**Práctica ataque DoS**

**Definiciones:**

\* DoS: Denial of Service (denegación de servicio) es un ataque a un sistema de computadores o red que causa que un servicio o recurso sea inaccesible a los usuarios legítimos

\* DDoS: Distributed Denial of Service

\* AntiDoS: "previene" un ataque DoS

**Trabajo:**

\* Grupos de 4 personas (2 atacantes vs 2 defensores)

\* Scripts en BASH: buscarlos, probarlos y documentarlos

Descripción: buscar un scripts hecho en bash, y documentar su código

**Tips:**

\* Ataque mediante paquetes ping, etc... utilizando el protocolo icmp, o ataques web (wget) utilizando los protocolos tcp/udp (para los ataques web es necesario que el defensor tenga un servidor web instalado como apache2)

\* Defensa mediante detección de paquetes icmp con tcpdump o de paquetes tcp/udp con netstat. Después de detectar la dirección ip atacante se suele cortar mediante firewall (iptables)

Es posible que sea necesario instalar algunos programas para que los scripts funcionen como hping3, etc…

HPING3 (ATAQUE)

<https://www.redeszone.net/tutoriales/seguridad/hping3-manipular-paquetes-tcp-ip-ataques/>

<https://youtu.be/gxL4Mf41qK0>

—---------------------------------------------------------------------------------------------------------------

<https://blog.zerial.org/seguridad/ddos-attacker-script-peticiones-automatizadas-para-lograr-dos/>

DEFENSA:

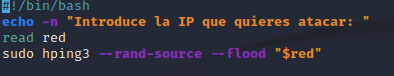
<https://blog.elhacker.net/2013/12/ddos-deflate-script-bash-para-mitigar-ataques-ddos-dos.html>

<https://sololinuxes.tumblr.com/post/189702340955/anti-ddos-bash-script>

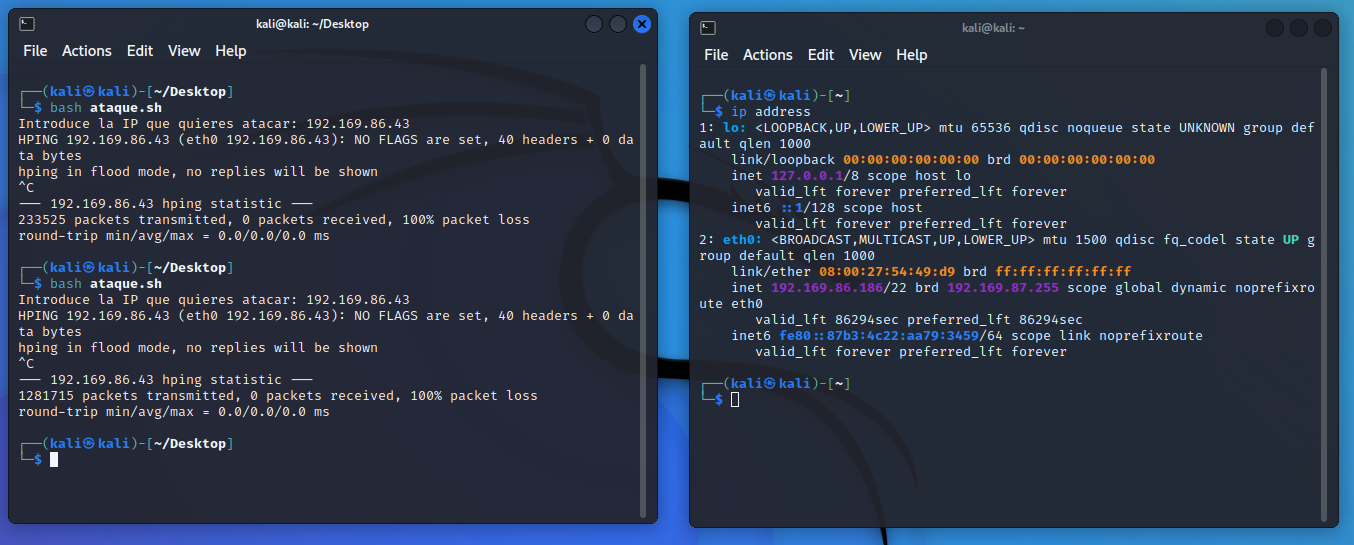
<https://github.com/anti-ddos/Anti-DDOS>

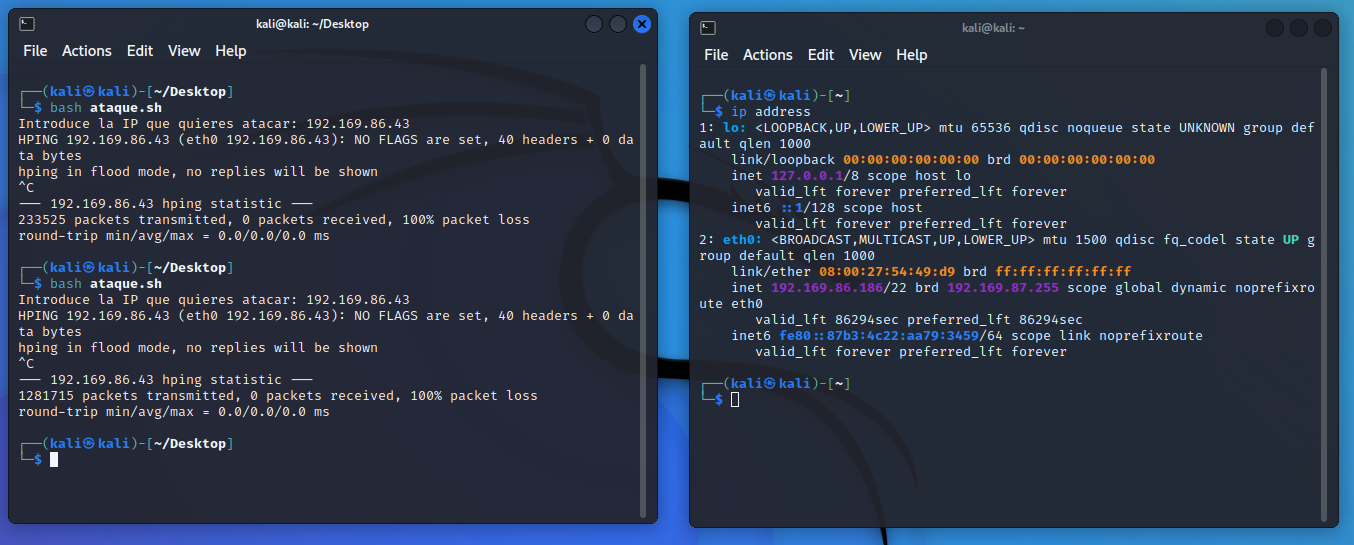
**ATAQUE DDOS EMPLEADO: HPING3**

* **SCRIPT:** realizamos un ataque con hping3, –flood para que sea masivo y –rand-source para que esté camuflada.

****

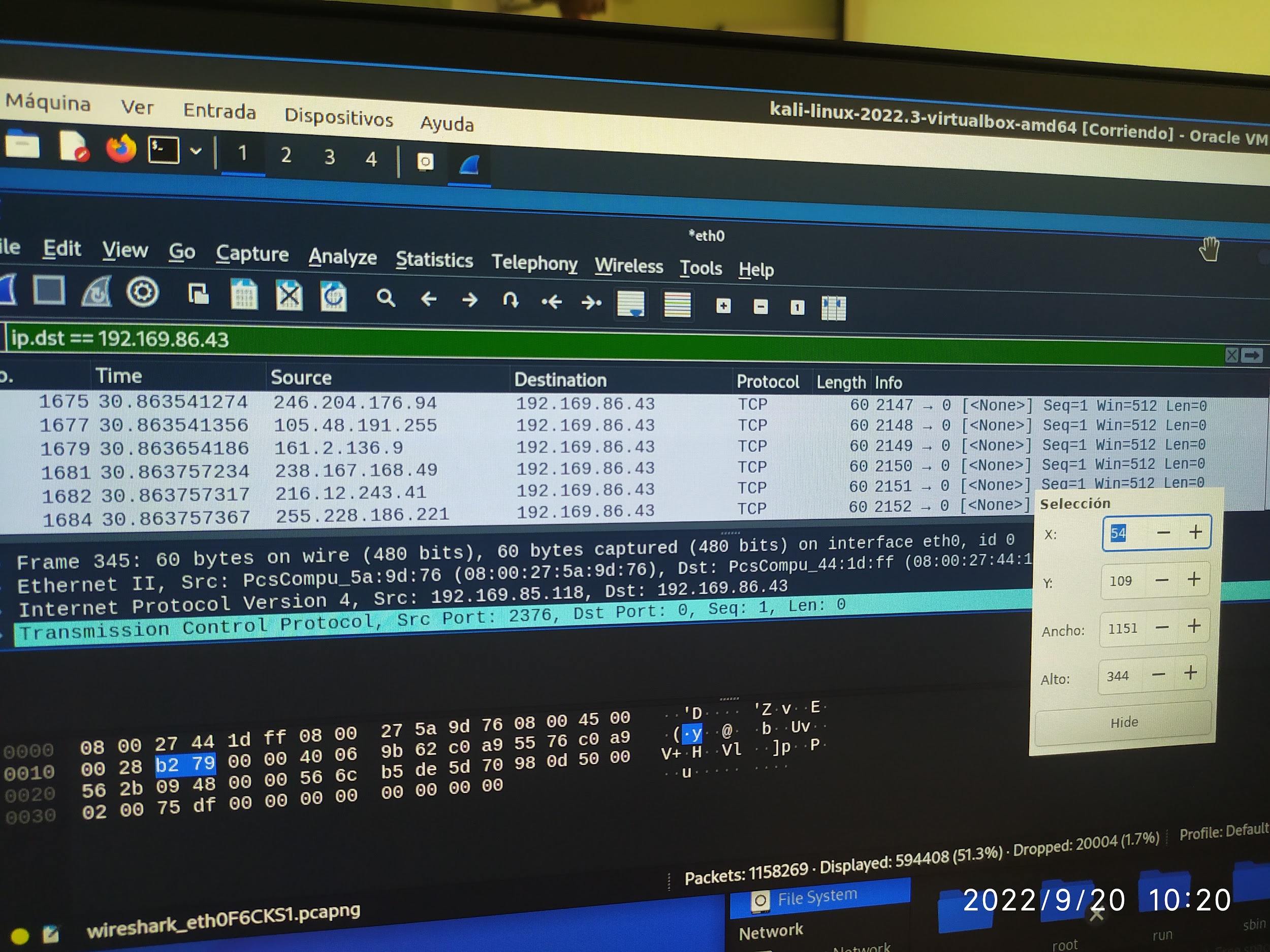
* **Desde ordenador Marco con ip 192.169.86.186:**

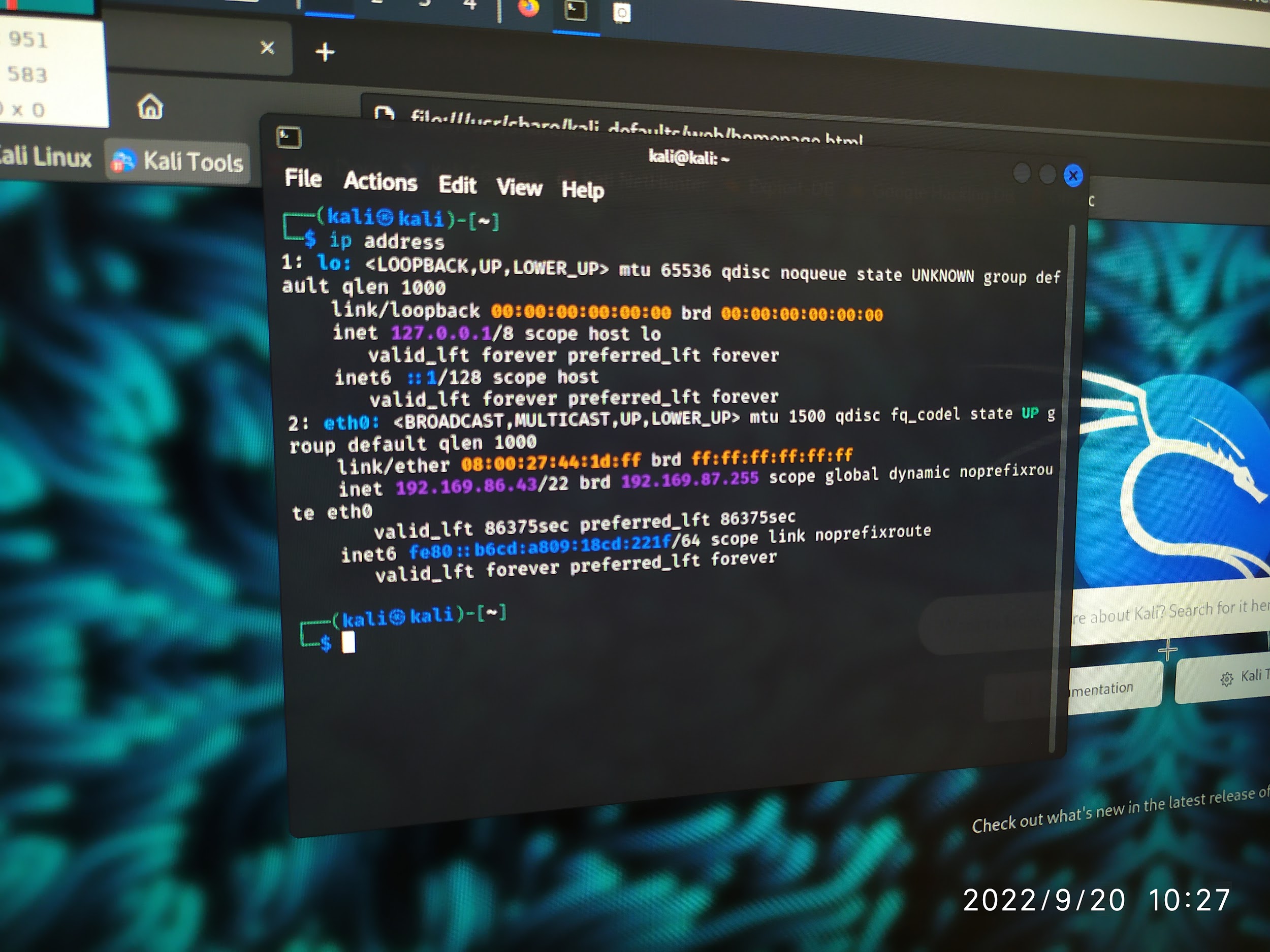




**Defensa desde ordenador rubén con ip 192.169.86.43:**

A partir de darnos cuenta de no tener acceso a internet y un exagerado tráfico de red (wireshark) nos ponemos manos a la obra porque sospechamos que puede ser un ataque Dos.





El programa de defensa frente a un ataque DDOS ,primero depura usando iptables,luego te muestra las opciones de registro ,te limita los valores por defecto de la tasa de paquetes,te muestra manda a los puertos sin privilegios ,carga los módulos de kernel necesarios,con esta mitigamos los ataques de envenenamiento provenientes de ARP u otros ataques similares.

La configuración del Kernel deshabilitamos el reenvío de ips,habilitamos la protección contra la suplantación de identidad,nos protegemos contra ataques masivos,ignoramos todas las peticiones de ECHO e ICMP.

Para las políticas por defecto lo ponemos todo por defecto,establecemos todos los tipos de tablas a modo ACCEPT,

Realizamos una limpieza eliminando todos los paquetes relacionados con iptables.

Deshabilitamos por completo todas Ipv6 bloqueando el tráfico , deshabilitamos todas la reglas, cadenas y paquetes.

Permitimos ciertas peticiones de DNS,HTTPS,HTTP,POP3,IMAP4,POP3S,SMTP,FPT,SSH,NNTP,MySQL,PLESK.

Comprobamos el firewall.

Eliminamos la reglas relacionadas con Ipv6.

#!/bin/sh

#########################################################

# ANTI-DDOS BASH SCRIPT #

#########################################################

# CONTACT #

#########################################################

# DEVELOPER : İSMAİL TAŞDELEN #

# GMAIL : ismailtasdelen@protonmail.com #

# Linkedin : https://www.linkedin.com/in/ismailtasdelen #

# Telegram : https://t.me/ismailtasdelen #

#########################################################

# For debugging use iptables -v.

IPTABLES="/sbin/iptables"

IP6TABLES="/sbin/ip6tables"

MODPROBE="/sbin/modprobe"

RMMOD="/sbin/rmmod"

ARP="/usr/sbin/arp"

SSHPORT="22"

# Logging options.

#------------------------------------------------------------------------------

LOG="LOG --log-level debug --log-tcp-sequence --log-tcp-options"

LOG="$LOG --log-ip-options"

# Defaults for rate limiting

#------------------------------------------------------------------------------

RLIMIT="-m limit --limit 3/s --limit-burst 8"

# Unprivileged ports.

#------------------------------------------------------------------------------

PHIGH="1024:65535"

PSSH="1000:1023"

# Load required kernel modules

#------------------------------------------------------------------------------

"$MODPROBE" ip\_conntrack\_ftp

"$MODPROBE" ip\_conntrack\_irc

# Mitigate ARP spoofing/poisoning and similar attacks.

#------------------------------------------------------------------------------

# Hardcode static ARP cache entries here

# $ARP -s IP-ADDRESS MAC-ADDRESS

# Kernel configuration.

#------------------------------------------------------------------------------

# Disable IP forwarding.

# On => Off = (reset)

echo 1 > /proc/sys/net/ipv4/ip\_forward

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

# Enable IP spoofing protection

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

# Protect against SYN flood attacks

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

# Ignore all incoming ICMP echo requests

echo 0 > /proc/sys/net/ipv4/icmp\_echo\_ignore\_all

# Ignore ICMP echo requests to broadcast

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

# Log packets with impossible addresses.

for i in /proc/sys/net/ipv4/conf/\*/log\_martians; do echo 1 > "$i"; done

# Don't log invalid responses to broadcast

echo 1 > /proc/sys/net/ipv4/icmp\_ignore\_bogus\_error\_responses

# Don't accept or send ICMP redirects.

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

for i in /proc/sys/net/ipv4/conf/\*/send\_redirects; do echo 0 > "$i"; done

# Don't accept source routed packets.

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

# Disable multicast routing

for i in /proc/sys/net/ipv4/conf/\*/mc\_forwarding; do echo 0 > "$i"; done

# Disable proxy\_arp.

for i in /proc/sys/net/ipv4/conf/\*/proxy\_arp; do echo 0 > "$i"; done

# Enable secure redirects, i.e. only accept ICMP redirects for gateways

# Helps against MITM attacks.

for i in /proc/sys/net/ipv4/conf/\*/secure\_redirects; do echo 1 > "$i"; done

# Disable bootp\_relay

for i in /proc/sys/net/ipv4/conf/\*/bootp\_relay; do echo 0 > "$i"; done

# Default policies.

#------------------------------------------------------------------------------

# Drop everything by default.

"$IPTABLES" -P INPUT DROP

"$IPTABLES" -P FORWARD DROP

"$IPTABLES" -P OUTPUT DROP

# Set the nat/mangle/raw tables' chains to ACCEPT

"$IPTABLES" -t nat -P PREROUTING ACCEPT

"$IPTABLES" -t nat -P OUTPUT ACCEPT

"$IPTABLES" -t nat -P POSTROUTING ACCEPT

"$IPTABLES" -t mangle -P PREROUTING ACCEPT

"$IPTABLES" -t mangle -P INPUT ACCEPT

"$IPTABLES" -t mangle -P FORWARD ACCEPT

"$IPTABLES" -t mangle -P OUTPUT ACCEPT

"$IPTABLES" -t mangle -P POSTROUTING ACCEPT

# Cleanup.

#------------------------------------------------------------------------------

# Delete all

"$IPTABLES" -F

"$IPTABLES" -t nat -F

"$IPTABLES" -t mangle -F

# Delete all

"$IPTABLES" -X

"$IPTABLES" -t nat -X

"$IPTABLES" -t mangle -X

# Zero all packets and counters.

"$IPTABLES" -Z

"$IPTABLES" -t nat -Z

"$IPTABLES" -t mangle -Z

# Completely disable IPv6.

#------------------------------------------------------------------------------

# Block all IPv6 traffic

# If the ip6tables command is available, try to block all IPv6 traffic.

if test -x "$IP6TABLES"; then

# Set the default policies

# drop everything

"$IP6TABLES" -P INPUT DROP 2>/dev/null

"$IP6TABLES" -P FORWARD DROP 2>/dev/null

"$IP6TABLES" -P OUTPUT DROP 2>/dev/null

# The mangle table can pass everything

"$IP6TABLES" -t mangle -P PREROUTING ACCEPT 2>/dev/null

"$IP6TABLES" -t mangle -P INPUT ACCEPT 2>/dev/null

"$IP6TABLES" -t mangle -P FORWARD ACCEPT 2>/dev/null

"$IP6TABLES" -t mangle -P OUTPUT ACCEPT 2>/dev/null

"$IP6TABLES" -t mangle -P POSTROUTING ACCEPT 2>/dev/null

# Delete all rules.

"$IP6TABLES" -F 2>/dev/null

"$IP6TABLES" -t mangle -F 2>/dev/null

# Delete all chains.

"$IP6TABLES" -X 2>/dev/null

"$IP6TABLES" -t mangle -X 2>/dev/null

# Zero all packets and counters.

"$IP6TABLES" -Z 2>/dev/null

"$IP6TABLES" -t mangle -Z 2>/dev/null

fi

# Custom user-defined chains.

#------------------------------------------------------------------------------

# LOG packets, then ACCEPT.

"$IPTABLES" -N ACCEPTLOG

"$IPTABLES" -A ACCEPTLOG -j "$LOG" "$RLIMIT" --log-prefix "ACCEPT "

"$IPTABLES" -A ACCEPTLOG -j ACCEPT

# LOG packets, then DROP.

"$IPTABLES" -N DROPLOG

"$IPTABLES" -A DROPLOG -j "$LOG" "$RLIMIT" --log-prefix "DROP "

"$IPTABLES" -A DROPLOG -j DROP

# LOG packets, then REJECT.

# TCP packets are rejected with a TCP reset.

"$IPTABLES" -N REJECTLOG

"$IPTABLES" -A REJECTLOG -j "$LOG" "$RLIMIT" --log-prefix "REJECT "

"$IPTABLES" -A REJECTLOG -p tcp -j REJECT --reject-with tcp-reset

"$IPTABLES" -A REJECTLOG -j REJECT

# Only allows RELATED ICMP types

# (destination-unreachable, time-exceeded, and parameter-problem).

# TODO: Rate-limit this traffic?

# TODO: Allow fragmentation-needed?

# TODO: Test.

"$IPTABLES" -N RELATED\_ICMP

"$IPTABLES" -A RELATED\_ICMP -p icmp --icmp-type destination-unreachable -j ACCEPT

"$IPTABLES" -A RELATED\_ICMP -p icmp --icmp-type time-exceeded -j ACCEPT

"$IPTABLES" -A RELATED\_ICMP -p icmp --icmp-type parameter-problem -j ACCEPT

"$IPTABLES" -A RELATED\_ICMP -j DROPLOG

# Make It Even Harder To Multi-PING

"$IPTABLES" -A INPUT -p icmp -m limit --limit 1/s --limit-burst 2 -j ACCEPT

"$IPTABLES" -A INPUT -p icmp -m limit --limit 1/s --limit-burst 2 -j LOG --log-prefix PING-DROP:

"$IPTABLES" -A INPUT -p icmp -j DROP

"$IPTABLES" -A OUTPUT -p icmp -j ACCEPT

# Only allow the minimally required/recommended parts of ICMP. Block the rest.

#------------------------------------------------------------------------------

# TODO: This section needs a lot of testing!

# First, drop all fragmented ICMP packets (almost always malicious).

"$IPTABLES" -A INPUT -p icmp --fragment -j DROPLOG

"$IPTABLES" -A OUTPUT -p icmp --fragment -j DROPLOG

"$IPTABLES" -A FORWARD -p icmp --fragment -j DROPLOG

# Allow all ESTABLISHED ICMP traffic.

"$IPTABLES" -A INPUT -p icmp -m state --state ESTABLISHED -j ACCEPT "$RLIMIT"

"$IPTABLES" -A OUTPUT -p icmp -m state --state ESTABLISHED -j ACCEPT "$RLIMIT"

# Allow some parts of the RELATED ICMP traffic, block the rest.

"$IPTABLES" -A INPUT -p icmp -m state --state RELATED -j RELATED\_ICMP "$RLIMIT"

"$IPTABLES" -A OUTPUT -p icmp -m state --state RELATED -j RELATED\_ICMP "$RLIMIT"

# Allow incoming ICMP echo requests (ping), but only rate-limited.

"$IPTABLES" -A INPUT -p icmp --icmp-type echo-request -j ACCEPT "$RLIMIT"

# Allow outgoing ICMP echo requests (ping), but only rate-limited.

"$IPTABLES" -A OUTPUT -p icmp --icmp-type echo-request -j ACCEPT "$RLIMIT"

# Drop any other ICMP traffic.

"$IPTABLES" -A INPUT -p icmp -j DROPLOG

"$IPTABLES" -A OUTPUT -p icmp -j DROPLOG

"$IPTABLES" -A FORWARD -p icmp -j DROPLOG

# Selectively allow certain special types of traffic.

#------------------------------------------------------------------------------

# Allow loopback interface to do anything.

"$IPTABLES" -A INPUT -i lo -j ACCEPT

"$IPTABLES" -A OUTPUT -o lo -j ACCEPT

# Allow incoming connections related to existing allowed connections.

"$IPTABLES" -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

# Allow outgoing connections EXCEPT invalid

"$IPTABLES" -A OUTPUT -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT

# Miscellaneous.

#------------------------------------------------------------------------------

# We don't care about Milkosoft, Drop SMB/CIFS/etc..

"$IPTABLES" -A INPUT -p tcp -m multiport --dports 135,137,138,139,445,1433,1434 -j DROP

"$IPTABLES" -A INPUT -p udp -m multiport --dports 135,137,138,139,445,1433,1434 -j DROP

# Explicitly drop invalid incoming traffic

"$IPTABLES" -A INPUT -m state --state INVALID -j DROP

# Drop invalid outgoing traffic, too.

"$IPTABLES" -A OUTPUT -m state --state INVALID -j DROP

# If we would use NAT, INVALID packets would pass - BLOCK them anyways

"$IPTABLES" -A FORWARD -m state --state INVALID -j DROP

# PORT Scanners (stealth also)

"$IPTABLES" -A INPUT -m state --state NEW -p tcp --tcp-flags ALL ALL -j DROP

"$IPTABLES" -A INPUT -m state --state NEW -p tcp --tcp-flags ALL NONE -j DROP

# TODO: Some more anti-spoofing rules? For example:

# "$IPTABLES" -A INPUT -p tcp --tcp-flags ALL FIN,URG,PSH -j DROP

# "$IPTABLES" -A INPUT -p tcp --tcp-flags SYN,RST SYN,RST -j DROP

# "$IPTABLES" -A INPUT -p tcp --tcp-flags SYN,FIN SYN,FIN -j DROP

"$IPTABLES" -N SYN\_FLOOD

"$IPTABLES" -A INPUT -p tcp --syn -j SYN\_FLOOD

"$IPTABLES" -A SYN\_FLOOD -m limit --limit 2/s --limit-burst 6 -j RETURN

"$IPTABLES" -A SYN\_FLOOD -j DROP

# TODO: Block known-bad IPs (see http://www.dshield.org/top10.php).

# "$IPTABLES" -A INPUT -s INSERT-BAD-IP-HERE -j DROPLOG

# Drop any traffic from IANA-reserved IPs.

#------------------------------------------------------------------------------

"$IPTABLES" -A INPUT -s 0.0.0.0/7 -j DROP

"$IPTABLES" -A INPUT -s 2.0.0.0/8 -j DROP

"$IPTABLES" -A INPUT -s 5.0.0.0/8 -j DROP

"$IPTABLES" -A INPUT -s 7.0.0.0/8 -j DROP

"$IPTABLES" -A INPUT -s 10.0.0.0/8 -j DROP

"$IPTABLES" -A INPUT -s 23.0.0.0/8 -j DROP

"$IPTABLES" -A INPUT -s 27.0.0.0/8 -j DROP

"$IPTABLES" -A INPUT -s 31.0.0.0/8 -j DROP

"$IPTABLES" -A INPUT -s 36.0.0.0/7 -j DROP

"$IPTABLES" -A INPUT -s 39.0.0.0/8 -j DROP

"$IPTABLES" -A INPUT -s 42.0.0.0/8 -j DROP

"$IPTABLES" -A INPUT -s 49.0.0.0/8 -j DROP

"$IPTABLES" -A INPUT -s 50.0.0.0/8 -j DROP

"$IPTABLES" -A INPUT -s 77.0.0.0/8 -j DROP

"$IPTABLES" -A INPUT -s 78.0.0.0/7 -j DROP

"$IPTABLES" -A INPUT -s 92.0.0.0/6 -j DROP

"$IPTABLES" -A INPUT -s 96.0.0.0/4 -j DROP

"$IPTABLES" -A INPUT -s 112.0.0.0/5 -j DROP

"$IPTABLES" -A INPUT -s 120.0.0.0/8 -j DROP

"$IPTABLES" -A INPUT -s 169.254.0.0/16 -j DROP

"$IPTABLES" -A INPUT -s 172.16.0.0/12 -j DROP

"$IPTABLES" -A INPUT -s 173.0.0.0/8 -j DROP

"$IPTABLES" -A INPUT -s 174.0.0.0/7 -j DROP

"$IPTABLES" -A INPUT -s 176.0.0.0/5 -j DROP

"$IPTABLES" -A INPUT -s 184.0.0.0/6 -j DROP

"$IPTABLES" -A INPUT -s 192.0.2.0/24 -j DROP

"$IPTABLES" -A INPUT -s 197.0.0.0/8 -j DROP

"$IPTABLES" -A INPUT -s 198.18.0.0/15 -j DROP

"$IPTABLES" -A INPUT -s 223.0.0.0/8 -j DROP

"$IPTABLES" -A INPUT -s 224.0.0.0/3 -j DROP

# Selectively allow certain outbound connections, block the rest.

#------------------------------------------------------------------------------

# Allow outgoing DNS requests. Few things will work without this.

"$IPTABLES" -A OUTPUT -m state --state NEW -p udp --dport 53 -j ACCEPT

"$IPTABLES" -A OUTPUT -m state --state NEW -p tcp --dport 53 -j ACCEPT

# Allow outgoing HTTP requests. Unencrypted, use with care.

"$IPTABLES" -A OUTPUT -m state --state NEW -p tcp --dport 80 -j ACCEPT

# Allow outgoing HTTPS requests.

"$IPTABLES" -A OUTPUT -m state --state NEW -p tcp --dport 443 -j ACCEPT

# Allow outgoing SMTPS requests. Do NOT allow unencrypted SMTP!

# "$IPTABLES" -A OUTPUT -m state --state NEW -p tcp --dport 465 -j ACCEPT

# Allow outgoing "submission" (RFC 2476) requests.

"$IPTABLES" -A OUTPUT -m state --state NEW -p tcp --dport 587 -j ACCEPT

# Allow outgoing POP3S requests.

"$IPTABLES" -A OUTPUT -m state --state NEW -p tcp --dport 995 -j ACCEPT

# Allow outgoing SSH requests.

"$IPTABLES" -A OUTPUT -m state --state NEW -p tcp --dport "$SSHPORT" -j ACCEPT

# Allow outgoing FTP requests. Unencrypted, use with care.

"$IPTABLES" -A OUTPUT -m state --state NEW -p tcp --dport 21 -j ACCEPT

# Allow outgoing NNTP requests. Unencrypted, use with care.

# "$IPTABLES" -A OUTPUT -m state --state NEW -p tcp --dport 119 -j ACCEPT

# Allow outgoing NTP requests. Unencrypted, use with care.

# "$IPTABLES" -A OUTPUT -m state --state NEW -p udp --dport 123 -j ACCEPT

# Allow outgoing IRC requests. Unencrypted, use with care.

# Note: This usually needs the ip\_conntrack\_irc kernel module.

# "$IPTABLES" -A OUTPUT -m state --state NEW -p tcp --dport 6667 -j ACCEPT

# Allow outgoing requests to various proxies. Unencrypted, use with care.

# "$IPTABLES" -A OUTPUT -m state --state NEW -p tcp --dport 8080 -j ACCEPT

# "$IPTABLES" -A OUTPUT -m state --state NEW -p tcp --dport 8090 -j ACCEPT

# Allow outgoing DHCP requests. Unencrypted, use with care.

# TODO: This is completely untested, I have no idea whether it works!

# TODO: I think this can be tightened a bit more.

"$IPTABLES" -A OUTPUT -m state --state NEW -p udp --sport 67:68 --dport 67:68 -j ACCEPT

# Allow outgoing CVS requests. Unencrypted, use with care.

# "$IPTABLES" -A OUTPUT -m state --state NEW -p tcp --dport 2401 -j ACCEPT

# Allow outgoing MySQL requests. Unencrypted, use with care.

# "$IPTABLES" -A OUTPUT -m state --state NEW -p tcp --dport 3306 -j ACCEPT

# Allow outgoing SVN requests. Unencrypted, use with care.

# "$IPTABLES" -A OUTPUT -m state --state NEW -p tcp --dport 3690 -j ACCEPT

# Allow outgoing PLESK requests. Unencrypted, use with care.

# "$IPTABLES" -A OUTPUT -m state --state NEW -p tcp --dport 8443 -j ACCEPT

# Allow outgoing Tor (http://tor.eff.org) requests.

# Note: Do \_not\_ use unencrypted protocols over Tor (sniffing is possible)!

# "$IPTABLES" -A OUTPUT -m state --state NEW -p tcp --dport 9001 -j ACCEPT

# "$IPTABLES" -A OUTPUT -m state --state NEW -p tcp --dport 9002 -j ACCEPT

# "$IPTABLES" -A OUTPUT -m state --state NEW -p tcp --dport 9030 -j ACCEPT

# "$IPTABLES" -A OUTPUT -m state --state NEW -p tcp --dport 9031 -j ACCEPT

# "$IPTABLES" -A OUTPUT -m state --state NEW -p tcp --dport 9090 -j ACCEPT

# "$IPTABLES" -A OUTPUT -m state --state NEW -p tcp --dport 9091 -j ACCEPT

# Allow outgoing OpenVPN requests.

"$IPTABLES" -A OUTPUT -m state --state NEW -p udp --dport 1194 -j ACCEPT

# TODO: ICQ, MSN, GTalk, Skype, Yahoo, etc...

# Selectively allow certain inbound connections, block the rest.

#------------------------------------------------------------------------------

# Allow incoming DNS requests.

"$IPTABLES" -A INPUT -m state --state NEW -p udp --dport 53 -j ACCEPT

"$IPTABLES" -A INPUT -m state --state NEW -p tcp --dport 53 -j ACCEPT

# Allow incoming HTTP requests.

"$IPTABLES" -A INPUT -m state --state NEW -p tcp --dport 80 -j ACCEPT

# Allow incoming HTTPS requests.

"$IPTABLES" -A INPUT -m state --state NEW -p tcp --dport 443 -j ACCEPT

# Allow incoming POP3 requests.

"$IPTABLES" -A INPUT -m state --state NEW -p tcp --dport 110 -j ACCEPT

# Allow incoming IMAP4 requests.

"$IPTABLES" -A INPUT -m state --state NEW -p tcp --dport 143 -j ACCEPT

# Allow incoming POP3S requests.

"$IPTABLES" -A INPUT -m state --state NEW -p tcp --dport 995 -j ACCEPT

# Allow incoming SMTP requests.

"$IPTABLES" -A INPUT -m state --state NEW -p tcp --dport 25 -j ACCEPT

# Allow incoming SSH requests.

"$IPTABLES" -A INPUT -m state --state NEW -p tcp --dport "$SSHPORT" -j ACCEPT

# Allow incoming FTP requests.

"$IPTABLES" -A INPUT -m state --state NEW -p tcp --dport 21 -j ACCEPT

# Allow incoming NNTP requests.

# "$IPTABLES" -A INPUT -m state --state NEW -p tcp --dport 119 -j ACCEPT

# Allow incoming MySQL requests.

# "$IPTABLES" -A INPUT -m state --state NEW -p tcp --dport 3306 -j ACCEPT

# Allow incoming PLESK requests.

# "$IPTABLES" -A INPUT -m state --state NEW -p tcp --dport 8843 -j ACCEPT

# Allow incoming BitTorrent requests.

# TODO: Are these already handled by ACCEPTing established/related traffic?

# "$IPTABLES" -A INPUT -m state --state NEW -p tcp --dport 6881 -j ACCEPT

# "$IPTABLES" -A INPUT -m state --state NEW -p udp --dport 6881 -j ACCEPT

# Allow incoming nc requests.

# "$IPTABLES" -A INPUT -m state --state NEW -p tcp --dport 2030 -j ACCEPT

# "$IPTABLES" -A INPUT -m state --state NEW -p udp --dport 2030 -j ACCEPT

# Explicitly log and reject everything else.

#------------------------------------------------------------------------------

# Use REJECT instead of REJECTLOG if you don't need/want logging.

"$IPTABLES" -A INPUT -j REJECTLOG

"$IPTABLES" -A OUTPUT -j REJECTLOG

"$IPTABLES" -A FORWARD -j REJECTLOG

#------------------------------------------------------------------------------

# Testing the firewall.

#------------------------------------------------------------------------------

# You should check/test that the firewall really works, using

# iptables -vnL, nmap, ping, telnet, ...

# Appending rules : Let’s add some more IPv6 rules to our firewall.

sudo ip6tables -A INPUT -p tcp --dport "$SSHPORT" -s HOST\_IPV6\_IP -j ACCEPT

sudo ip6tables -A INPUT -p tcp --dport 80 -j ACCEPT

sudo ip6tables -A INPUT -p tcp --dport 21 -j ACCEPT

sudo ip6tables -A INPUT -p tcp --dport 25 -j ACCEPT

# To see the IPv6 rules with line numbers, type the following command:

sudo ip6tables -L -n --line-numbers

# Deleting rules

sudo ip6tables -D INPUT -p tcp --dport 21 -j ACCEPT

# Exit gracefully.

#------------------------------------------------------------------------------

exit 0