#!/bin/bash

#Release: v.5

udp\_dir='/root/udp'

source $udp\_dir/module

# ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━

# Local Module

#source ./module/module

# ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━

request\_public\_ip=$(grep -m 1 -oE '^[0-9]{1,3}(\.[0-9]{1,3}){3}$' <<<"$(wget -T 10 -t 1 -4qO- "http://ip1.dynupdate.no-ip.com/" || curl -m 10 -4Ls "http://ip1.dynupdate.no-ip.com/")")

#======= multi-login limiter =====

limiter() {

ltr() {

clear

msg -bar

# Check if limiter is already running and remove from scheduled tasks if so

for i in $(atq | awk '{print $1}'); do

if [[ ! $(at -c $i | grep '/root/udp/limiter.sh') = "" ]]; then

atrm $i

sed -i '/root/udp/limiter.sh/d' /var/spool/cron/crontabs/root

print\_center -verd "Limiter stopped"

enter

return

fi

done

# Prompt user to configure limiter

print\_center -ama "CONFIGURE LIMITER"

msg -bar

print\_center -ama "Block users when they exceed the maximum number of connections"

msg -bar

# Set interval for running limiter

unset option

while [[ -z $option ]]; do

msg -nama "Run limiter every (in minutes): "

read option

if [[ ! $option =~ ^[0-9]+$ ]]; then

del 1

print\_center -verm2 "Only numbers are allowed"

sleep 2

del 1

unset option && continue

elif [[ $option -le 0 ]]; then

del 1

print\_center -verm2 "Minimum time is 1 minute"

sleep 2

del 1

unset option && continue

fi

del 1

echo -e "$(msg -nama "Run limiter every: ") $(msg -verd "$option minutes")"

echo "$option" >${udp\_dir}/limit

done

msg -bar

# Set interval for automatically unblocking users blocked by the limiter

print\_center -ama "Users blocked by the limiter will be unlocked automatically (enter 0 for manual unlock)"

msg -bar

unset option

while [[ -z $option ]]; do

msg -nama "Unblock users every (in minutes): "

read option

if [[ ! $option =~ ^[0-9]+$ ]]; then

tput cuu1 && tput dl1

print\_center -verm2 "Only numbers are allowed"

sleep 2

tput cuu1 && tput dl1

unset option && continue

fi

tput cuu1 && tput dl1

[[ $option -le 0 ]] && echo -e "$(msg -nama "Unblock: ") $(msg -verd "manual")" || echo -e "$(msg -nama "Unblock users every: ") $(msg -verd "$option minutes")"

echo "$option" >${udp\_dir}/unlimit

done

# Start limiter with new settings

nohup ${udp\_dir}/limiter.sh &>/dev/null &

msg -bar

print\_center -verd "Limiter started"

enter

}

l\_exp() {

clear

msg -bar

l\_cron=$(cat /var/spool/cron/crontabs/root | grep -w 'limiter.sh' | grep -w 'ssh')

if [[ -z "$l\_cron" ]]; then

print\_center -ama '0 1 \* \* \* /root/udp/limiter.sh --ssh' >>/var/spool/cron/crontabs/root

print\_center -verd "${a82:-scheduled expiration limiter\nwill be executed every day at 1 am\naccording to the time scheduled on the server}"

else

sed -i '/limiter.sh --ssh/d' /var/spool/cron/crontabs/root

print\_center -verm2 "${a83:-expired limiter stopped}"

fi

enter

return

}

log() {

clear

msg -bar

print\_center -ama "${a84:-LIMITER REGISTRY}"

msg -bar

[[ ! -e /root/udp/limit.log ]] && touch /root/udp/limit.log

if [[ -z $(cat /root/udp/limit.log) ]]; then

print\_center -ama "${a85:-no account limiter record}"

msg -bar

sleep 2

return

fi

msg -teal "$(cat /root/udp/limit.log)"

msg -bar

print\_center -ama "►► ${a86:-Press enter to continue...} ◄◄"

print\_center -ama "►► ${a87:-0 to clear record} ◄◄"

read option

[[ $option = "0" ]] && print\_center -ama "" >/root/udp/limit.log

}

[[ $(cat /var/spool/cron/crontabs/root | grep -w 'limiter.sh' | grep -w 'ssh') ]] && lim\_e=$(msg -verd "[ON]") || lim\_e=$(msg -verm2 "[OFF]")

clear

msg -bar

print\_center -ama "${a11:-ACCOUNT LIMITER}"

msg -bar

menu\_func "${a64:-MULTI-LOGIN LIMITER}" "${a65:-EXPIRED LIMITER} $lim\_e" "${a66:-LIMITER LOG}"

back

msg -ne " ${a67:-option}: "

read option

case $option in

1) ltr ;;

2) l\_exp ;;

3) log ;;

0) return ;;

esac

}

#======= CONFIGURATION OF SSH UDP ACCOUNTS =======

data\_user() {

cat\_users=$(cat "/etc/passwd" | grep 'home' | grep 'false' | grep -v 'syslog' | grep -v '::/' | grep -v 'hwid\|token')

[[ -z "$(echo "${cat\_users}" | head -1)" ]] && print\_center -verm2 "${a96:-NO REGISTERED SSH USERS}" && return 1

dat\_us=$(printf '%-13s%-14s%-10s%-4s%-6s%s' "${a48:-User}" "${a49:-Pass}" "${a97:-Date}" "${a98:-Days}" 'Lmt' 'Stat')

msg -azu " $dat\_us"

msg -bar

i=1

while read line; do

u=$(echo "$line" | awk -F ':' '{print $1}')

fecha=$(chage -l "$u" | sed -n '4p' | awk -F ': ' '{print $2}')

mes\_dia=$(echo $fecha | awk -F ',' '{print $1}' | sed 's/ //g')

ano=$(echo $fecha | awk -F ', ' '{printf $2}' | cut -c 3-)

us=$(printf '%-12s' "$u")

pass=$(echo "$line" | awk -F ':' '{print $5}' | cut -d ',' -f2)

[[ "${#pass}" -gt '12' ]] && pass="${a99:-Discount}"

pass="$(printf '%-12s' "$pass")"

unset stat

if [[ $(passwd --status $u | cut -d ' ' -f2) = "P" ]]; then

stat="$(msg -verd "ULK")"

else

stat="$(msg -verm2 "BLK")"

fi

Limit=$(echo "$line" | awk -F ':' '{print $5}' | cut -d ',' -f1)

[[ "${#Limit}" = "1" ]] && Limit=$(printf '%2s%-4s' "$Limit") || Limit=$(printf '%-6s' "$Limit")

echo -ne "$(msg -verd "$i")$(msg -verm2 "-")$(msg -azu "${us}") $(msg -azu "${pass}")"

if [[ $(echo $fecha | awk '{print $2}') = "" ]]; then

exp="$(printf '%8s%-2s' '[X]')"

exp+="$(printf '%-6s' '[X]')"

echo " $(msg -verm2 "$fecha")$(msg -verd "$exp")$(echo -e "$stat")"

else

if [[ $(date +%s) -gt $(date '+%s' -d "${fecha}") ]]; then

exp="$(printf '%-5s' "Exp")"

echo " $(msg -verm2 "$mes\_dia/$ano") $(msg -verm2 "$exp")$(msg -ama "$Limit")$(echo -e "$stat")"

else

EXPTIME="$(($(($(date '+%s' -d "${fecha}") - $(date +%s))) / 86400))"

[[ "${#EXPTIME}" = "1" ]] && exp="$(printf '%2s%-3s' "$EXPTIME")" || exp="$(printf '%-5s' "$EXPTIME")"

echo " $(msg -verm2 "$mes\_dia/$ano") $(msg -verd "$exp")$(msg -ama "$Limit")$(echo -e "$stat")"

fi

fi

let i++

done <<<"$cat\_users"

}

# ======== user Details ====

detail\_user() {

clear

active\_users=('' $(show\_users))

if [[ -z ${active\_users[@]} ]]; then

msg -bar

print\_center -verm2 "${a62:-No registered user}"

msg -bar

sleep 3

return

else

msg -bar

print\_center -ama "${a63:-DETAILS OF USERS}"

msg -bar

fi

data\_user

msg -bar

enter

}

#======== user block ======

block\_user() {

clear

active\_users=('' $(show\_users))

msg -bar

print\_center -ama "${a9:-BLOCK/UNBLOCK USERS}"

msg -bar

data\_user

back

print\_center -ama "${a52:-Type a Username from the list}"

msg -bar

unset selection

while [[ ${selection} = "" ]]; do

msg -nazu "${a53:-Please type a username}: " && read selection

del 1

done

[[ ${selection} = "0" ]] && return

if [[ ! $(print\_center -ama "${selection}" | egrep '[^0-9]') ]]; then

user\_del="${active\_users[$selection]}"

else

user\_del="$selection"

fi

[[ -z $user\_del ]] && {

msg -verm "${a54:-Error, Invalid User}"

msg -bar

return 1

}

[[ ! $(echo ${active\_users[@]} | grep -w "$user\_del") ]] && {

msg -verm "${a54:-Error, Invalid User}"

msg -bar

return 1

}

msg -nama " ${a48:-Username}: $user\_del >>>> "

if [[ $(passwd --status $user\_del | cut -d ' ' -f2) = "P" ]]; then

pkill -u $user\_del &>/dev/null

droplim=$(droppids | grep -w "$user\_del" | awk '{print $2}')

kill -9 $droplim &>/dev/null

usermod -L $user\_del &>/dev/null

sleep 2

msg -verm2 "${a60:-Blocked}"

else

usermod -U $user\_del

sleep 2

msg -verd "${a61:-Unblocked}"

fi

msg -bar

sleep 3

}

#======== user remover =========

renew\_user\_fun() {

#nome dias

datexp=$(date "+%F" -d " + $2 days") && valid=$(date '+%C%y-%m-%d' -d " + $2 days")

if chage -E $valid $1; then

print\_center -ama "${a100:-Renewed User Successfully}"

else

print\_center -verm "${a101:-Error, Renewal failed!}"

fi

}

renew\_user() {

clear

active\_users=('' $(show\_users))

msg -bar

print\_center -ama "${a8:-RENEW USERS}"

msg -bar

data\_user

back

print\_center -ama "${a52:-Type a Username from the list}"

msg -bar

unset selection

while [[ -z ${selection} ]]; do

msg -nazu "${a53:-Select an Option}: " && read selection

del 1

done

[[ ${selection} = "0" ]] && return

if [[ ! $(print\_center -ama "${selection}" | egrep '[^0-9]') ]]; then

useredit="${active\_users[$selection]}"

else

useredit="$selection"

fi

[[ -z $useredit ]] && {

msg -verm "${a54:-Error, Invalid User}"

msg -bar

sleep 3

return 1

}

[[ ! $(print\_center -ama ${active\_users[@]} | grep -w "$useredit") ]] && {

msg -verm "${a54:-Error, Invalid User}"

msg -bar

sleep 3

return 1

}

while true; do

msg -ne "${a58:-New Duration}: $useredit"

read -p ": " userdays

if [[ -z "$userdays" ]]; then

print\_center -ama -e '\n\n\n'

err\_fun 7 && continue

elif [[ "$userdays" != +([0-9]) ]]; then

print\_center -ama -e '\n\n\n'

err\_fun 8 && continue

elif [[ "$userdays" -gt "360" ]]; then

print\_center -ama -e '\n\n\n'

err\_fun 9 && continue

fi

break

done

msg -bar

renew\_user\_fun "${useredit}" "${userdays}"

msg -bar

sleep 3

}

#======== remove client =========

droppids() {

port\_dropbear=$(ps aux | grep 'dropbear' | awk NR==1 | awk '{print $17;}')

log=/var/log/auth.log

loginsukses='Password auth succeeded'

pids=$(ps ax | grep 'dropbear' | grep " $port\_dropbear" | awk -F " " '{print $1}')

for pid in $pids; do

pidlogs=$(grep $pid $log | grep "$loginsukses" | awk -F" " '{print $3}')

i=0

for pidend in $pidlogs; do

let i=i+1

done

if [ $pidend ]; then

login=$(grep $pid $log | grep "$pidend" | grep "$loginsukses")

PID=$pid

user=$(print\_center -ama $login | awk -F" " '{print $10}' | sed -r "s/'/ /g")

waktu=$(print\_center -ama $login | awk -F" " '{print $2"-"$1,$3}')

while [ ${#waktu} -lt 13 ]; do

waktu=$waktu" "

done

while [ ${#user} -lt 16 ]; do

user=$user" "

done

while [ ${#PID} -lt 8 ]; do

PID=$PID" "

done

print\_center -ama "$user $PID $waktu"

fi

done

}

rm\_user() {

pkill -u $1

droplim=$(droppids | grep -w "$1" | awk '{print $2}')

kill -9 $droplim &>/dev/null

userdel --force "$1" &>/dev/null

msj=$?

}

show\_users() {

for u in $(cat /etc/passwd | grep 'home' | grep 'false' | grep -v 'syslog' | grep -v 'hwid' | grep -v 'token' | grep -v '::/' | awk -F ':' '{print $1}'); do

print\_center -ama "$u"

done

}

remove\_user() {

clear

active\_users=('' $(show\_users))

msg -bar

print\_center -ama "${a7:-REMOVE USERS}"

msg -bar

data\_user

back

print\_center -ama "${a52:-Type or Select a User}"

msg -bar

unset selection

while [[ -z ${selection} ]]; do

msg -nazu "${a53:-Please type a username}: " && read selection

tput cuu1 && tput dl1

done

[[ ${selection} = "0" ]] && return

if [[ ! $(print\_center -ama "${selection}" | egrep '[^0-9]') ]]; then

user\_del="${active\_users[$selection]}"

else

user\_del="$selection"

fi

[[ -z $user\_del ]] && {

msg -verm "${a54:-Error, Invalid User}"

msg -bar

return 1

}

[[ ! $(echo ${active\_users[@]} | grep -w "$user\_del") ]] && {

msg -verm "${a54:-Error, Invalid User}"

msg -bar

return 1

}

print\_center -ama "${a55:-Selected User}: $user\_del"

rm\_user "$user\_del"

if [[ $msj = 0 ]]; then

print\_center -verd "[${a56:-Removed}]"

else

print\_center -verm "[${a57:-Not Removed}]"

fi

enter

}

#========create client =============

add\_user() {

Fecha=$(date +%d-%m-%y-%R)

[[ $(cat /etc/passwd | grep $1: | grep -vi [a-z]$1 | grep -v [0-9]$1 >/dev/null) ]] && return 1

valid=$(date '+%C%y-%m-%d' -d " +$3 days")

osl\_v=$(openssl version | awk '{print $2}')

osl\_v=${osl\_v:0:5}

if [[ $osl\_v = '1.1.1' ]]; then

pass=$(openssl passwd -6 $2)

else

pass=$(openssl passwd -1 $2)

fi

useradd -M -s /bin/false -e ${valid} -K PASS\_MAX\_DAYS=$3 -p ${pass} -c $4,$2 $1 &>/dev/null

msj=$?

}

new\_user() {

clear

active\_users=('' $(show\_users))

msg -bar

print\_center -ama "${a6:-CREATE USER ACCOUNT}"

msg -bar

data\_user

back

while true; do

msg -ne " ${a41:-Username}: "

read nameuser

nameuser="$(echo $nameuser | sed 'y/áÁàÀãÃâÂéÉêÊíÍóÓõÕôÔúÚñÑçÇªº/aAaAaAaAeEeEiIoOoOoOuUnNcCao/')"

nameuser="$(echo $nameuser | sed -e 's/[^a-z0-9 -]//ig')"

if [[ -z $nameuser ]]; then

err\_fun 1 && continue

elif [[ "${nameuser}" = "0" ]]; then

return

elif [[ "${#nameuser}" -lt "4" ]]; then

err\_fun 2 && continue

elif [[ "${#nameuser}" -gt "12" ]]; then

err\_fun 3 && continue

elif [[ "$(echo ${active\_users[@]} | grep -w "$nameuser")" ]]; then

err\_fun 14 && continue

fi

break

done

while true; do

msg -ne " ${a42:-Password}"

read -p ": " userpass

userpass="$(echo $userpass | sed 'y/áÁàÀãÃâÂéÉêÊíÍóÓõÕôÔúÚñÑçÇªº/aAaAaAaAeEeEiIoOoOoOuUnNcCao/')"

if [[ -z $userpass ]]; then

err\_fun 4 && continue

elif [[ "${#userpass}" -lt "4" ]]; then

err\_fun 5 && continue

elif [[ "${#userpass}" -gt "12" ]]; then

err\_fun 6 && continue

fi

break

done

while true; do

msg -ne " ${a43:-Number of Days}"

read -p ": " userdays

if [[ -z "$userdays" ]]; then

err\_fun 7 && continue

elif [[ "$userdays" != +([0-9]) ]]; then

err\_fun 8 && continue

elif [[ "$userdays" -gt "360" ]]; then

err\_fun 9 && continue

fi

break

done

while true; do

msg -ne " ${a44:-Connection Limit}"

read -p ": " limiteuser

if [[ -z "$limiteuser" ]]; then

err\_fun 11 && continue

elif [[ "$limiteuser" != +([0-9]) ]]; then

err\_fun 12 && continue

elif [[ "$limiteuser" -gt "999" ]]; then

err\_fun 13 && continue

fi

break

done

add\_user "${nameuser}" "${userpass}" "${userdays}" "${limiteuser}"

clear

msg -bar

if [[ $msj = 0 ]]; then

print\_center -verd "${a45:-User Created Successfully}"

else

print\_center -verm2 "${a46:-Error, user not created}"

enter

return 1

fi

msg -bar

msg -ne " ${a47:-Server IP}: " && msg -ama " $request\_public\_ip"

msg -ne " ${a47:-Port Range}: " && msg -ama " 1-65535"

msg -ne " ${a48:-Username}: " && msg -ama " $nameuser"

msg -ne " ${a49:-Password}: " && msg -ama " $userpass"

msg -ne " ${a50:-Number of Days}: " && msg -ama " $userdays"

msg -ne " ${a44:-Connection Limit}: " && msg -ama " $limiteuser"

msg -ne " ${a51:-Expiration Date}: " && msg -ama "$(date "+%F" -d " + $userdays days")"

enter

}

Uninstall\_UDP\_CUSTOM() {

title "${a32:-Uninstall UDP CUSTOM}"

read -rp " $(msg -ama "${a33:-Do you wish to continue? [Y/n]}:") " -e -i Y UNINS

[[ $UNINS != @(Y|y|S|s) ]] && return

systemctl stop custom-server.service &>/dev/null

systemctl disable custom-server.service &>/dev/null

rm -rf /etc/systemd/system/custom-server.service

rm -rf /root/udp

rm -rf /usr/bin/udp

systemctl stop udpgw.service &>/dev/null

rm -rf /etc/systemd/system/udpgw.service

rm -rf /usr/bin/udpgw

del 1

print\_center -ama "${a34:-Uninstallation completed!}"

enter

exit

}

reset\_udp\_custom() {

if [[ $(systemctl is-active custom-server.service) = 'active' ]]; then

systemctl stop custom-server.service &>/dev/null

systemctl disable custom-server.service &>/dev/null

print\_center -ama "${a35:-UDP CUSTOM Reset Done!}"

else

systemctl start custom-server.service &>/dev/null

if [[ $(systemctl is-active custom-server.service) = 'active' ]]; then

systemctl enable custom-server.service &>/dev/null

systemctl start custom-server.service &>/dev/null

print\_center -verd "${a36:-UDP CUSTOM Running!}"

else

print\_center -verm2 "${a37:-failed to start request-server.service!}"

fi

fi

enter

}

exclude() {

title "${a20:-Exclude UDP ports}"

print\_center -ama "${a21:-UDP CUSTOM covers full range of ports,}"

print\_center -ama "${a22:-However, you can exclude UDP ports.}"

msg -bar3

print\_center -ama "${a23:-Examples of ports you can exclude:}:"

print\_center -ama "dnstt (slowdns) udp 53 5300"

print\_center -ama "wireguard udp 51820"

print\_center -ama "openvpn udp 1194"

msg -bar

print\_center -verd "${a24:-enter the ports separated by spaces}"

print\_center -verd "${a25:-Example}: 53 5300 51820 1194"

msg -bar3

in\_opcion\_down "${a26:-type ports or hit enter to skip}"

del 2

tmport=($opcion)

for ((i = 0; i < ${#tmport[@]}; i++)); do

num=$((${tmport[$i]}))

if [[ $num -gt 0 ]]; then

echo "$(msg -ama " ${a27:-port to exclude} >") $(msg -azu "$num") $(msg -verd "OK")"

Port+=" $num"

else

msg -verm2 " ${a28:-not a port} > ${tmport[$i]}?"

continue

fi

done

if [[ -z $Port ]]; then

unset Port

print\_center -ama "${a29:-no ports excluded}"

else

Port=" -exclude=$(echo "$Port" | sed "s/ /,/g" | sed 's/,//')"

fi

msg -bar3

}

add\_exclude() {

title "${a20:-Exclude UDP ports}"

print\_center -ama "${a21:-UDP CUSTOM covers full range of ports,}"

print\_center -ama "${a22:-However, you can exclude UDP ports.}"

msg -bar3

print\_center -ama "${a23:-Examples of ports you can exclude:}:"

print\_center -ama "dnstt (slowdns) udp 53 5300"

print\_center -ama "wireguard udp 51820"

print\_center -ama "openvpn udp 1194"

msg -bar

print\_center -verd "${a24:-enter the ports separated by spaces}"

print\_center -verd "${a25:-Example}: 53 5300 51820 1194"

in\_opcion\_down "${a26:-type ports or hit enter to skip}"

del 4

tmport=($opcion)

unset Port

for ((i = 0; i < ${#tmport[@]}; i++)); do

num=$((${tmport[$i]}))

if [[ $num -gt 0 ]]; then

echo "$(msg -ama " ${a27:-port to exclude} >") $(msg -azu "$num") $(msg -verd "OK")"

Port+=" $num"

else

msg -verm2 " ${a28:-not a port} > ${tmport[$i]}?"

continue

fi

done

if [[ $Port = "" ]]; then

unset Port

print\_center -ama "${a29:-no ports excluded}"

else

exclude=$(cat /etc/systemd/system/custom-server.service | grep 'exclude')

if systemctl is-active custom-server.service &>/dev/null; then

systemctl stop custom-server.service &>/dev/null

systemctl disable custom-server.service &>/dev/null

iniciar=1

fi

if [[ -z $exclude ]]; then

Port=" -exclude=$(echo "$Port" | sed "s/ /,/g" | sed 's/,//')"

sed -i "s/ -mode/$Port -mode/" /etc/systemd/system/custom-server.service

else

exclude\_port=$(echo $exclude | awk '{print $4}' | cut -d '=' -f2)

Port="-exclude=$exclude\_port$(echo "$Port" | sed "s/ /,/g")"

sed -i "s/-exclude=$exclude\_port/$Port/" /etc/systemd/system/custom-server.service

fi

if [[ $iniciar = 1 ]]; then

systemctl enable custom-server.service &>/dev/null

systemctl start custom-server.service &>/dev/null

fi

fi

enter

}

quit\_exclude() {

title "${a88:-REMOVE EXCLUSION PORT}"

exclude=$(cat /etc/systemd/system/custom-server.service | grep 'exclude' | awk '{print $4}')

ports=($port)

for ((i = 0; i < ${#ports[@]}; i++)); do

a=$(($i + 1))

echo " $(msg -verd "[$a]") $(msg -verm2 '>') $(msg -azu "${ports[$i]}")"

done

if [[ ! ${#ports[@]} = 1 ]]; then

let a++

msg -bar

echo " $(msg -verd "[0]") $(msg -verm2 ">") $(msg -bra "\033[1;41m${a89:-GO BACK}") $(msg -verd "[$a]") $(msg -verm2 "> ${a90:-REMOVE ALL}")"

msg -bar

else

msg -bar

echo " $(msg -verd "[0]") $(msg -verm2 ">") $(msg -bra "\033[1;41m${a89:-GO BACK}")"

msg -bar

fi

option=$(selection\_fun $a)

[[ $option = 0 ]] && return

if systemctl is-active custom-server.service &>/dev/null; then

systemctl stop custom-server.service &>/dev/null

systemctl disable custom-server.service &>/dev/null

iniciar=1

fi

if [[ $option = $a ]]; then

sed -i "s/$exclude /" /etc/systemd/system/custom-server.service

print\_center -ama "${a91:-Removed all excluded ports}"

else

let option--

unset Port

for ((i = 0; i < ${#ports[@]}; i++)); do

[[ $i = $option ]] && continue

echo "$(msg -ama " ${a27:-Port to exclude} >") $(msg -azu "${ports[$i]}") $(msg -verd "OK")"

Port+=" ${ports[$i]}"

done

Port=$(echo $Port | sed 's/ /,/g')

sed -i "s/$exclude/-exclude=$Port/" /etc/systemd/system/custom-server.service

fi

if [[ $iniciar = 1 ]]; then

systemctl enable custom-server.service &>/dev/null

systemctl start custom-server.service &>/dev/null

fi

enter

}

vps\_info() {

# ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━

#information

# ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━

if [[ "$(grep -c "Ubuntu" /etc/issue.net)" = "1" ]]; then

system=$(cut -d' ' -f1 /etc/issue.net)

system+=$(echo ' ')

system+=$(cut -d' ' -f2 /etc/issue.net | awk -F "." '{print $1}')

else

system=$(cut -d' ' -f1 /etc/issue.net)

fi

\_system=$(printf '%-14s' "$system")

os\_arch=$(uname -i)

ip=$(wget -qO- ipinfo.io/ip)

region=$(wget -qO- ipinfo.io/region)

isp=$(wget -qO- ipinfo.io/org)

timezone=$(wget -qO- ipinfo.io/timezone)

ossys=$(neofetch | grep "OS" | cut -d: -f2 | sed 's/ //g')

host=$(neofetch | grep "Host" | cut -d: -f2 | sed 's/ //g')

kernel=$(neofetch | grep "Kernel" | cut -d: -f2 | sed 's/ //g')

uptime=$(neofetch | grep "Uptime" | cut -d: -f2 | sed 's/ //g')

cpu=$(top -bn1 | awk '/Cpu/ { cpu = "" 100 - $8 "%" }; END { print cpu }')

\_core=$(printf '%-1s' "$(grep -c cpu[0-9] /proc/stat)")

memory=$(neofetch | grep "Memory" | cut -d: -f2 | sed 's/ //g')

ram=$(free -m | awk 'NR==2{printf "%.2f%%", $3\*100/$2 }')

clear

title\_sub "\033[3;40m${a10:-•UDP CUSTOM} Resleeved Net"

print\_center -ama ' Release: v.5'

print\_center -ama ' VPS Info'

msg -bar

echo " "

echo -e "$CLAY ⇢ Operating System :$NC $system $NC"

echo -e "$CLAY ⇢ Arch :$NC $os\_arch $NC"

echo -e "$CLAY ⇢ IP Address :$NC $ip $NC"

echo -e "$CLAY ⇢ Region :$NC $region $NC"

echo -e "$CLAY ⇢ ISP :$NC $isp $NC"

echo -e "$CLAY ⇢ Date :$NC $(date +%A) $(date +%m-%d-%Y)"

echo -e "$CLAY ⇢ Up Time :$NC $uptime $NC"

echo -e "$CLAY ⇢ CPU Load :$NC $cpu $NC"

echo -e "$CLAY ⇢ CPU Cores :$NC $\_core $NC"

echo -e "$CLAY ⇢ Memory Usage :$NC $memory $NC"

echo -e "$CLAY ⇢ Ram Usage :$NC $ram $NC"

msg -bar0

enter

}

# [UDP - CUSTOM]

Menu\_UDP\_CUSTOM() {

title\_sub "\033[3;40m${a10:-• UDP CUSTOM} •"

print\_center -ama 'Release: v.5'

print\_center -ama 'Resleeved Net'

print\_center -ama 'Tunnel App : HTTP CUSTOM'

msg -bar

ram=$(printf '%-8s' "$(free -m | awk 'NR==2{printf "%.2f%%", $3\*100/$2 }')")

cpu=$(printf '%-1s' "$(top -bn1 | awk '/Cpu/ { cpu = "" 100 - $8 "%" }; END { print cpu }')")

echo " $(msg -verd ' ⇢ IP:') $(msg -azu "$request\_public\_ip") $(msg -verd 'Ram:') $(msg -azu "$ram") $(msg -verd 'CPU:') $(msg -azu "$cpu")"

msg -bar

if [[ $(systemctl is-active custom-server.service) = 'active' ]]; then

state="\e[1m\e[32m[ON]"

else

state="\e[1m\e[31m[OFF]"

fi

print\_center -ama "${a12:-Resleeved Net Menu}"

msg -bar3

echo -e " $(msg -verd "[1]") $(msg -verm2 '>') $(msg -azu "${a2:-Turn On/Off UDP}") $state"

echo " $(msg -verd "[2]") $(msg -verm2 '>') $(msg -verd "${a6:-Create User}")"

echo " $(msg -verd "[3]") $(msg -verm2 '>') $(msg -verm2 "${a7:-Remove User}")"

echo " $(msg -verd "[4]") $(msg -verm2 '>') $(msg -ama "${a8:-Renew User}")"

echo " $(msg -verd "[5]") $(msg -verm2 '>') $(msg -blu "${a9:-Block/Unlock User}")"

echo " $(msg -verd "[6]") $(msg -verm2 '>') $(msg -verm3 "${a10:-User Details}")"

echo " $(msg -verd "[7]") $(msg -verm2 '>') $(msg -teal "${a11:-Limit Accounts}")"

msg -bar3

back

# option=$(selection\_fun $num)

read -p " ⇢ Enter your selection: " option

case $option in

1) reset\_udp\_custom ;;

2) new\_user ;;

3) remove\_user ;;

4) renew\_user ;;

5) block\_user ;;

6) detail\_user ;;

7) limiter ;;

0) menu\_main ;;

esac

}

# [MAIN MENU]A

menu\_main() {

main\_title "\033[3;40m${a10:-• UDP CUSTOM} •"

print\_center -ama 'Release: v.5'

print\_center -ama 'Resleeved Net'

msg -bar

# calculate RAM and CPU usage

ram=$(free -m | awk 'NR==2{printf "%.2f%%", $3\*100/$2 }')

cpu=$(top -bn1 | awk '/Cpu/ { cpu = "" 100 - $8 "%" }; END { print cpu }')

# print system information

echo " $(msg -verd ' ⇢ IP:') $(msg -azu "$request\_public\_ip") $(msg -verd 'Ram:') $(msg -azu "$ram") $(msg -verd 'CPU:') $(msg -azu "$cpu")"

msg -bar0

# print options menu

print\_center -ama "${a12:-Resleeved Net UDP}"

msg -bar3

echo " $(msg -verd "[1]") $(msg -verm2 '>') $(msg -verd "${a6:-UDP CUSTOM}")"

msg -bar3

echo " $(msg -verd "[2]") $(msg -verm2 '>') $(msg -teal "${a10:-VPS Info}")"

msg -bar3

echo " $(msg -verd "[3]") $(msg -verm2 '>') $(msg -verm2 "${a3:-Uninstall UDP CUSTOM}")"

exit2home

# prompt user for option selection

read -p " ⇢ Enter your selection: " option

# handle option selection

case $option in

1)

Menu\_UDP\_CUSTOM

;;

2)

vps\_info

;;

3)

Uninstall\_UDP\_CUSTOM

;;

0)

exit

;;

esac

}

while [[ $? -eq 0 ]]; do

menu\_main

done