What is iptables?

IPTables is the default command-line-based stateless firewall for many Linux machines. In short, it is the thing that decides what happens to all incoming and outgoing packets. When a connection tries to establish itself to a system, IPTables matches it to a rule in its list(if none match default action is taken). Important note here is that the order of the rules in IPTables matters because it starts checking from the first rule and goes down the chain.

General IPTables format : iptables → Tables → Chains → Rules

*IMPORTANT : if you are using ufw, its settings will override any iptables rules!

iptables chains		
INPUT Chain used to control the behavior of incoming connections Ex. a user attempting to SSH into your server: iptables will match the user's IP address and port to a rule in the input chain.	**FORWARD** Chain used for incoming connections that aren't actually being delivered locally. Primarily used on pass-through devices like routers. Ex. NAT, routing, etc.	**OUTPUT** Chain used for outgoing connections Ex. you attempt to ping google.com : iptables will check what rules are regarding ping and google.com.
POSTROUTING(NAT) Alter packets before routing; packet translation happens immediately after the packet comes to the system(and before routing). An example use case is translating the destination IP address of the packet to something that matches the routing on the local machine; used for destination NAT(DNAT).	**PREROUTING(NAT)** Alter packets after routing; packet translation happens when the packers are leaving the system. Used for source NAT(SNAT).	**OUTPUT(NAT)** NAT for locally generated packets on the firewall

iptables target values	

ACCEPT	Firewall will accept the packet.
	Example of pinging INPUT ACCEPT :
	`C:\Users\geek>ping 192.168.6.129`
	
	`Pinging 192.168.6.129 with 32 bytes of data:`
	`Reply from 192.168.6.129: bytes=32 time<1ms TTL=64`
	`Reply from 192.168.6.129: bytes=32 time<1ms TTL=64`
	`Reply from 192.168.6.129: bytes=32 time<1ms TTL=64`
	`Reply from 192.168.6.129: bytes=32 time<1ms TTL=64`
	
	`Ping statistics for 192.168.6.129:`
	`Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),`
	`Approximate round trip times in milli-seconds:`
	`Minimum = 0ms, Maximum = 0ms, Average = 0ms`

DROP	Firewall will drop the packet. Drop/ignore the connection. Use this option if you don't want the sender to realize your system exists.
	`C:\Users\geek>ping 192.168.6.129`
	
	`Pinging 192.168.6.129 with 32 bytes of data:`
	`Request timed out.`
	`Request timed out.`
	`Request timed out.`
	`Request timed out.`
	
	`Ping statistics for 192.168.6.129:`
	`Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),`

QUEUE	Firewall will pass the packet to the userspace.

RETURN / REJECT	Firewall will stop executing the next set of rules in the current chain for this packet. The control will be returned to the calling chain.Use this option if you don't want the sender to establish a connection but want them to know that your firewall blocked them.
	`C:\Users\geek>ping 192.168.6.129`
	
	`Pinging 192.168.6.129 with 32 bytes of data:`
	`Reply from 192.168.6.129: Destination port unreachable.`
	`Reply from 192.168.6.129: Destination port unreachable.`
	`Reply from 192.168.6.129: Destination port unreachable.`
	`Reply from 192.168.6.129: Destination port unreachable.`
	
	`Ping statistics for 192.168.6.129:`
	`Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),`

Policy Chain DEFAULT behavior

When no matches are found for a specific packet, iptables will default to a set of pre-defined rules.To check what your iptables default policy chains are :

```
# iptables -L | grep policy
┌──(root㉿pwner3000)-[/home/deusxmachina]
└─# iptables -L | grep policy
Chain INPUT (policy ACCEPT)
Chain FORWARD (policy ACCEPT)
Chain OUTPUT (policy ACCEPT)
```

To change the default policy chain rules :

```
# iptables --policy <chain-name> <target-value>
# iptables --policy INPUT DROP
# iptables --policy OUTPUT ACCEPT
# iptables --policy FORWARD DROP
# iptables -L | grep policy
┌──(root㉿pwner3000)-[/home/deusxmachina]
└─# iptables -L | grep policy
Chain INPUT (policy DROP)
Chain FORWARD (policy DROP)
Chain OUTPUT (policy ACCEPT)
```

Allowing or blocking specific connections

Aside from the default policy chains, you can configure additional rules to allow or block specific connections. You can either append a rule :

```
# iptables -A <chain-name> <rule-specification> [options]

{block all connections from IP address 10.0.0.10}
# iptables --append INPUT --source 10.0.0.10 -j DROP
# iptables -A INPUT -s 10.0.0.10 -j DROP                    // same command as above

{allow all connections to IP addresses 10.0.0.1 ~ 10.0.0.255}
# iptables --append OUTPUT --destination 10.0.0.0/24 -j ACCEPT
```

```
# iptables -A OUTPUT -d 10.0.0.0/24 -j ACCEPT                    // same command as above
# iptables -A OUTPUT -d 10.0.0.0/255.255.255.0 -j ACCEPT         // same command as above

{block SSH connections from 10.0.0.10 - assuming SSH is running on port #22}
# iptables -A INPUT -p tcp --dport 22 -s 10.0.0.10 -j DROP
# iptables -A INPUT -p tcp --dport ssh -s 10.0.0.10 -j DROP      // same command as above

{block SSH connections from anywhere}
# iptables -A INPUT -p tcp --dport ssh -j DROP
```

Solve two-way communication problems with ‘Connection States’

You may have recognized a problem with the previous SSH filtering rules :
Many protocols require a two-way communication to function. Ex. when you ping www.google.com you need an OUTPUT rule allowing ping to the target AND an INPUT rule to retrieve the ping response. In the case of SSH, our previous OUTPUT and INPUT ACCEPT configuration would allow both external users to connect to our SSH server and us to connect to their SSH server. What if we want only external users to connect to our SSH(and block us from establishing a connection to their SSH)?
In the following example, we allow SSH connections FROM 10.0.0.1

```
# iptables -A INPUT -p tcp --dport ssh -s 10.0.0.1 -m state --state NEW,ESTABLISHED -j ACCEPT
```

But new SSH connections TO 10.0.0.1 are not. However, the system is permitted to send back information over SSH as long as the session has already been established(as in not initiating the connection).

```
# iptables -A OUTPUT -p tcp --sport 22 -d 10.0.0.1 -m state --state ESTABLISHED -j ACCEPT
(default deny any)
```

Save iptables configuration

To prevent current iptables configured rules from disappearing upon the next service reboot, run the following command(for Ubuntu)

```
# iptables-save
```