Linux Security

Lecture: Basic Setup Before Building A Working Server Practice

a) Login to your remote server via SSH connection. You can use either use Terminal (Mac / Linux) or Putty in Windows. You should login as root.

- **Change default password for root**

```
# passwd
```

- **Create new user**

```
#/usr/sbin/adduser newuser
```

- setup password for that user

```
#passwd newuser
```

- **Setup root privileges to that user**

```
#/usr/sbin/visudo
```

```
## Allow root to run any commands anywhere
root      ALL=(ALL)         ALL
newuser   ALL=(ALL)         ALL
```

- **Change SSH default port and disable root login**

```
#vi /etc/ssh/sshd_config
```

- find following lines:

```
#port 22
```

  - Remove the # symbol and change the "22" (it is default port) to to any number between 1025 and 65536.
  - Find next:

```
#PermitRootLogin yes
```

  - Remove the # symbol and change **yes** to **no**
  - Find next:

```
#UseDNS yes
```

  - Remove the # symbol and change **yes** to **no**
  - add this line in the very bottom of that file, to allow new user to login via SSH to your server.

```
AllowUsers newuser
```

- At the end, your file should contain:

```
PermitRootLogin no
```

```
UseDNS no
AllowUsers newuser
```

- Reload SSH service to make sure the new configuration is used by the service

```
#etc/init.d/sshd reload
```

- Don't close the current SSH session you're on to, before testing that the new username you created can login to your server via SSH
- Launch another Putty instance then login using new SSH port and the new username.

Lecture: Configuring firewall rules - Iptables

Prerequisites

- Iptables package should exist
- Iptables service should run
- Bind-utils package required for *host* command to run
- Jwhois package required for *whois* command to run
- VM1 IP address: <chose yours>
- VM2 IP address: <choose yours>

Basic concept understanding

- Chains – there are 3 predefined chains in the filter table to which we can add rules for processing IP packets passing through those chains:

    1. INPUT - All packets destined for the host computer.
    2. OUTPUT - All packets originating from the host computer.
    3. FORWARD - All packets neither destined for nor originating from the host computer,
       but passing through (routed by) the host computer. This chain is used if you are using your computer as a router

- Displaying the status of your firewall

```
# iptables -L -n –v
```

    - -L : List rules.
    - -v : Display detailed information. This option makes the list command show the interface name, the rule options, the packet and byte counters with the

suffix 'K', 'M' or 'G' for 1000, 1,000,000 and 1,000,000,000 multipliers respectively.
  - -n : Display IP address and port in numeric format. Do not use DNS to resolve names. This will speed up listing

- Inspect firewall with line numbers

```
# iptables -n -L -v --line-numbers
```

- Display INPUT or OUTPUT chain rules

```
# iptables -L INPUT -n –v
# iptables -L OUTPUT -n -v --line-numbers
```

- Flush all existing rules

```
# iptables –F
```

- Delete chain

```
# iptables -L INPUT -n --line-numbers
# iptables -D INPUT 2
```

- Insert firewall rules (in this example: insert rule between 1 and 2)

```
# iptables -I INPUT 2 -s 202.54.1.2 -j DROP
```

- Set the default firewall policy (such as DROP, REJECT, or ACCEPT)

```
# iptables -P INPUT DROP
```

- Save firewall rules

```
# service iptables save
```

Defining a simple rule set

1. Flush all existing rules

```
# iptables –F
```

2. Allow all incoming packets destined for the localhost interface to be accepted

```
# iptables -A INPUT -i lo -j ACCEPT
```

a. use the -A switch to append (or add) a rule to a specific chain, in our case the INPUT chain;
b. use the -i switch (for interface) to specify packets matching or destined for the lo (localhost, 127.0.0.1) interface;

c. use -j (jump) to the target action for packets matching the rule - in this case ACCEPT

3. Examine the state of a packet and determine if it is NEW, ESTABLISHED or RELATED.

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

a. We use –m switch to load a module (state); The state module is able to examine the state of a packet and determine if it is NEW, ESTABLISHED or RELATED
b. NEW refers to incoming packets that are new incoming connections that weren't initiated by the host system
c. ESTABLISHED and RELATED refers to incoming packets that are part of an already established connection or related to and already established connection.

4. Allow SSH connections over tcp port 22

```
#iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

5. Drop all incoming packets that don’t match previously defined rules

```
# iptables -P INPUT DROP
```

a. The -P switch sets the default policy on the specified chain
b. If an incoming packet does not match one of the rules, it will be dropped

6. Prevent packets passing through our server (use the server as a router)

```
# iptables -P FORWARD DROP
```

a. Similarly, here we've set the default policy on the FORWARD chain to DROP as we're not using our computer as a router so there should not be any packets passing through our computer

7. Allow all outgoing traffic

```
# iptables -P OUTPUT ACCEPT
```

- by far the easiest way to work with iptables is to create a simple script to do it all for you

```
#!/bin/bash
#
```

```
# iptables example configuration script
#
# Flush all current rules from iptables
#
 iptables -F
#
# Allow SSH connections on tcp port 22
#
 iptables -A INPUT -p tcp --dport 22 -j ACCEPT
#
# Set default policies for INPUT, FORWARD and OUTPUT chains
#
 iptables -P INPUT DROP
 iptables -P FORWARD DROP
 iptables -P OUTPUT ACCEPT
#
# Set access for localhost
#
 iptables -A INPUT -i lo -j ACCEPT
#
# Accept packets belonging to established and related connections
#
 iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
#
# Save settings
#
 /sbin/service iptables save
#
# List rules
#
 iptables -L -v
```

- Make this script executable and run it from the shell

Example 1:
Suppose we have a small network of computers that use the 192.168.0.x private subnet. We can open up our firewall to incoming packets from a single trusted IP address (for example, 192.168.0.14):

a. Accept packets from trusted IP addresses (change the IP address as appropriate)

```
#iptables -A INPUT -s 192.168.0.14 -j ACCEPT
```

- we first append (-A) a rule to the INPUT chain for the source (-s) IP address 192.168.0.14 to ACCEPT all packets

b. Add a plus of security by filtering also by IP and mac address

#iptables -A INPUT -s 192.168.0.14 -m mac --mac-source 00:50:8D:FD:E6:32 -j ACCEPT

- mac address filtering won't work across the internet but it certainly works fine on a LAN.

Example 2:
We want to block outgoing traffic to a particular host or domain such as www.youtube.com.

1. Find out all ip address of www.youtube.com

```
# host -t a www.youtube.com
```

```
www.youtube.com is an alias for youtube-ui.l.google.com.
youtube-ui.l.google.com has address 173.194.113.2
youtube-ui.l.google.com has address 173.194.113.3
youtube-ui.l.google.com has address 173.194.113.4
youtube-ui.l.google.com has address 173.194.113.5
youtube-ui.l.google.com has address 173.194.113.6
youtube-ui.l.google.com has address 173.194.113.7
youtube-ui.l.google.com has address 173.194.113.8
youtube-ui.l.google.com has address 173.194.113.9
youtube-ui.l.google.com has address 173.194.113.14
youtube-ui.l.google.com has address 173.194.113.0
youtube-ui.l.google.com has address 173.194.113.1
```

2. Find CIDR (Classless Inter-Domain Routing) for 173.252.91.4

#whois 173.194.113.2| grep CIDR

CIDR: 173.194.0.0/16

3. Prevent outgoing access to www.youtube.com

```
# iptables -A OUTPUT -p tcp -d 173.194.0.0/16 -j DROP
```

4. Domain name also can be used

```
# iptables -A OUTPUT -p tcp -d www.youtube.com -j DROP
# iptables -A OUTPUT -p tcp -d youtube.com -j DROP
```

Configuring a web proxy – Squid

Prerequisites

- Squid package should be installed
- Squid service should be running
- Squid server ip: <chose yours>

Basic concept understanding

Squid is a proxy server for caching and filtering web content. Squid proxy is used by various organization and internet providers to reduce bandwidth and to increase response time.

Squid proxy service will cache the requested web-content and re-using it for the further request of the same content.

Configure squid proxy server

- Squid configuration file location: /etc/squid/squid.conf
- There is a default minimal configuration, take a look over it
- Start squid server on system boot, on all used runlevels:

```
# chkconfig --levels 235 squid on
```

- Setup your web browser to access Internet through proxy server on port 3128.
    - Firefox: Options / Preferences » Advanced » Network » Settings » Choose "Manual proxy configuration" » Type your Proxy server ip and port no 3128
- Browse some sites and check the access log file on proxy server

```
# cat /var/log/squid/access.log
```

Configure squid proxy as web filter

1. Restricting Access to specific web sites (www.telacad.ro, www.facebook.com)
    - create a file (/etc/squid/blockedsites.squid) and add the site names one per line

```
#blocked sites
.telacad.ro
.facebook.com
```

- Open the /etc/squid/squid.conf and create a new acl ” blocksites” and acl type “dstdomain” in the acl section like the below:

```
# ACL blocksites
acl blocksites dstdomain "/etc/squid/blockedsites.squid"
```

- add the following line “http_access deny blocksites” to http_section to deny the access to the acl “blocksites”

```
# Deny access to blocksites ACL
http_access deny blocksites
```

- restart squid service
- Try to access telacad.ro and facebook.com in your browser, and check the log file to see if the request is denied

2. Restricting Access to specific keywords
   - create a file (/etc/squid/blockkeywords.squid) and add the keywords one per line: sex, porn, xxx, gambling, poker, games
   - Open the /etc/squid/squid.conf and create a new acl “blockkeywords” and acl type “url_regex” in the acl section

```
# ACL blockkeywords
acl blockkeywords url_regex -i "/etc/squid/blockkeywords.squid"
```

- add the following line “http_access deny blockkeywords” to http_section to deny the access to the acl “blockkeywords”

```
# Deny access to blockkeywords ACL
http_access deny blockkeywords
```

3. Restricting Access to specific ip address
   - create a file (/etc/squid/blockip.squid) and add the ip adresses one per line (you can put the ip address of your colleagues web servers)
   - Open the /etc/squid/squid.conf and create a new acl “blockip” and acl type “src” in the acl section

```
# ACL blockip
acl blockip src "/etc/squid/blockip.squid"
```

- add the following line “http_access deny blockip” to http_section to deny the access to the acl “blockip”

```
# Deny access to blockip ACL
http_access deny blockip
```

4. Allow Full access to specific ip address (adding excepts)
    - create a file "/etc/squid/allowip.squid " and add the ip address one per line and create an acl "allowip" and acl type "src" in the acl section

```
# ACL allowip
acl allowip src "/etc/squid/allowip.squid"
```

    - add the "!allowip" in the http_access as below

```
# Deny access to blocksites ACL
http_access deny blocksites !allowip
# Deny access to blockkeywords ACL
http_access deny blockkeywords !allowip
```

5. Restricting Download size
    - Add the below line at the bottom of the http_access section

```
#Restrict download size
reply_body_max_size 240 MB all

Or

reply_body_max_size 240 MB !allowip
```

    - You can test this by going to https://www.gentoo.org/downloads/ for example, and try to download a 256MB ISO. You can increase your limit to allow downloading the requested ISO.

6. Deny connections during specific time (office hours for example)
    - Create an acl for proxy clients

```
#you can set here the users subnet or your own ip for testing purpose
acl accountant src 192.168.10.0/24
```

    - Create an acl office time for Mon-Fri, 09:00 to 18:00 (24hrs)

```
acl officetime time MTWHF 10:00-17:00
```

    - Create an acl for cancan domain (any required sites)

acl cancan dstdomain .cancan.ro

- Deny access to "http" cancan to accountant only in office times

http_reply_access deny cancan accountant officetime

Configuring Squid as Transparent Proxy

- Set the proxy option to Automatic in Firefox
- find the below line

```
# Squid normally listens to port 3128
http_port 3128
```

- and replace with

```
# Squid normally listens to port 3128
http_port 3128 intercept
```

- Create a file "/root/squidfw.sh" and add the following firewall script (modify as per your interface configuration)

```
# squid server IP
SQUID_SERVER="192.168.1.11"
# Interface connected to Internet
INTERNET="eth0"
# Interface connected to LAN
LAN_IN="eth1"
# Squid port
SQUID_PORT="3128"
# DO NOT MODIFY BELOW
# Clean old firewall
iptables -F
iptables -X
# Load IPTABLES modules for NAT and IP conntrack support
modprobe ip_conntrack
modprobe ip_conntrack_ftp
# For win xp ftp client
#modprobe ip_nat_ftp
echo 1 > /proc/sys/net/ipv4/ip_forward
# Setting default filter policy
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
# Unlimited access to loop back
```

```
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
# Allow UDP, DNS and Passive FTP
iptables -A INPUT -i $INTERNET -m state --state ESTABLISHED, RELATED -j ACCEPT
# set this system as a router for Rest of LAN
iptables --table nat --append POSTROUTING --out-interface $INTERNET -j MASQUERADE
iptables --append FORWARD --in-interface $LAN_IN -j ACCEPT
# unlimited access to LAN
iptables -A INPUT -i $LAN_IN -j ACCEPT
iptables -A OUTPUT -o $LAN_IN -j ACCEPT
# DNAT port 80 request comming from LAN systems to squid 3128 ($SQUID_PORT) aka
transparent proxy
iptables -t nat -A PREROUTING -i $LAN_IN -p tcp --dport 80 -j DNAT --to
$SQUID_SERVER:$SQUID_PORT
# if it is same system
iptables -t nat -A PREROUTING -i $INTERNET -p tcp --dport 80 -j REDIRECT --to-port
$SQUID_PORT
# DROP everything and Log it
iptables -A INPUT -j LOG
iptables -A INPUT -j DROP
```

- run the script
- Change default gateway ip to squid server ip on the user machines.
  Now you can access Internet without setting proxy in the browser settings.

Use Case – block brute force attacks (SSH server attacks)

Using IPtables to Stop SSH Brute Force Attacks

- rate-limiting requests to SSH

```
#iptables -I INPUT -p tcp --dport 22 -i eth0 -m state --state NEW -m recent --set
#iptables -I INPUT -p tcp --dport 22 -i eth0 -m state --state NEW -m recent  --update --
seconds 60 --hitcount 4 -j DROP
```

- set a log rule to see what's being dropped

```
#iptables -N LOGDROP
#iptables -A LOGDROP -j LOG
#iptables -A LOGDROP -j DROP
#iptables -I INPUT -p tcp --dport 22 -i eth0 -m state --state NEW -m recent --set
```

#iptables -I INPUT -p tcp --dport 22 -i eth0 -m state --state NEW -m recent --update --seconds 60 --hitcount 4 -j LOGDROP