TECHNICAL DETAILS

IPTABLES SETUP:

Firewall rulesets are configured to block all incoming and outgoing traffic, except for the traffic that satisfies the specified rules. This is a strong defence mechanism because it ensures that only authorised traffic is allowed to pass through the firewall, while all other traffic is dropped. This approach is effective because it helps to prevent unpredictable and potentially malicious traffic from entering or exiting the network.

When setting up a firewall ruleset with iptables, it is important to be careful and precise because iptables processes rules in order of priority. If the order of the rules is incorrect, it can cause the firewall to malfunction or disrupt the flow of traffic. Therefore, it is important to carefully consider the order of the rules in the firewall ruleset and to test the configuration thoroughly before implementing it in a production environment. This will help to ensure that the firewall operates as intended and provides effective protection for the network.

Initial setup of packet filtering:

1. Creating user-defined chains to design the firewall.

   $ iptables -N TCP
   $ iptables -N UDP

   By default, Iptables has 3 chains [INPUT, OUTPUT and FORWARD] for the inbound, outbound and forward traffic.

2. As our main concentration is on the inbound traffic and outbound is not properly definable. Thus we set the outbound chain to accept all the traffic.

   $ iptables -P OUTPUT ACCEPT

3. Allowing the SSH traffic to enter the host machine

   $ iptables -A INPUT -p tcp --dport -s 0/0 -j ACCEPT

4. Setting up the default ruleset to drop all the inbound traffic in case something slips out of our rule.

$ iptables -P INPUT DROP

5. Making the INPUT chain to allow traffic that belongs to established connections or is related to these connections. This includes ICMP errors and echo replies. This rule is important because it ensures that the firewall does not disrupt existing connections or prevent hosts from sending and receiving important ICMP messages.

   $ iptables -A INPUT -m conntrack cnstate RELATED, ESTABLISHED -j ACCEPT

   Having the connection states, that means:

   - ESTABLISHED state indicates that the packet's linked to a connection that has seen packets in both directions, It has attempted a connection previously or already has active connection.

   - RELATED state indicates that it is starting a new connection but associated with existing connection.

6. Adding the interfaces, that doesn't require a filtering. Loopback interfaced is added which is a virtual interface that is always up and reachable as long as at least one of the IP interfaces on the switch is operational.

   $ iptables -A INPUT -I lo -j ACCEPT

7. Traffic states [NEW, ESTABLISHED, RELATED or INVALID] which makes this a "stateful" firewall rather than a less secure "stateless" one. States are tracked using the "nf_conntrack_*" kernel modules that gets automatically loaded as rules are added.

   Packets with invalid headers or checksums, invalid TCP flags, invalid ICMP messages, out of sequence packets, such packets are considered as INVAID state and dropped

   $ iptables -A INPUT -m conntrack --ctstate INVALID -j DROP
8. Accepting incoming ICMP echo requests or pings, only the first packet will count as NEW and further it will be considered as RELATED or ESTABLISHED state.

   $ iptables -A INPUT -p icmp --icmp-type 8 -m conntrack --ctstate NEW -j ACCEPT

9. Pushing the incoming TCP, UDP packets with NEW state to the newly created chains. After the accepting a connection, it will be handled by RELATED/ESTABLISHED ruleset.

$ iptables -A INPUT -p udp -m conntrack --ctstate NEW -j UDP
$ iptables -A INPUT -p tcp --syn -m conntrack --ctstate NEW -j TCP

10. Rejecting the packets that doesn't match out above ruleset and other different protocols with TCP RESET packets for TCP connections and ICMP port unreachable messages for UDP and other protocol connections

$ iptables -A INPUT -p udp -j REJECT --reject-with icmp-port-unreachable
$ iptables -A INPUT -p tcp -j REJECT --reject-with tcp-reset
$ iptables -A INPUT -j REJECT --reject-with icmp-proto-unreachable

Building the specifics of user-defined TCP and UDP chains

1. Accepting connections on the ports, 80, 443, 22 and 53 for HTTP, HTTPS, SSH and DNS traffic.

$ iptables -A TCP -p tcp --dport 80 -j ACCEPT
$ iptables -A TCP -p tcp --dport 443 -j ACCEPT
$ iptables -A TCP -p tcp --dport 22 -j ACCEPT
$ iptables -A TCP -p tcp --dport 53 -j ACCEPT
$ iptables -A UDP -p udp --dport 53 -j ACCEPT

Tricking Port Scanners

1. Port scanners are tools that are used to identify open ports on a target system. Nmap is one such utility that can be used for performing port scans. There are different types of scans that can be performed with Nmap, such as SYN scans[-sS] and UDP scans [-sU], each of which uses a different process to identify open ports.

2. SYN Scans [-sS]
   In a SYN scan, the attacker sends a SYN packet to a port to determine whether it is open or closed. If the port is open, it will respond with a SYN/ACK packet. If the port is closed, it will respond with a TCP RESET packet. When the attacker repeatedly attempts to scan the same ports, the firewall will respond to each request with a TCP RESET packet, effectively blocking the attacker's attempts to scan the ports.

$ iptables -I TCP -p tcp -m recent --update --rsource --seconds 60 --name TCP-PORTSCAN -j REJECT --reject-with tcp-reset
$ iptables -D INPUT -p tcp -j REJECT --reject-with tcp-reset
$ iptables -A INPUT -p tcp -m recent --set --rsource --name TCP-PORTSCAN -j REJECT --reject-with tcp-reset

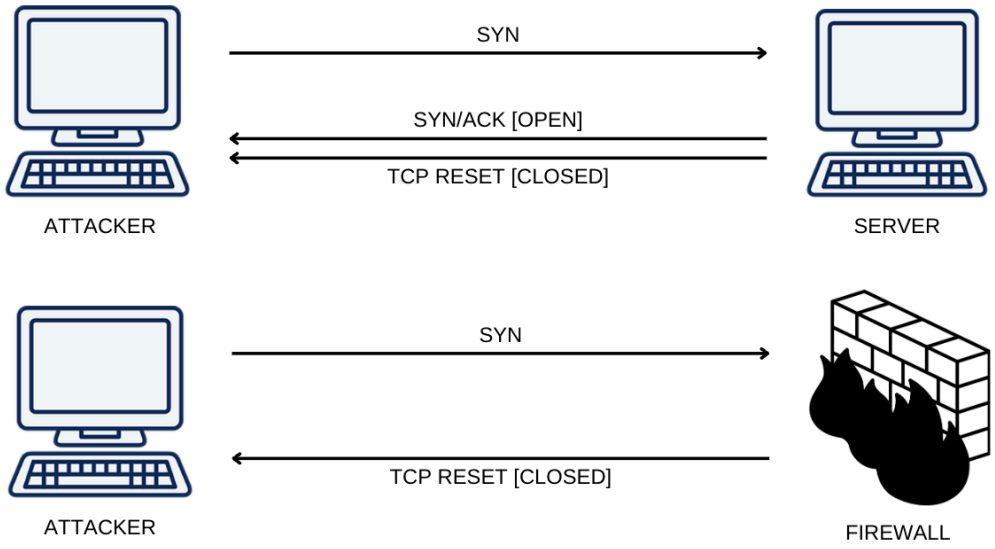

Fig. 3.1 SYN scan and Firewall reply representation

The firewall ruleset includes a counter that is set for 60 seconds for each source. This means that if the firewall responds to a SYN scan attempt with a TCP RESET packet, the counter will begin counting down from 60 seconds. Once the counter reaches zero, the firewall will reset the TCP RESET replies and allow the source to send new connection requests. This helps to prevent the firewall from blocking legitimate traffic, while still protecting against repeated SYN scan attempts.

3. UDP Scans [-sU]

   In a UDP scan, the attacker sends a UDP packet to a port to determine whether it is open or closed. Unlike TCP, which uses a handshake mechanism, UDP does not have a built-in mechanism for acknowledging received packets. Therefore, if the port is open, the attacker will not receive a reply from the server. If the port is closed, the server will send an ICMP "destination unreachable" packet after a delay. The firewall is configured to immediately send a non-delayed ICMP "destination unreachable" reply to any UDP scan attempt, and to set a 60-second counter for the source as described above. This helps to prevent the attacker from using a UDP scan to determine the open ports on the system.

   $ iptables -I UDP -p udp -m recent --update --rsource --seconds 60 --name UDP-PORTSCAN -j REJECT --reject-with icmp-port-unreachable
   $ iptables -D INPUT -p udp -j REJECT --reject-with icmp-port-unreachable
   $ iptables -A INPUT -p udp -m recent --set --rsource --name UDP-PORTSCAN -j REJECT --reject-with icmp-port-unreachable
   $ iptables -D INPUT -j REJECT --reject-with icmp-proto-unreachable
   $ iptables -A INPUT -j REJECT --reject-with icmp-proto-unreachable

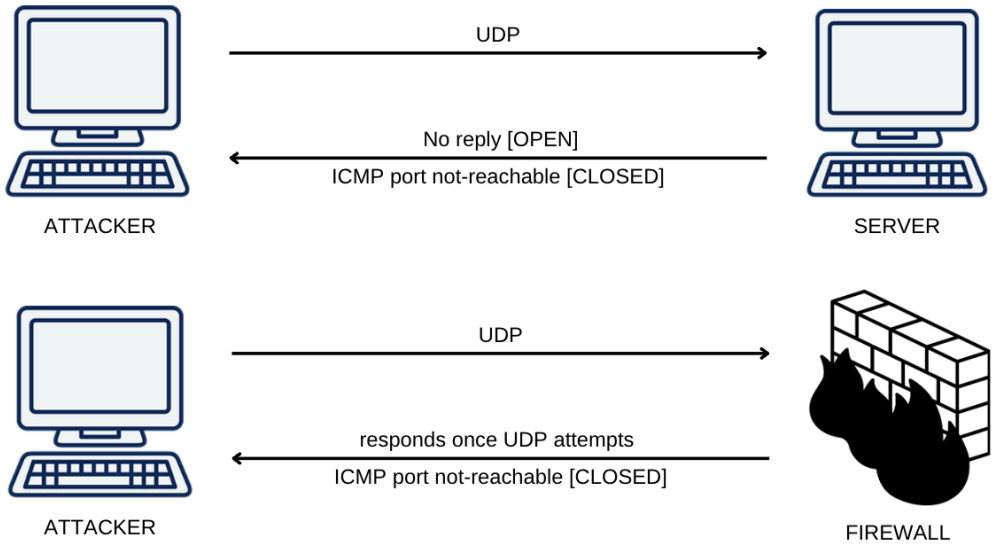

Fig. 3.2 UDP scan and Firewall reply representation

Blocking Brute Force Attacks

1. Brute force attacks are a common and easy-to-perform type of cyber attack that can be carried out using various tools. To protect against these attacks, it is possible to use firewall rulesets in iptables to set up rules that can help to prevent brute force attempts.

   $ iptables -N IN_SSH
   $ iptables -N LOG_AND_DROP
   $ iptables -A INPUT -p tcp --dport ssh -m conntrack --ctstate NEW -j IN_SSH
   $ iptables -A IN_SSH -m recent --name sshbf --rttl --rcheck --hitcount 3 --seconds 10 -j LOG_AND_DROP
   $ iptables -A IN_SSH -m recent --name sshbf --rttl --rcheck --hitcount 4 --seconds 1800 -j LOG_AND_DROP
   $ iptables -A IN_SSH -m recent --name sshbf --set -j ACCEPT
   $ iptables -A LOG_AND_DROP -j LOG --log-prefix "iptables deny: " --log-level 7
   $ iptables -A LOG_AND_DROP -j DROP

   The firewall ruleset includes a rule that allows a maximum of three connection requests, then blocks all other requests for 10 seconds. If there continues to attempt to establish connections more than four times, the firewall will block all incoming traffic from that source for 30 minutes. This rule is intended to prevent brute force attacks by making it more difficult for attackers to gain access to the system through repeated, rapid connection attempts.

Saving and Importing the rulesets

1. Exporting and Saving the built rulesets.

   $ iptables-save > ./rules_24112022_2:22PM.rules

2. Importing the rulesets to the system.

   $ iptables-restore < ./rules_24112022_2:22PM.rules

PACKET CAPTURE:

The TCPDump utility is used to continuously log network traffic on a Linux system. The traffic is logged every 65 seconds and the logs are saved in separate files. To accomplish this, the `watch` command is used to execute a script that runs TCPDump and saves the logs. The `-n 65` option specifies that the script should be run every 65 seconds. The command to run the script using `watch` would be:

$ sudo watch -n 65 ./logScript.sh

This will run the logScript.sh script every 65 seconds, using TCPDump to log the network traffic and save the logs in separate files.

```
abhisekr@Abhiseks-MacBook-Air Downloads % cat logScript.sh
#Variable
DATE=$(date '+%Y-%m-%d_%H%M%S')
SET_INTERFACE=eth0
SAVE_IN_FOLDER=/home/abhi/FirewallAndLog/Log/logs
SAVE_AS_FILE=tcpdump_$DATE.pcap

#Execute tcpdump command
sudo tcpdump -i $SET_INTERFACE -w "$SAVE_IN_FOLDER/$SAVE_AS_FILE"
exit
```

Fig. 3.3 Logging script file

Alternatively, tcpdump can be run by logging a fixed number of 1024 packets to a file and inserting them into a while loop in a bash script.

$ sudo ./logScript.sh

```
abhisekr@Abhiseks-MacBook-Air Downloads % cat logScript.sh
#!/bin/bash

while true
do
        DATE=$(date '+%Y-%m-%d_%H%M%S')
        SET_INTERFACE=eth0
        SAVE_IN_FOLDER=/home/abhi/FirewallAndLog/Log/logs
        SAVE_AS_FILE=tcpdump_$DATE.pcap

        sudo tcpdump -c 1024 -i $SET_INTERFACE -w "$SAVE_IN_FOLDER/$SAVE_AS_FILE"
done
```

Fig. 3.4 Alternative script file

ENVIRONMENT SETUP:

At the first point, all the firewall ruleset is added to a linux distort which is setup in a virtual machine on the same host machine. Host system uses SSH service to communicate with the virtual machine host.

Commands to setup SSH on the virtual machine:
$ sudo systemctl enable ssh --now
$ sudo systemctl start ssh

Above commands will enable and start the SSH service on the virtual machine, allowing you to communicate with it over the network.

References:

[1] Simple stateful firewall, https://wiki.archlinux.org/title/Simple_stateful_firewall
[2] Iptables and firewall, https://docs.ovh.com/asia/en/dedicated/firewall-iptables/
[3] Iptables ruleset saving, https://www.cyberciti.biz/faq/how-do-i-save-iptables-rules-or-settings/
[4] Port knocking, https://en.wikipedia.org/wiki/Port_knocking
[5] Nmap, https://en.wikipedia.org/wiki/Nmap