**Laboratory Exercise X – Compromised Server Securing and Lockdown**

Due Date: Date

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

**1. Overview**

This laboratory exercise provides hands-on experience with evaluating (scanning), identifying, and correcting vulnerabilities on an infected server.

**2. Resources required**

An Internet connected web browser and student login to the Cyber Range is required to complete this lab. Once in the Cyber Range, it requires access to two CentOS Linux VMs: one pre-configured infected server named **server.example.com** and audit server from which you can scan and test the infected server named **audit.example.com**.

[Note to instructors: This lab exercise requires an account on the 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 Cyber Range; this will be explained in the setup of your course.]

**3. Initial Setup**

Log into the Cyber Range. Once logged in, select the Compromised Server Securing and Lockdown 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-d3W0rm1nG**

You should now be logged in to a terminal session on **server.example.com** within your browser window. This machine is **server.example.com** where most of your work (on the server) will actually take place.

Since there is no graphical interface, and you will need a second login terminal to the audit system **audit.example.com**, you will need start a *screen session* (on the main server) so that you can switch between the main server and a second ssh session to the **audit.example.com** system. Follow the steps below to start your *screen* session. (NOTE: If you have never used *screen* before, here’s a good *screen* tutorial: <https://www.youtube.com/watch?v=hB6Y72DK8mc>)

1. **Start a screen session:**
To start a screen session on **server.example.com**, just type **screen**. This will open *screen* and place you at a new shell session:

 [student@server ~]$
 [student@server ~]$
 [student@server ~]$ **screen**

(the screen will clear, and drop you into a new shell in your screens session)
 [student@server ~]$

## TWW SCREEN NOTES:

**$ screen
 CTRL-a (release) SHIFT-S** # To do to split screen to two regions

**CTRL-a TAB**  # To flip to lower region

**CTRL-a c** # Top open new screen window in region

$ **ssh** **student@audit.example.com** # To log into audit servers

$ **sudo su -** # To become root on audit server

1. **Open a new screen window:**
Now within *screen*, hit keys **<CTRL>-a** (release) and then **C**, which will open a new screen window. **<CTRL>-a** (release) **a** will toggle between your two screen windows, and **<CTRL>-a** (release) **n** will go to the Next screen window (if you have more than two open). Use **<CTRL>-a** (release) **?** to get help using *screen*.
2. **Open a second ssh session to audit.example.com:**On one of your screen windows, ssh to the machine **audit.example.com**, and use
 **sudo su -** to become root:

 $ **ssh student@audit.example.com**

[student@audit ~]$ **sudo su -** # prompts you for your own password

 [root@audit ~]# # now you ARE the root user

and switch back and forth between your server and audit systems using **<CTRL>-a a** to verify you are logged in to both of your systems at the same time.

 **NOTE:** You will use the audit session later when you will need to **nmap** scan your server from the audit system. **nmap** is the preferred sysadmin and network security tool of choice when quickly scanning systems to check to see what ports and services are exposed.

**TIP:** **sudo** is the command that allows unprivileged users to run system or root level commands. Using **sudo** to invoke **su -** allows a normal user to become root. Don’t forget the tailing “**-**“ with **su**. That is what tells the system to load that user’s environment and path settings. The password **sudo** prompts you for is your student password.

**4. Tasks**

**Challenge**

On the **server.example.com** machine, using the general strategy (tasks below) and Cheat Sheet Commands handout, evaluate the integrity of your system by:

* scanning your systems network profile locally and remotely
* identifying any non-standard network profile/port bindings
* identifying what processes are responsible, killing those processes, verifying they cannot re-spawn after a reboot, and
* verifying system binaries (in /bin, /sbin, /usr/bin/, /usr/sbin/ and /usr/local/ ) are all “clean” at a basic level (files in these areas should all pass rpm verification checks.. especially MD5sum fingerprint verifications)
* remove and clean up any compromised system files or configurations
* **End goal**: Verify that system in a stable, semi-secured state after reboots (can never 100% sure after a system/root level compromise happens)

Read the questions in the **Questions** section below and keep them in mind as you complete the following tasks to secure and “de-worm” the compromised server (**server.example.com**):

**Task 1: Examine Port Bindings & Port Scans**

* + On system server, look at the IP/port bindings and see what’s running (**netstat**)
	+ On system audit, scan server (using **nmap** on the audit server, ports 1-10,000)

	**NOTE:** See the Cheat-Sheet Commands Handout for help on the various steps above and below. Read man pages, e.g. **man netstat**, if something doesn’t make sense.

**Task 2 Examine Running Services:** (chkconfig, service, ps, etc.)

* **Compare Running vs Configured Services:** Compare the bound vs running services
* **Stop & Start Services:** On server, try stopping each service (not sshd) and reuse **netstat**

**Task 3: Persistent Network Lockdown**

1. Lock down unused ports (edit /etc/sysconfig/iptables file)

**WARNING:** BE CAREFUL! Don’t block ports 22 or 3389 or you can lock yourself out!

**NOTE:** If you do lock yourself out, you may need to exit your web session, and “reset” your lab exercise, effectively losing the work you’ve done.

**Task 4: Stop Rogue Processes**

* **Kill Any Rogue Services:** On server.example.com, try a **killall httpd** and re-run **netstat -antp** a few times and see what happens. Try rebooting.. checking again.
* Kill any rogue processes (use ps & kill PIDs w/unexpected port bindings)

**Task 5: Verify System Packages**

* Check for any corrupted system binaries, system scripts or config files (rpm -Va), looking for any md5sum changes

**Task 6: Repair System Packages**

* Repair any corrupted/compromised system binaries (can use rpm), and remove any suspicious files (rm and/or yum install ...)

**Task 7: Reboot Check (YOUR GOAL)**

* After completing the last few steps, reboot and verify that there are no more rogue binaries and port bindings persisting, that the system passes an rpm -Va | grep bin/ check (that returns nothing), and that this semi-safe state persists across reboots (w/first four strategy steps above).

**TIP:** If the system seems to “reinfect” itself after every reboot (i.e. you can see strange port bindings with the **netstant -antp** command as root), then Google the “centos 6 boot process” (after the kernel is loaded), and find out what boot scripts are invoked and control the system at boot time.

**Questions:**

Answers the following questions about what you discovered as you stepped through the tasks above.

1. ***What ports did you find listening on the server’s public IPs (0.0.0.0) and what processes were associated with each? How did you get this information (list specific commands & switches)?***

1. ***What does it mean to be a standard port binding? Which of the ports/process combinations from Q1, above, are non-standard bindings? How do you know they are non-standard?***

1. ***What actions did you take to block non-standard ports? How did you ensure they were still blocked on reboots?***

1. ***How did you stop any non-standard port bindings or rouge processes? How did you ensure these processes didn't return on reboot?***

1. ***What does it mean for a system binary to be “clean”? Which system binaries were not clean and how did you discover this?***

1. ***What actions did you take to repair the bad binaries?***

1. ***What other changes did you make to server to make it more safe and stable across reboots?***

**5. References**

* Cheat Sheet Commands Handout

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

**KSAs, from NIST SP 800-181:** <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-181.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_2019_Knowledge_Units.pdf>

(you may need to accept an invalid iag.gov SSL certificate to reach this PDF)

* Cybersecurity Foundations (CSF)