Lab: Protecting your server

6COSC002W

Ayman El Hajjar

Week 9

Requirements and Notes

- **NOTE**- In some activities you need **only** your Kali Linux machine to be connected to the internet.
- In this lab, you need the following VM machines
  1. Kali Linux
  2. OWASP Vulnerable machine
- **REMEMBER:** In order for the DHCP server same the lab documents, you need to start the VMs in the **CORRECT ORDER** as above

Understanding ports

- When two computers communicate across the network, they need a way of making sure that information flows from somewhere to somewhere else.
- They do so using a connection each end of which is a socket. A socket is the term used to refer to a combination of an address and a port.
- While many network devices have what we might call physical ports that we plug ethernet cables into, the term port in IP networking refers to a designated, logical space where traffic or communication for a particular process or service is intended to go.
- Ports are given numbers from zero to 65,535. This allows many services to listen and communicate on one network interface
  - The port numbers are divided into three general ranges. Zero through 1,023 is called the well-known ports range.

  - 1,024 to 49,151 is called the registered ports range.
  - And the ports above that are called dynamic. The first range of ports is where many common services run.
  - For example, SSH usually runs on port 22. HTTPS on 443 and TP on 123 and so on.

Preventing Unauthorized access

- When we have services running on our system, chances are we want certain clients to be able to connect to them.
- But it's also likely that we want to prevent access to those systems by people who we don't intend to have connect to them.
- In order to make this happen, we can use a firewall which is software that monitors network traffic and, using sets of rules, decides whether to allow the traffic to pass through, to redirect it somewhere else, or to prevent it from reaching its intended destination.
- we will use ufw firewall on the vulnerable machine to protect our devices from unauthorized access.

ufw

Using ufw

- In a terminal, we first need to check if the firewall is active or not. Write the following command :
  - **ufw status**
  - If the status is inactive, we can type **ufw enable** to enable the firewall and activate it on system startup
  - You can check again using **ufw status**.
    - Now this is quite simple, however now we blocked all remote connections, no remote systems will be able to connect (genuine users or malicious).
    - Everything is blocked from the access externally across the network.
    - If you try from another machine to access the web server now, you will not be able to as all external communication are note allowed.
  - That's not what we want.
    1. Let us first check on the vulnerable machine which ports are actually open. you can do this by typing **sudo lsof -i -P -n | grep LISTEN**
       - You can see there is quite a few open but since the firewall is on, all external connection will be blocked.
    2. Let us first open HTTP (80) and HTTPS (443) ports. these needs to allow traffic since we have a web server. if you have noticed from the previous list of open ports, both are setup to use TCP.
       - **sudo ufw allow 80/tcp** and
       - **sudo ufw allow 443/tcp**
    3. To delete the first one (80/tcp) you type **sudo ufw delete 1** To delete the second one (443/tcp) you type **sudo ufw delete 2** and so on or
    4. You can type **sudo ufw deny 80/tcp** to deny access
  - We can specify rules in the basic format that we've seen with a port and a protocol, or we can get a little more specific, allowing or denying access to or from certain networks or addresses.
  - Let's write a rule that allows HTTP and HTTPS access only from systems on our network.

Using ufw

- To do that, you will need to use an app designation instead of an explicit coordinate protocol.
- Some apps need to have more than one port configured and to easily add common apps, we can tell UFW which one we want to configure.
- To check what apps are configured in ufw we can type
  - **sudo ufw app list**
- W can see what apps UFW knows about with UFW app list.
- We can get some more information with ufw app info and the app name. we want to allow our apache web server to be accessed by the network domain only.
- To check which protocol the app use we type
  - sudo ufw app info "Apache Full"
- To allow access for our network only.
  - **sudo ufw allow from 192.168.56.0/24 to 192.168.56.255 app "Apache Full"**
- Ifn this lab we need to exclude the attack device ip address kali. 192.168.56.101 as they are all on the same network
  - **sudo ufw deny from 192.168.56.101/24 to any app "Apache Full"**

- ufw stores rules in a series of files in ls -l /etc/ufw
- Administrators ususally write those rules in a text file -specially if they have many rules defining the access control (internally and externally)
- let us now delete all rules to continue with the next activity

IPTABLES

- Iptables is a software package that acts as a firewall.
- Rules for the Iptables firewall software can seem complex and difficult to understand.
- These rules are put together in arrangements called chains, and using these, packets are evaluated against rules one at a time.

- If a rule or condition matches the packet, whatever action is specified by that rule is taken.
- If a packet doesn't match, the next rule is evaluated against it, and so on, until there's a match, or until the end of the chain is reached.
- Chains have a default action to take if a packet gets all the way through without matching anything. Chains can refer to other chains, making it fairly easy to set up conditional flows.
- The standard chains are for:
  - input- information coming into the system.
  - forward- in the case where the system is doing that
  - routing for systems behind it
- There are also a few predefined actions that we can use either in rules, or as default actions. They are:
  - allow, which permits a packet to pass through.
  - Drop, which ignores traffic.
  - reject, which actively responds that traffic is refused.
    - The difference here is that the traffic being treated to a drop response will time out after a while, and a client whose traffic is being rejected will be immediately notified that a connection is not possible
- Let us take a look at the Iptables chains on our vulnerable machine.
- TO do so type:
  - iptables -L -n

AS you can see this is a long list of outputs. To scroll up you press on shift + PUP and to scroll down press on shift + pgDown

```
root@owaspbwa:/etc/ufw/applications.d# ipt
iptables          iptables-restore  iptables-xml
iptables-apply    iptables-save     iptunnel
root@owaspbwa:/etc/ufw/applications.d# ipt
iptables          iptables-restore  iptables-xml
iptables-apply    iptables-save     iptunnel
root@owaspbwa:/etc/ufw/applications.d# iptables -L -n
Chain INPUT (policy DROP)
target     prot opt source               destination
ufw-before-logging-input  all  --  0.0.0.0/0            0.0.0.0/0
ufw-before-input  all  --  0.0.0.0/0            0.0.0.0/0
ufw-after-input  all  --  0.0.0.0/0            0.0.0.0/0
ufw-after-logging-input  all  --  0.0.0.0/0            0.0.0.0/0
ufw-reject-input  all  --  0.0.0.0/0            0.0.0.0/0
ufw-track-input  all  --  0.0.0.0/0            0.0.0.0/0

Chain FORWARD (policy DROP)
target     prot opt source               destination
ufw-before-logging-forward  all  --  0.0.0.0/0            0.0.0.0/0
ufw-before-forward  all  --  0.0.0.0/0            0.0.0.0/0
ufw-after-forward  all  --  0.0.0.0/0            0.0.0.0/0
ufw-after-logging-forward  all  --  0.0.0.0/0            0.0.0.0/0
ufw-reject-forward  all  --  0.0.0.0/0            0.0.0.0/0

Chain OUTPUT (policy ACCEPT)
```

- We can see in the picture above that the default polic for input is DROP
- i a packet makes it all the way through the rule chain without being matched and acted on, the firewall will drop the packet. The input chain has a list of rules on it
- The chain will be evaluated in order. As our packet goes down in the chain.
- Note: If you add any rule using ufw you will be able to see it. Try **sudo ufw deny from 192.168.56.101/24 to any app "Samba"** You can see this below in the images

```
Chain ufw-user-input (1 references)
target     prot opt source               destination
DROP       udp  --  192.168.56.102       0.0.0.0/0            multiport dports 13
7,138 /* 'dapp_Samba' */
DROP       tcp  --  192.168.56.102       0.0.0.0/0            multiport dports 13
9,445 /* 'dapp_Samba' */
```

- Samba works on port 137, 138, 139 and 445
- you can check these using the command we used before for
  - **sudo ufw app info "Samba"**
- The structure of an iptable is a complicated structure. To understand iptable commands structure type **man iptables**
- Before you continue VISIT THIS WEB PAGE (CLICK HERE) to see iptables with examples

- Aside from the rule for Samba already set in UFW let us write a rule that allows access through port 80 using TCP protocol so that users can access the website
- In order to allow the access we want, we need to make a rule that will match inbound traffic to port 80.
- To do that
  - We will start by the option -A to append the chain and the name of the chain. for example INPUT.
  - we can use the D-port, or destination port option, which we can take advantage of alongside the '-p' or protocol option.
  - we also want to allow TCP on only a specific port, so we use --dport option.
  - And then, in order to act on something that matches this rule, we need to use '-j,' or jump, and a target like accept. This could also be drop or return, or a different chain that's constructed to deal with special cases. The accept target tells the firewall to allow the packet through to wherever it's trying to get. So, with all these pieces, we'll put together an argument.
- Now we can allow web server traffic (shown below):
  - We added the two ports (http port 80, and https port 443) to the ACCEPT chain - allowing traffic in on those ports.

- **sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT**
- **sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT**

- Now let us deny web server traffic access from the attacker machine
  - if you wish to block ip address 192.168.56.101 then type the command to port 80 as follows:

- **sudo iptables -A INPUT -s 192.168.56.101 -p tcp --destination-port 80 -j DROP**
- or 443 **sudo iptables -A INPUT -s 192.168.56.101 -p tcp --destination-port 443 -j DROP**

- Now let us deny all access from the attacker machine
  - if you wish to block ip address 192.168.56.101 then type the command as follows:

- **sudo iptables -A INPUT -s 192.168.56.101 -j DROP**

Save the configuration

- Now that we have all the configuration in, we can **list** the rules to see if anything is missing. (shown below):

- **sudo iptables -L -n**

- Now we can finally **save** our firewall configuration: (shown below):

- **sudo iptables-save**

Open the saved configuration file

- We can open the saved configuration file using nano tool.

- **nano iptables-save**

- Now we can finally **save** our firewall configuration: (shown below):
- You can always edit the text file and save it rather than entering rules via the command line

- **sudo iptables-save**

Flush/Delete the configuration

- If you are looking to **remove** your iptables rules, you can flush them as shown below:

- **sudo iptables -F**

Flush/Delete the configuration

- If you are looking to **remove** a specific rule then you need to use -D option (To delete) instead of -A option (To Append) with the whole rule (exactly as it is written)
- For example if you wish to allow ip address 192.168.56.101 again to the vulnerable machine then type the command as follows:

- **sudo iptables -D INPUT -s 192.168.56.101 -j DROP**