IPTABLES: PERSONAL FIREWALL TO PROTECT MY LAPTOP

Posted on August 31, 2014Author [Rp](https://tech.meetrp.com/blog/author/rp/)

Firewall! What a high sounding word! Means high protection & a safe cocoon for all the newbies. Thats exactly what I did setup in my laptop: A firewall! A personalized firewall.

My use cases & reasons were very simple. Every once in a while, I expose my laptop to unsafe open Internet like the cafes, restaurants & hotel-accommodations. Apart from that, thanks to my work, I run many services like apache2, sshd, mongodb, mysqld, etc in my laptop, which are susceptible to malicious attacks. Club these two & I got a time-bomb ticking right on my lap!!

I did a bit of research and finally (I think & I hope this is final!) I have arrived at the min-ship requirement for my laptop to function correctly under any network without compromising itself!! These are things I felt like handling within my firewall:

1. Enable (or Disable) a few kernel features
2. Make the default rule to DROP instead of the ACCEPT
3. Allow all packets from RELATED/ESTALBLISHED connectons
4. Always allow loopback devices
5. Drop all IANA reserved IPs
6. Allow skype incoming
7. Allow DHCP outgoing
8. Allow DNS outgoing
9. Allow HTTP outgoing
10. Allow NTP outgoing
11. Allow ping outgoing
12. Allow SMTP outgoing
13. *Allow SSH outgoing*

Simple, yeah?! 🙂

*All of these steps are captured in my script –*[*firewall.txt*](https://tech.meetrp.com/blog/wp-content/uploads/2014/08/firewall.txt)*(Updated script:*[*meetrp github*](https://github.com/meetrp/personalfirewall)*). Just executing the script will enable everything as described above. But if you want to understand or wanna do them one-by-one yourselves then continue reading! 🙂*

**Enable (or Disable) a few kernel features**  
The common rule in protecting oneself is: “**Deactivate everything you do not need**.” Keeping in line with this principle, I have disabled (or enabled) a few kernel parameters to protect my laptop from malicious (or unwanted) intrusion.

1. **Ignore the broadcast pings**: ICMP echo messages are the messages used by the “ping” command-line tool. By ignoring broadcast ICMP echo requests, your machine won’t respond when someone tries to ping a broadcast address (such as 255.255.255.255, or, say, 192.168.1.255 on a 192.168.1.0/24 subnet) to find all the hosts on the network or subnet at the same time.

$> echo 1 > /proc/sys/net/ipv4/icmp\_echo\_ignore\_broadcasts

1. **Deactivate source routed packets**: Attackers could be using source-routed packets to generate traffic that seems to be intra-net, but actually was created outside and has been redirected.

$> for iter in /proc/sys/net/ipv4/conf/\*/accept\_source\_route; do; echo 0 > $iter; done

1. **Disable ICMP redirects**: ICMP redirects are used by routers to specify better routing paths out of one network, based on the host choice, so basically it affects the way packets are routed and destinations. The atacker can then on basically alter your host’s routing tables and diver traffic towards external hosts on a path of his/her choice; the new path is kept active by the router for 10 minutes.

$> for iter in /proc/sys/net/ipv4/conf/\*/accept\_redirects; do; echo 0 > $iter; done

1. **Disable IP forwarding**: If there are mulitple network interfaces (like eth0, eth1, wlan0) active at the same time, then traffic coming in from one interface can be forwarded to another interface. This feature is not required in a traditional laptop

$> echo 0 > /proc/sys/net/ipv4/ip\_forward

1. **Turn on source address verfication**: By default, routers route everything, even packets which ‘obviously’ don’t belong on your network. A common example is private IP space escaping onto the Internet. If you have an interface with a route of 195.96.96.0/24 to it, you do not expect packets from 212.64.94.1 to arrive there. Enabling this verification implies if the reply to a packet wouldn’t go out the interface this packet came in, then this is a bogus packet and should be ignored.

$> for iter in /proc/sys/net/ipv4/conf/\*/rp\_filter; do; echo 1 > $iter; done

1. **Turn on syn cookies protection**: The TCP Syn is DoS (Denial of Service) attack. It consumes resources on your Linux server. The attacker begin with the TCP connection handshake sending the SYN packet, and then never completing the process to open the connection. This results into massive half-open connections.

$> echo 1 > /proc/sys/net/ipv4/tcp\_syncookies

**Where is the rules set?**  
Check the attached: [firewall.txt](https://tech.meetrp.com/blog/wp-content/uploads/2014/08/firewall.txt) (*Updated script:*[*meetrp github*](https://github.com/meetrp/personalfirewall))! Rename this file with ‘.sh’ extension & execute it.

$> ls -l ./firewall.txt

-rw-rw-r-- 1 rp rp 17288 Aug 31 00:15 ./firewall.txt

$> mv firewall.txt myfirewall.sh

$> ls -l \*firewall\*

-rwxrwxr-x 1 rp rp 17288 Aug 31 00:15 myfirewall.sh

$> chmod +x ./myfirewall.sh

$> sudo ./myfirewall.sh

[Sunday 31 August 2014 00:16:20] Not a root!

[Sunday 31 August 2014 00:16:20] ignore ICMP echo broadcasts

[Sunday 31 August 2014 00:16:20] log all packets

[Sunday 31 August 2014 00:16:21] enable reverse path filtering

[Sunday 31 August 2014 00:16:21] enable syn cookies protetion

[Sunday 31 August 2014 00:16:21] disable ICMP redirects

[Sunday 31 August 2014 00:16:21] disable ip forwarding

[Sunday 31 August 2014 00:16:21] disable source route

[Sunday 31 August 2014 00:16:21] -------------- IPv4 ---------------

[Sunday 31 August 2014 00:16:21] clear all rules

[Sunday 31 August 2014 00:16:21] default drop

[Sunday 31 August 2014 00:16:21] allow all related &amp; established

[Sunday 31 August 2014 00:16:21] allow loop back

[Sunday 31 August 2014 00:16:21] drop all IANA reserved IPs

[Sunday 31 August 2014 00:16:21] --> eth0

[Sunday 31 August 2014 00:16:21] allow skype in

[Sunday 31 August 2014 00:16:21] allow DHCP out

[Sunday 31 August 2014 00:16:21] allow DNS out

[Sunday 31 August 2014 00:16:21] allow HTTP out

[Sunday 31 August 2014 00:16:21] allow NTP out

[Sunday 31 August 2014 00:16:21] allow ping out

[Sunday 31 August 2014 00:16:21] allow SMTP out

[Sunday 31 August 2014 00:16:21] allow SSH out

[Sunday 31 August 2014 00:16:21] --> wlan0

[Sunday 31 August 2014 00:16:21] allow skype in

[Sunday 31 August 2014 00:16:21] allow DHCP out

[Sunday 31 August 2014 00:16:21] allow DNS out

[Sunday 31 August 2014 00:16:21] allow HTTP out

[Sunday 31 August 2014 00:16:21] allow NTP out

[Sunday 31 August 2014 00:16:21] allow ping out

[Sunday 31 August 2014 00:16:22] allow SMTP out

[Sunday 31 August 2014 00:16:22] allow SSH out

[Sunday 31 August 2014 00:16:22] -------------- IPv6 ---------------

[Sunday 31 August 2014 00:16:22] clear all rules

[Sunday 31 August 2014 00:16:22] default drop

This is my firewall setup script. Whenever I want, I execute this script and voila, my firewall is setup.

**Dump the IPTables for verfication**

$> sudo iptables -S

-P INPUT DROP

-P FORWARD DROP

-P OUTPUT DROP

-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

-A INPUT -i lo -j ACCEPT

-A INPUT -s 0.0.0.0/7 -j DROP

-A INPUT -s 2.0.0.0/8 -j DROP

-A INPUT -s 5.0.0.0/8 -j DROP

-A INPUT -s 7.0.0.0/8 -j DROP

-A INPUT -s 10.0.0.0/8 -j DROP

-A INPUT -s 23.0.0.0/8 -j DROP

-A INPUT -s 27.0.0.0/8 -j DROP

-A INPUT -s 31.0.0.0/8 -j DROP

-A INPUT -s 36.0.0.0/7 -j DROP

-A INPUT -s 39.0.0.0/8 -j DROP

-A INPUT -s 42.0.0.0/8 -j DROP

-A INPUT -s 49.0.0.0/8 -j DROP

-A INPUT -s 50.0.0.0/8 -j DROP

-A INPUT -s 77.0.0.0/8 -j DROP

-A INPUT -s 78.0.0.0/7 -j DROP

-A INPUT -s 92.0.0.0/6 -j DROP

-A INPUT -s 96.0.0.0/4 -j DROP

-A INPUT -s 112.0.0.0/5 -j DROP

-A INPUT -s 120.0.0.0/8 -j DROP

-A INPUT -s 169.254.0.0/16 -j DROP

-A INPUT -s 172.16.0.0/12 -j DROP

-A INPUT -s 173.0.0.0/8 -j DROP

-A INPUT -s 174.0.0.0/7 -j DROP

-A INPUT -s 176.0.0.0/5 -j DROP

-A INPUT -s 184.0.0.0/6 -j DROP

-A INPUT -s 192.0.2.0/24 -j DROP

-A INPUT -s 197.0.0.0/8 -j DROP

-A INPUT -s 198.18.0.0/15 -j DROP

-A INPUT -s 223.0.0.0/8 -j DROP

-A INPUT -s 224.0.0.0/3 -j DROP

-A INPUT -i eth0 -p udp -m udp --dport 16514 -j ACCEPT

-A INPUT -i eth0 -p tcp -m tcp --dport 16514 -j ACCEPT

-A INPUT -i wlan0 -p udp -m udp --dport 16514 -j ACCEPT

-A INPUT -i wlan0 -p tcp -m tcp --dport 16514 -j ACCEPT

-A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

-A OUTPUT -o lo -j ACCEPT

-A OUTPUT -o eth0 -p udp -m udp --sport 67:68 --dport 67:68 -j ACCEPT

-A OUTPUT -o eth0 -p udp -m udp --dport 53 -j ACCEPT

-A OUTPUT -o eth0 -p tcp -m tcp --dport 80 -m state --state NEW -j ACCEPT

-A OUTPUT -o eth0 -p tcp -m tcp --dport 443 -m state --state NEW -j ACCEPT

-A OUTPUT -o eth0 -p udp -m udp --sport 123 --dport 123 -j ACCEPT

-A OUTPUT -o eth0 -p icmp -m icmp --icmp-type 8 -j ACCEPT

-A OUTPUT -o eth0 -p tcp -m tcp --dport 25 -m state --state NEW -j ACCEPT

-A OUTPUT -o eth0 -p tcp -m tcp --dport 22 -m state --state NEW -j ACCEPT

-A OUTPUT -o wlan0 -p udp -m udp --sport 67:68 --dport 67:68 -j ACCEPT

-A OUTPUT -o wlan0 -p udp -m udp --dport 53 -j ACCEPT

-A OUTPUT -o wlan0 -p tcp -m tcp --dport 80 -m state --state NEW -j ACCEPT

-A OUTPUT -o wlan0 -p tcp -m tcp --dport 443 -m state --state NEW -j ACCEPT

-A OUTPUT -o wlan0 -p udp -m udp --sport 123 --dport 123 -j ACCEPT

-A OUTPUT -o wlan0 -p icmp -m icmp --icmp-type 8 -j ACCEPT

-A OUTPUT -o wlan0 -p tcp -m tcp --dport 25 -m state --state NEW -j ACCEPT

-A OUTPUT -o wlan0 -p tcp -m tcp --dport 22 -m state --state NEW -j ACCEPT

$> sudo ip6tables -S

-P INPUT DROP

-P FORWARD DROP

-P OUTPUT DROP

If you notice I have plenty of rules for IPv4 but dropped the IPv6 entirely!! Who wants IPv6 right away?! I still get only IPv4 address.;)

If you wanna make these rules permanent and persistent then follow these steps:

**Save the rules set while networking is going down**

$> cat /etc/network/if-down.d/saveiptables

#!/bin/bash

/sbin/iptables-save > /etc/ipv4tables.rules

/sbin/ip6tables-save > /etc/ipv6tables.rules

exit 0

**Restore the rules while networking is coming up**

$> cat /etc/network/if-up.d/loadiptables

#!/bin/bash

/sbin/iptables-restore < /etc/ipv4tables.rules

/sbin/ip6tables-restore < /etc/ipv6tables.rules

exit 0

Btw, don't forget to change it to executable!

$> sudo chmod +x /etc/network/if-down.d/saveiptables /etc/network/if-up.d/loadiptable

After this, hopefully, my laptop is secure as compared to before.

FYI, these were my experiments limited to my understanding. If I can be of any help & esp vice-versa, please feel free to contact me!

*Updated script:*[*meetrp github*](https://github.com/meetrp/personalfirewall)

**Courtesy**

1. [Security - Linux StepByStep](http://linux-sxs.org/security/hunleyfw.html)
2. [The Kernel - Linux inside](http://kernel-project.kickino.org/index_moz_en.php?action=security)
3. [IPTables Tips and Tricks: More Than Just ACCEPT or DROP](http://youtu.be/D7LgjSOWCxg)
4. [Saving iptables rules to be persistent](http://terraltech.com/saving-iptables-rules-to-be-persistent/)
5. <http://hermann-uwe.de/files/fw_laptop>
6. [Laptop Iptables configuration](http://www.pantz.org/software/iptables/laptopiptables.html)
7. [iptables: Small manual and tutorial with some examples and tips](http://www.garron.me/en/linux/iptables-manual.html)