without the **`testing`** option, then simply stop and restart your exercise session through the Virginia Cyber Range web interface.

vi. After editing the firewall script to allow web traffic through, then re-run **`/etc/firewall.sh`** on the `networking` server and then rescan (using **`nmap`**) `networking.example.com` from the `audit.example.com` server again and adjust the firewall script entries until you get the following results from the `audit` scan:

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

TIP: If you mess up your firewall script, there's a backup of the original file called `/etc/firewall.sh_ORIG` that you can copy back over the main firewall script.

NOTE: In a production system, the firewall script must be loaded every time the system boots *before* the network brings the system fully on line. One should never load a firewall *after* the networking subsystem is up, else the system could briefly be on line (e.g. by using rc.local, etc) with no firewall blocks in place. Doing so leaves makes the system vulnerable to "reboot window" attacks.

b. **System:** Employ & test AppArmor *Mandatory Access Control (MAC)* system MAC suites such as AppArmor or seLinux are typically kernel level, system wide safety nets that keep rogue or unapproved processes or services from accessing parts of the system they should not. MAC systems are good to run on any system, but are most commonly utilized on externally facing, networked systems that could see hacking attempts, buffer overflows, illegal applications or user privilege escalation attempts. Unexpected or unauthorized applications get scope-restricted to a subset of system resources. Even root can be *sandboxed* to only do certain operations on a tightly restricted MAC enabled system.

Configuring AppArmor to apply sshd profile restrictions:
AppArmor profiles are kernel level MAC configurations files called profiles that restrict system, applications and user access. Adding new application profiles requires the reloading or restarting of the AppArmor service. But before this can be done for ssh (in this exercise), the additional profiles package needs to be installed and the ssh profile added to the active AppArmor service.

i. Before going forward, examine the `networking.example.com` server:

Q5: Record the "sbin" daemon profile config files you currently see in the /etc/apparmor.d/ directory?

_____

TIP: Daemon programs are almost always in the /sbin or /usr/sbin directories. Thus, the following command will quickly show just the apparmor configured daemon profiles: `ls -1 /etc/apparmor.d/ | grep sbin` .. and tacking on `| wc -l` to the end will get you a "word count lines" of how many there are.

ii. On the `networking` server, now install the packages apparmor (user space tools) and `apparmor-profiles` for additional AppArmor profile configs:

```
# aptitude install apparmor apparmor-profiles
```

Tip: In this Debian/Ubuntu based Linux system the **`apt-get install <package-name>`** or **`aptitude install <package-name>`** is used to install packages. On newer Debian based systems, **`apt`** is used and on Red Hat/rpm systems the **`yum`** meta package manager is used to download and install packages from approved repositories.

Q6: How many daemon profile config files do you currently see in the /etc/apparmor.d/ directory?

```
# ls -1 /etc/apparmor.d/ | grep sbin | wc -l
```

_____

Q7: What additional services are now protected by apparmor?

_____

_____

iii. Reboot-Check: After a reboot that you can verify apparmor by either running the **`apparmor_status`** command (showing all running profiles) or the following command and getting a "Y" response:

```
# cat /sys/module/apparmor/parameters/enabled
Y
```

Hint-2: To ensure the AppArmor MAC suite starts up and is configured to start, use the **`service`** command to start and stop it, and the **`update-rc.d`** command to configure it to start at boot time. Newer systems use the systemd **`systemctl`** suite to control this.

iv. To enable or write other apparmor profiles, you would place them into the /etc/apparmor.d/ profile config directory and restart the service:

```
# cp -a /usr/share/doc/apparmor-profiles/extras/usr.sbin.sshd \
    /etc/apparmor.d/
# service apparmor restart
 * Reloading AppArmor profiles
```

Task 2: Running Services Layer: Network exposed services

Configuring system to shut down services/ports not being used:

a. cupsd: Disable/Remove cups/631
The cups service can be seen running with the service command:

```
# service cups status
   cups start/running, process 1768
```

You can also see it is bound to a public IP address with the **netstat** command:

```
# netstat -antp|grep -e ^Proto -e cupsd
Proto Recv-Q Send-Q Local Address    Foreign Address  State   PID/Program name
tcp        0      0 0.0.0.0:631      0.0.0.0:*        LISTEN  1768/cupsd
```

Since this is a headless web server, we do not need or want the unused cupsd printer daemon running at all, much less being exposed to the world. You could stop + disable a service using the `update-rc.d cups stop + update-rc.d -f cups remove` (on older upstart/Ubuntu systems) or `systemctl stop cups.service + systemctl disable cups.service` (on newer systems). However, in this case on a non-GUI non-desktop based system, it should probably just be removed completely from the system.

Stop the service (above) and then remove it from the system using apt-get/aptitude so we don't have to worry about it from now on.

> **NOTE:** Record the command you use to remove the cups package, and record how many related sub packages are also removed.
>
> **Hint:** To remove packages, see the man page for `apt-get` / `aptitude` (on older Ubuntu systems) or just `apt` on newer systems.

After removing the cups package, reboot the system to ensure your configuration is persistent. After rebooting, the same netstat command should look like this:

```
# netstat -antp|grep -e ^Proto -e cupsd
Proto Recv-Q Send-Q Local Address    Foreign Address  State   PID/Program name
```

Once you indeed have the **netstat** results seen above, answer the following questions:

[Any blue text should be replaced by instructor using material and font color changed to black.]

Course Title
Term: (Fall, Spring, Summer, Winter) 20XX

Q1: What command did you run to remove the cups package?

_____

_____

_____

Q2: How many packages were removed?

_____

_____

_____

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: Outer Layers: Kernel level network and access controls
Q1
Q2
Q3
Q4
Q5
Q6
Q7

Task 2: Running Services Layer: Network exposed services
Q1
Q2

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.

[Any blue text should be replaced by instructor using material and font color changed to black.]

Course Title
Term: (Fall, Spring, Summer, Winter) 20XX

- 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