NUST CYBER SECURITY TEAM 2019

Linux Playbook

Linux Security Hardening Tips:

1. Keep Linux Kernel and Software Up to Date

Applying security patches is an important part of maintaining Linux server. Linux provides all necessary tools to keep your system updated, and also allows for easy upgrades between versions. All security update should be reviewed and applied as soon as possible. Again, use the RPM package manager such as yum and/or apt-get and/or dpkg to apply all security updates.

 # yum update

OR

 # apt-get update && apt-get upgrade

2. Minimize Software to Minimize Vulnerability in Linux

Do you really need all sort of web services installed? Avoid installing unnecessary software to avoid vulnerabilities in software. Use the RPM package manager such as yum or apt-get and/or dpkg to review all installed set of software packages on a system. Delete all unwanted packages.

 # yum list installed

 # yum list packageName

 # yum remove packageName

OR

 # dpkg --list

 # dpkg --info packageName

 # apt-get remove packageName

3. Linux User Accounts and Strong Password Policy

Change the passwords of root and any other administrator accounts. Use the useradd / usermod commands to create and maintain user accounts. Make sure you have a good and strong password policy. For example, a good password includes at least 8 characters long and mixture of alphabets, number, special character, upper & lower alphabets etc. Most important pick a password you can remember. Don't forget to change passwords for MySQL and other programs!

4. Locking User Accounts After Login Failures

Under Linux you can use the faillog command to display faillog records or to set login failure limits. faillog formats the contents of the failure log from /var/log/faillog database / log file. It also can be used for maintains failure counters and limits.To see failed login attempts, enter:

#faillog

To unlock an account after login failures, run:

#faillog -r -u userName

Note you can use passwd command to lock and unlock accounts:

Lock account:

#passwd -l userName

Unlock account:

#passwd -u userName

5. How to Verify No Accounts Have Empty Passwords?

Type the following command:

 # awk -F: '($2 == "") {print}' /etc/shadow

Lock all empty password accounts:

 # passwd -l accountName

6. Make Sure No Non-Root Accounts Have UID Set To 0

Only root account have UID 0 with full permissions to access the system. Type the following command to display all accounts with UID set to 0:

 # awk -F: '($3 == "0") {print}' /etc/passwd

You should only see one line as follows:

root:x:0:0:root:/root:/bin/bash

If you see other lines, delete them or make sure other accounts are authorized by you to use UID 0.

7. Disable Unwanted Linux Services

Disable all unnecessary services and daemons (services that runs in the background). You need to remove all unwanted services from the system start-up. Type the following command to list all services which are started at boot time in run level # 3:

 # chkconfig --list | grep '3:on'

To disable service, enter:

 # service serviceName stop

 # chkconfig serviceName off

8. Find Listening Network Ports

Use the following command to list all open ports and associated programs:

#netstat -tulpn

OR use the ss command as follows:

#ss -tulpn

OR

#nmap -sT -O localhost

#nmap -sT -O server.example.com

9. Configure Iptables and TCP Wrappers based Firewall on Linux

Iptables is a user space application program that allows you to configure the firewall (Netfilter) provided by the Linux kernel. Use firewall to filter out traffic and allow only necessary traffic. Also use the TCPWrappers a host-based networking ACL system to filter network access to Internet. You can prevent many denial of service attacks with the help of Iptables

10. Logging and Auditing

You need to configure logging and auditing to collect all hacking and cracking attempts. By default syslog stores data in /var/log/ directory. This is also useful to find out software misconfiguration which may open your system to various attacks.

11. Secure OpenSSH Server

The SSH protocol is recommended for remote login and remote file transfer. However, ssh is open to many attacks.

12. Install And Use Intrusion Detection System

A network intrusion detection system (NIDS) is an intrusion detection system that tries to detect malicious activity such as denial of service attacks, port scans or even attempts to crack into computers by monitoring network traffic.

13. Use fail2ban/denyhost as IDS (Install an Intrusion Detection System)

Fail2ban or denyhost scans the log files for too many failed login attempts and blocks the IP address which is showing malicious signs. See how to install and use denyhost for Linux. One can install fail2ban easily:

#sudo apt-get install fail2ban

OR

#sudo yum install fail2ban

Edit the config file as per your needs:

#sudo vi /etc/fail2ban/jail.conf

Restart the service:

#sudo systemctl restart fail2ban.service

14. Secure Apache/PHP/Nginx server

Edit httpd.conf file and add the following:

ServerTokens Prod

ServerSignature Off

TraceEnable Off

Options all -Indexes

15. Disable root Login

It's a good practice not to login as root user. You should use sudo to execute root level commands as and when required. sudo does greatly enhances the security of the system without sharing root password with other users and admins. sudo provides simple auditing and tracking features too.

Useful tools:

1. Fail2ban

The fail2ban application monitors server log files for intrusion attempts and other suspicious activity. After a predefined number of failures from a host, fail2ban blocks its IP address automatically for a specific duration.

With fail2ban, you can help secure your server against unauthorized access attempts. It is particularly effective in reducing the risk from scripted attacks and botnets.

- To install the fail2ban package for your Linux distribution:
  #apt-get install fail2ban
  #yum install fail2ban

- Configuring fail2ban:

#cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local

The *jail.conf* file contains a basic configuration that you can use as a starting point, but it may be overwritten during updates. Fail2ban uses the separate *jail.local* file to actually read your configuration settings.

> Open the *jail.local* file in your preferred text editor.

> Locate the [**DEFAULT**] section, which contains the following global options:

ignoreip: This option enables you to specify IP addresses or hostnames that fail2ban will ignore. For example, you could add your home or office IP address so fail2ban does not prevent you from accessing your own server. To specify multiple addresses, separate them with a space.

bantime: This option defines in seconds how long an IP address or host is banned. The default is 600 seconds (10 minutes).

maxretry: This option defines the number of failures a host is allowed before it is banned.

findtime: This option is used together with the **maxretry** option. If a host exceeds the **maxretry** setting within the time period specified by the **findtime** option, it is banned for the length of time specified by the **bantime** option.

With fail2ban's global options configured, you are now ready to enable and disable jails for the specific protocols and services you want to protect. By default, fail2ban monitors SSH login attempts (you can search for the [**ssh-iptables**] section in the *jail.local* file to view the specific settings for the SSH jail).

The *jail.local* file includes default jail settings for several protocols. Often, all you need to do to enable a jail is change its **enabled = false** line to **enabled = true** and restart fail2ban. You can also define custom jails and filters for additional flexibility

- Save your changes to the jail.local file.
- Restart the fail2ban service and load the new configuration
  #service fail2ban restart
  To display a list of IP addresses currently banned by fail2ban, type the following command:
  #iptables -S

2. IP tables

Iptables is a great firewall included in the net filter framework of Linux. A firewall is a network security system that monitors and controls the incoming and outgoing network traffic based on predetermined security rules.

Iptables rules:

a. MANGLE
   1. Rules to modify the packets
b. NAT (Network Address Translation)
   1. PREROUTING
   2. POSTROUTING
c. FILTER

1. INPUT
2. OUTPUT
3. FORWARD

The iptables rules manage the packets of a specific protocol, for example, if you want to deny an internet connection iptables can do it.

See what rules are already configured:
 # iptables -L

This allows anyone accesses to anything from anywhere and deletes the rules of iptables:

 # iptables -F

Policies:

a. ACCEPT – Allow the traffic
b. DROP – Deny the traffic

If you want to change the policies you can do it with the following command:
#iptables -P CHAIN POLITICS

Allowing the packets from your LAN:
 # iptables -A INPUT -s 192.168.100.0/24 -j ACCEPT

Allowing the internet traffic:

 # iptables -A INPUT -m state –state ESTABLISHED,RELATED -j ACCEPT

Allowing all outbound traffic:

 # iptables -A OUTPUT -j ACCEPT

Allowing HTTP and HTTPS connections from anywhere (the normal ports for websites):

 # iptables -A INPUT -p tcp –dport 80 -j ACCEPT

 # iptables -A INPUT -p tcp –dport 443 -j ACCEPT

Allowing SSH connections. The –dport number is the same as in /etc/ssh/sshd_config):

 # iptables -A INPUT -p tcp -m state –state NEW –dport 22 -j ACCEPT

Blocking an ip address with iptables – The Politics for INPUT must be DROP:

Add a new rule to drop the traffic for the correspondent ip address (

 # iptables -A INPUT -s 66.211.214.131 -j DROP

Add a new rule to allow the rest of the internet traffic (All the rules to drop traffic must be created before this rule:

 # iptables -A INPUT -m state –state ESTABLISHED,RELATED -j ACCEPT

Common iptables options:

-A	Append, this option is to add a new rule
-I	Insert a new rule
-D	Delete a rule
-R	Change the position of a rule
-L	List the rules
-L –line-numbers	Show the position number of each rule
-F	Delete all the rules
-F CHAIN	Delete the rules of an specific chain
-N CHAIN_NAME	Create a new chain
-X CHAIN	Delete a chain
-P	Change a politics
iptables -A CHAIN **-s**	Specify a source (ip address)
iptables -A CHAIN **-p**	Specify the protocol
iptables -A CHAIN -p tcp **–dport**	Specify the port
iptables -A CHAIN … **-j**	Determine a politics for an specific rule

3. Cron Jobs

The cron daemon on Linux runs tasks in the background at specific times; it's like the Task Scheduler on Windows. Add tasks to your system's crontab files using the appropriate syntax and cron will automatically run them for you.

Crontab files can be used to automate backups, system maintenance and other repetitive tasks. The syntax is powerful and flexible, so you can have a task run every fifteen minutes or at a specific minute on a specific day every year.

To open your user account's crontab file:

#crontab -e

This example runs a file named mail.php under the username of 'example_username'. This should be the same username you're currently logged in under. This example runs the cron job at 8:13 pm.

Custom cron job

MAILTO="user@example.com"

13 20 * * * php /home/example_username/mail.php

Syntax:

 # * * * * * command to execute

| | | | |

| | | | |

| | | | └──── day of week (0 - 6) (0 to 6 are Sunday to Saturday, or use names; 7 is Sunday, the same as 0)

| | | └────────── month (1 - 12)

| | └──────────────── day of month (1 - 31)

| └────────────────────── hour (0 - 23)

└──────────────────────────── min (0 - 59)

Replace your existing crontab with your custom crontab file:

 # crontab /home/username/filename

Edit your server's crontab:

 # crontab -e

View your crontab:

 # crontab -l

Remove your crontab:

#crontab -r

Examples:

Example 1: This runs a command at 4:10 PM PDT/PST, and emails you the regular and error output to the destination specified by MAILTO.

```
10 16 * * * perl /home/username/bin/yourscript.pl
```

Example 2: This will tar all files in the public_html folder and create an archive in the FILES-backup folder using today's date/time.

```
0 2 * * 6 tar -zcf FILES-backup/"$(date '+\%m-\%d-
\%y').tar.gz" public_html
```

Example 3: This runs a PHP script called cron.php at the top of every hour.

```
0 * * * * php /home/username/cron.php
```

Example 4: This will dump a MYSQL database in the FILES-backup folder using today's date/time.

```
0 * * * * mysqldump -u Jimmy -pJimmyPW JimmyDB > DB-
backups/$(date +\%m-\%d-\%Y-\%H.\%M.\%S)-JIMMY-db.sql
```

4. SNORT

Snort is an open source network intrusion prevention system, capable of performing real-time traffic analysis and packet logging on IP networks. It can perform protocol analysis, content searching/matching, and can be used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts, and much more.

 # snort -d -l /var/log/snort/ -h 10.0.0.0/24 -A console -c /etc/snort/snort.conf

d= tells snort to show data
l= determines the logs directory
h= specifies the network to monitor
A= instructs snort to print alerts in the console
c= specifies Snort the configuration file

Since we instructed Snort to save logs, we can read them by running*:*

 # snort -r

Snort's NIDS mode works based on rules specified in the /etc/snort/snort.conf file.

Within the snort.conf file we can find commented and uncommented rules

The rules path normally is /etc/snort/rules

Creating our own rule:

For example creating a new rule to notify about incoming SSH connections. Open /etc/snort/rules/yourrule.rules, and inside paste the following text:

#alert tcp $EXTERNAL_NET any -> $HOME_NET 22 (msg:"SSH incoming"; flow:stateless; flags:S+; sid:100006927; rev:1;)

We are telling Snort to alert about any tcp connection from any external source to our ssh port (in this case the default port) including the text message “SSH INCOMING”, where stateless instructs Snort to ignore the connection’s state.

Now, add the rule you created to /etc/snort/snort.conf file. Open the config file in an editor and search for #7, which is the section with rules. Add an uncommented rule like;

include $RULE_PATH/yourrule.rules