Road Warrior – OpenVPN Setup for IRLP

Contents

Setup the Server

1) Get a cheap VM in the cloud. (like Virmach)
   VirMach has a $10 dollar a year server if paid annually.
   It has 250 GB of bandwidth, 10 GB SSD drive, and 192 MB of memory.
2) Order with Ubuntu 14.04 x86 (one of the most reliable Linux operating systems)
3) After ordering make sure you can log into server with Putty or via the VNC console provided.
4) Make sure you record your information (Hostname, IP address, root password and your Hosting service account information – username, password)
5) At your control panel on hosting site, turn on TUN/TAP.
6) Log into the server using root.
7) Update your OS
   a. Apt-get update

Install OpenVPN

8) Download OpenVPN install Script on your new server, as root.
   a. wget https://git.io/vpn -O openvpn-install.sh
9) Make the install script executable but only for the current user.
   a. chmod u+x ./openvpn-install.sh
10) Run the install script
   a. ./openvpn-install.sh
11) Answer the setup questions for setting up OpenVPN on the server.

    You can accept the defaults. If successful you should see:

    Your client configuration is available at: /root/client.ovpn
    If you want to add more clients at a later date, you simply need to run this script again!

12) Start the OpenVPN server (service)
   a. /etc/init.d/openvpn start

Firewall to allow IRLP

13) Download the IRLP firewall rules.
    a. wget http://kd3su.crabdance.com/firewall
14) Copy the firewall file to the init.d dir & make it executable:
    a. cp ./firewall /etc/init.d/firewall
    b. cd /etc/init.d
    c. chmod +x firewall
15) Update the firewall rules
    a. update-rc.d -f firewall defaults
16) Start the firewall
    a. Service start firewall

Install OpenVPN client on your IRLP node.

This assumes you are using Debian, Centos, or Ubuntu operating system and not a Nano Node or something similar.

1) Copy your client.ovpn from virtual server to your node/client via FTP or WinSCP
2) Install OpenVPN on your node/client.
    a. Apt-get install openvpn
3) Start OpenVPN on node/client
    a. openvpn --config client.ovpn

At this point you should have a fully functional VPN server and client!

** Note: see below how to install the OpenVPN client on your Nano Node **

Install OpenVPN client on Nano Node.

1) Log onto your windows PC or laptop.
2) Using WinSCP copy your client.ovpn file from your OpenVPN server to your windows desktop. (if you don't have the application WinSCP, you can download it here.)
3) Once on your desktop, rename the file to: client.conf
   The reason for this is that there is already a VPN client on the Nano Node, and it expects to see a file called "client.conf"
4) Open client.conf with "notepad" as to validate it's a readable file. If the file is corrupt, it will look like all ASCI text. (nothing but gibberish you can't read)
5) Open a browser and login into your Nano Node admin page. http://your_nano_node_IP/admin (Nano node will have to be plugged into the same router as your Windows PC/laptop and connected via ethernet cable.)
6) Once you are connected to the admin page: follow the Nano Node manual for setting up your VPN. Manual can be found here. Look for "Set VPN" in the manual.

Generate a second client on OpenVPN server

1) Re-Run the install script

   a. ./openvpn-install.sh
   b. A menu will appear, select: " 1) Add a new user"
   c. Follow the prompts, until you are prompted for the client name. Add a new unique client name, i.e. not client, as you already have a client called "client".

OpenVPN install script and its contents. In case you need to recreate the script.

```
#!/bin/bash
#
# https://github.com/Nyr/openvpn-install
#
# Copyright (c) 2013 Nyr. Released under the MIT License.



# Detect Debian users running the script with "sh" instead of bash
if readlink /proc/$$/exe | grep -q "dash"; then
	echo "This script needs to be run with bash, not sh"
	exit
fi

if [[ "$EUID" -ne 0 ]]; then
	echo "Sorry, you need to run this as root"
	exit
fi

if [[ ! -e /dev/net/tun ]]; then
	echo "The TUN device is not available
You need to enable TUN before running this script"
	exit
fi

if [[ -e /etc/debian_version ]]; then
	OS=debian
	GROUPNAME=nogroup
	RCLOCAL='/etc/rc.local'
elif [[ -e /etc/centos-release || -e /etc/redhat-release ]]; then
	OS=centos
	GROUPNAME=nobody
	RCLOCAL='/etc/rc.d/rc.local'
```

```
else
	echo "Looks like you aren't running this installer on Debian, Ubuntu or CentOS"
	exit
fi

newclient () {
	# Generates the custom client.ovpn
	cp /etc/openvpn/client-common.txt ~/$1.ovpn
	echo "<ca>" >> ~/$1.ovpn
	cat /etc/openvpn/easy-rsa/pki/ca.crt >> ~/$1.ovpn
	echo "</ca>" >> ~/$1.ovpn
	echo "<cert>" >> ~/$1.ovpn
	cat /etc/openvpn/easy-rsa/pki/issued/$1.crt >> ~/$1.ovpn
	echo "</cert>" >> ~/$1.ovpn
	echo "<key>" >> ~/$1.ovpn
	cat /etc/openvpn/easy-rsa/pki/private/$1.key >> ~/$1.ovpn
	echo "</key>" >> ~/$1.ovpn
	echo "<tls-auth>" >> ~/$1.ovpn
	cat /etc/openvpn/ta.key >> ~/$1.ovpn
	echo "</tls-auth>" >> ~/$1.ovpn
}

if [[ -e /etc/openvpn/server.conf ]]; then
	while :
	do
	clear
		echo "Looks like OpenVPN is already installed."
		echo
		echo "What do you want to do?"
		echo "   1) Add a new user"
		echo "   2) Revoke an existing user"
		echo "   3) Remove OpenVPN"
		echo "   4) Exit"
		read -p "Select an option [1-4]: " option
		case $option in
```

```
1)
echo
echo "Tell me a name for the client certificate."
echo "Please, use one word only, no special characters."
read -p "Client name: " -e CLIENT
cd /etc/openvpn/easy-rsa/
EASYRSA_CERT_EXPIRE=3650 ./easyrsa build-client-full $CLIENT nopass
# Generates the custom client.ovpn
newclient "$CLIENT"
echo
echo "Client $CLIENT added, configuration is available at:" ~/"$CLIENT.ovpn"
exit
;;
2)
# This option could be documented a bit better and maybe even be simplified
# ...but what can I say, I want some sleep too
NUMBEROFCLIENTS=$(tail -n +2 /etc/openvpn/easy-rsa/pki/index.txt | grep -c "^V")
if [[ "$NUMBEROFCLIENTS" = '0' ]]; then
        echo
        echo "You have no existing clients!"
        exit
fi
echo
echo "Select the existing client certificate you want to revoke:"
tail -n +2 /etc/openvpn/easy-rsa/pki/index.txt | grep "^V" | cut -d '=' -f 2 | nl -s ') '
if [[ "$NUMBEROFCLIENTS" = '1' ]]; then
        read -p "Select one client [1]: " CLIENTNUMBER
else
        read -p "Select one client [1-$NUMBEROFCLIENTS]: " CLIENTNUMBER
fi
CLIENT=$(tail -n +2 /etc/openvpn/easy-rsa/pki/index.txt | grep "^V" | cut -d '=' -f 2 | sed -n
"$CLIENTNUMBER"p)
echo
read -p "Do you really want to revoke access for client $CLIENT? [y/N]: " -e REVOKE
if [[ "$REVOKE" = 'y' || "$REVOKE" = 'Y' ]]; then
```

```
                cd /etc/openvpn/easy-rsa/
                ./easyrsa --batch revoke $CLIENT
                EASYRSA_CRL_DAYS=3650 ./easyrsa gen-crl
                rm -f pki/reqs/$CLIENT.req
                rm -f pki/private/$CLIENT.key
                rm -f pki/issued/$CLIENT.crt
                rm -f /etc/openvpn/crl.pem
                cp /etc/openvpn/easy-rsa/pki/crl.pem /etc/openvpn/crl.pem
                # CRL is read with each client connection, when OpenVPN is dropped to nobody
                chown nobody:$GROUPNAME /etc/openvpn/crl.pem
                echo
                echo "Certificate for client $CLIENT revoked!"
        else
                echo
                echo "Certificate revocation for client $CLIENT aborted!"
        fi
        exit
        ;;
        3)
        echo
        read -p "Do you really want to remove OpenVPN? [y/N]: " -e REMOVE
        if [[ "$REMOVE" = 'y' || "$REMOVE" = 'Y' ]]; then
                PORT=$(grep '^port ' /etc/openvpn/server.conf | cut -d " " -f 2)
                PROTOCOL=$(grep '^proto ' /etc/openvpn/server.conf | cut -d " " -f 2)
                if pgrep firewalld; then
                        IP=$(firewall-cmd --direct --get-rules ipv4 nat POSTROUTING | grep '\-s 10.8.0.0/24
'"'"'!'"'"' -d 10.8.0.0/24 -j SNAT --to ' | cut -d " " -f 10)
                        # Using both permanent and not permanent rules to avoid a firewalld reload.
                        firewall-cmd --zone=public --remove-port=$PORT/$PROTOCOL
                        firewall-cmd --zone=trusted --remove-source=10.8.0.0/24
                        firewall-cmd --permanent --zone=public --remove-port=$PORT/$PROTOCOL
                        firewall-cmd --permanent --zone=trusted --remove-source=10.8.0.0/24
                        firewall-cmd --direct --remove-rule ipv4 nat POSTROUTING 0 -s 10.8.0.0/24 ! -d
10.8.0.0/24 -j SNAT --to $IP
                        firewall-cmd --permanent --direct --remove-rule ipv4 nat POSTROUTING 0 -s 10.8.0.0/24
! -d 10.8.0.0/24 -j SNAT --to $IP
```

```
                else
                        IP=$(grep 'iptables -t nat -A POSTROUTING -s 10.8.0.0/24 ! -d 10.8.0.0/24 -j SNAT --to '
$RCLOCAL | cut -d " " -f 14)
                        iptables -t nat -D POSTROUTING -s 10.8.0.0/24 ! -d 10.8.0.0/24 -j SNAT --to $IP
                        sed -i '/iptables -t nat -A POSTROUTING -s 10.8.0.0\/24 ! -d 10.8.0.0\/24 -j SNAT --to /d'
$RCLOCAL
                        if iptables -L -n | grep -qE '^ACCEPT'; then
                                iptables -D INPUT -p $PROTOCOL --dport $PORT -j ACCEPT
                                iptables -D FORWARD -s 10.8.0.0/24 -j ACCEPT
                                iptables -D FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
                                sed -i "/iptables -I INPUT -p $PROTOCOL --dport $PORT -j ACCEPT/d"
$RCLOCAL
                                sed -i "/iptables -I FORWARD -s 10.8.0.0\/24 -j ACCEPT/d" $RCLOCAL
                                sed -i "/iptables -I FORWARD -m state --state RELATED,ESTABLISHED -j
ACCEPT/d" $RCLOCAL
                        fi
                fi
                if sestatus 2>/dev/null | grep "Current mode" | grep -q "enforcing" && [[ "$PORT" != '1194' ]]; then
                        semanage port -d -t openvpn_port_t -p $PROTOCOL $PORT
                fi
                if [[ "$OS" = 'debian' ]]; then
                        apt-get remove --purge -y openvpn
                else
                        yum remove openvpn -y
                fi
                rm -rf /etc/openvpn
                rm -f /etc/sysctl.d/30-openvpn-forward.conf
                echo
                echo "OpenVPN removed!"
        else
                echo
                echo "Removal aborted!"
        fi
        exit
        ;;
        4) exit;;
    esac
```

```
        done
else
        clear
        echo 'Welcome to this OpenVPN "road warrior" installer!'
        echo
        # OpenVPN setup and first user creation
        echo "I need to ask you a few questions before starting the setup."
        echo "You can leave the default options and just press enter if you are ok with them."
        echo
        echo "First, provide the IPv4 address of the network interface you want OpenVPN"
        echo "listening to."
        # Autodetect IP address and pre-fill for the user
        IP=$(ip addr | grep 'inet' | grep -v inet6 | grep -vE '127\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}' | grep -oE '[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}' | head -1)
        read -p "IP address: " -e -i $IP IP
        # If $IP is a private IP address, the server must be behind NAT
        if echo "$IP" | grep -qE '^(10\.|172\.1[6789]\.|172\.2[0-9]\.|172\.3[01]\.|192\.168)'; then
                echo
                echo "This server is behind NAT. What is the public IPv4 address or hostname?"
                read -p "Public IP address / hostname: " -e PUBLICIP
        fi
        echo
        echo "Which protocol do you want for OpenVPN connections?"
        echo "   1) UDP (recommended)"
        echo "   2) TCP"
        read -p "Protocol [1-2]: " -e -i 1 PROTOCOL
        case $PROTOCOL in
                1)
                PROTOCOL=udp
                ;;
                2)
                PROTOCOL=tcp
                ;;
        esac
        echo
```

```
echo "What port do you want OpenVPN listening to?"
read -p "Port: " -e -i 1194 PORT
echo
echo "Which DNS do you want to use with the VPN?"
echo "   1) Current system resolvers"
echo "   2) 1.1.1.1"
echo "   3) Google"
echo "   4) OpenDNS"
echo "   5) Verisign"
read -p "DNS [1-5]: " -e -i 1 DNS
echo
echo "Finally, tell me your name for the client certificate."
echo "Please, use one word only, no special characters."
read -p "Client name: " -e -i client CLIENT
echo
echo "Okay, that was all I needed. We are ready to set up your OpenVPN server now."
read -n1 -r -p "Press any key to continue..."
if [[ "$OS" = 'debian' ]]; then
        apt-get update
        apt-get install openvpn iptables openssl ca-certificates -y
else
        # Else, the distro is CentOS
        yum install epel-release -y
        yum install openvpn iptables openssl ca-certificates -y
fi
# Get easy-rsa
EASYRSAURL='https://github.com/OpenVPN/easy-rsa/releases/download/v3.0.5/EasyRSA-nix-3.0.5.tgz'
wget -O ~/easyrsa.tgz "$EASYRSAURL" 2>/dev/null || curl -Lo ~/easyrsa.tgz "$EASYRSAURL"
tar xzf ~/easyrsa.tgz -C ~/
mv ~/EasyRSA-3.0.5/ /etc/openvpn/
mv /etc/openvpn/EasyRSA-3.0.5/ /etc/openvpn/easy-rsa/
chown -R root:root /etc/openvpn/easy-rsa/
rm -f ~/easyrsa.tgz
cd /etc/openvpn/easy-rsa/
# Create the PKI, set up the CA and the server and client certificates
```

```
./easyrsa init-pki
./easyrsa --batch build-ca nopass
EASYRSA_CERT_EXPIRE=3650 ./easyrsa build-server-full server nopass
EASYRSA_CERT_EXPIRE=3650 ./easyrsa build-client-full $CLIENT nopass
EASYRSA_CRL_DAYS=3650 ./easyrsa gen-crl
# Move the stuff we need
cp pki/ca.crt pki/private/ca.key pki/issued/server.crt pki/private/server.key pki/crl.pem /etc/openvpn
# CRL is read with each client connection, when OpenVPN is dropped to nobody
chown nobody:$GROUPNAME /etc/openvpn/crl.pem
# Generate key for tls-auth
openvpn --genkey --secret /etc/openvpn/ta.key
# Create the DH parameters file using the predefined ffdhe2048 group
echo '-----BEGIN DH PARAMETERS-----
MIIBCAKCAQEA//////////+t+FRYortKmq/cViAnPTzx2LnFg84tNpWp4TZBFGQz
+8yTnc4kmz75fS/jY2MMddj2gbICrsRhetPfHtXV/WVhJDP1H18GbtCFY2VVPe0a
87VXE15/V8k1mE8McODmi3fipona8+/och3xWKE2rec1MKzKT0g6eXq8CrGCsyT7
YdEIqUuyyOP7uWrat2DX9GgdT0Kj3jlN9K5W7edjcrsZCwenyO4KbXCeAvzhzffi
7MA0BM0oNC9hkXL+nOmFg/+OTxIy7vKBg8P+OxtMb61zO7X8vC7CIAXFjvGDfRaD
ssbzSibBsu/6iGtCOGEoXJf//////////wIBAg==
-----END DH PARAMETERS-----' > /etc/openvpn/dh.pem
# Generate server.conf
echo "port $PORT
proto $PROTOCOL
dev tun
sndbuf 0
rcvbuf 0
ca ca.crt
cert server.crt
key server.key
dh dh.pem
auth SHA512
tls-auth ta.key 0
topology subnet
server 10.8.0.0 255.255.255.0
ifconfig-pool-persist ipp.txt" > /etc/openvpn/server.conf
```

```
echo 'push "redirect-gateway def1 bypass-dhcp"' >> /etc/openvpn/server.conf
# DNS
case $DNS in
	1)
	# Locate the proper resolv.conf
	# Needed for systems running systemd-resolved
	if grep -q "127.0.0.53" "/etc/resolv.conf"; then
		RESOLVCONF='/run/systemd/resolve/resolv.conf'
	else
		RESOLVCONF='/etc/resolv.conf'
	fi
	# Obtain the resolvers from resolv.conf and use them for OpenVPN
	grep -v '#' $RESOLVCONF | grep 'nameserver' | grep -E -o '[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}' | while read line;
do
		echo "push \"dhcp-option DNS $line\"" >> /etc/openvpn/server.conf
	done
	;;
	2)
	echo 'push "dhcp-option DNS 1.1.1.1"' >> /etc/openvpn/server.conf
	echo 'push "dhcp-option DNS 1.0.0.1"' >> /etc/openvpn/server.conf
	;;
	3)
	echo 'push "dhcp-option DNS 8.8.8.8"' >> /etc/openvpn/server.conf
	echo 'push "dhcp-option DNS 8.8.4.4"' >> /etc/openvpn/server.conf
	;;
	4)
	echo 'push "dhcp-option DNS 208.67.222.222"' >> /etc/openvpn/server.conf
	echo 'push "dhcp-option DNS 208.67.220.220"' >> /etc/openvpn/server.conf
	;;
	5)
	echo 'push "dhcp-option DNS 64.6.64.6"' >> /etc/openvpn/server.conf
	echo 'push "dhcp-option DNS 64.6.65.6"' >> /etc/openvpn/server.conf
	;;
esac
echo "keepalive 10 120
```

```
cipher AES-256-CBC
user nobody
group $GROUPNAME
persist-key
persist-tun
status openvpn-status.log
verb 3
crl-verify crl.pem" >> /etc/openvpn/server.conf
        # Enable net.ipv4.ip_forward for the system
        echo 'net.ipv4.ip_forward=1' > /etc/sysctl.d/30-openvpn-forward.conf
        # Enable without waiting for a reboot or service restart
        echo 1 > /proc/sys/net/ipv4/ip_forward
        if pgrep firewalld; then
                # Using both permanent and not permanent rules to avoid a firewalld
                # reload.
                # We don't use --add-service=openvpn because that would only work with
                # the default port and protocol.
                firewall-cmd --zone=public --add-port=$PORT/$PROTOCOL
                firewall-cmd --zone=trusted --add-source=10.8.0.0/24
                firewall-cmd --permanent --zone=public --add-port=$PORT/$PROTOCOL
                firewall-cmd --permanent --zone=trusted --add-source=10.8.0.0/24
                # Set NAT for the VPN subnet
                firewall-cmd --direct --add-rule ipv4 nat POSTROUTING 0 -s 10.8.0.0/24 ! -d 10.8.0.0/24 -j SNAT --to $IP
                firewall-cmd --permanent --direct --add-rule ipv4 nat POSTROUTING 0 -s 10.8.0.0/24 ! -d 10.8.0.0/24 -j SNAT --to $IP
        else
                # Needed to use rc.local with some systemd distros
                if [[ "$OS" = 'debian' && ! -e $RCLOCAL ]]; then
                        echo '#!/bin/sh -e
exit 0' > $RCLOCAL
                fi
                chmod +x $RCLOCAL
                # Set NAT for the VPN subnet
                iptables -t nat -A POSTROUTING -s 10.8.0.0/24 ! -d 10.8.0.0/24 -j SNAT --to $IP
                sed -i "1 a\iptables -t nat -A POSTROUTING -s 10.8.0.0/24 ! -d 10.8.0.0/24 -j SNAT --to $IP" $RCLOCAL
                if iptables -L -n | grep -qE '^(REJECT|DROP)'; then
```

```
                        # If iptables has at least one REJECT rule, we asume this is needed.
                        # Not the best approach but I can't think of other and this shouldn't
                        # cause problems.
                        iptables -I INPUT -p $PROTOCOL --dport $PORT -j ACCEPT
                        iptables -I FORWARD -s 10.8.0.0/24 -j ACCEPT
                        iptables -I FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
                        sed -i "1 a\iptables -I INPUT -p $PROTOCOL --dport $PORT -j ACCEPT" $RCLOCAL
                        sed -i "1 a\iptables -I FORWARD -s 10.8.0.0/24 -j ACCEPT" $RCLOCAL
                        sed -i "1 a\iptables -I FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT" $RCLOCAL
                fi
fi
# If SELinux is enabled and a custom port was selected, we need this
if sestatus 2>/dev/null | grep "Current mode" | grep -q "enforcing" && [[ "$PORT" != '1194' ]]; then
                # Install semanage if not already present
                if ! hash semanage 2>/dev/null; then
                        yum install policycoreutils-python -y
                fi
                semanage port -a -t openvpn_port_t -p $PROTOCOL $PORT
fi
# And finally, restart OpenVPN
if [[ "$OS" = 'debian' ]]; then
                # Little hack to check for systemd
                if pgrep systemd-journal; then
                        systemctl restart openvpn@server.service
                else
                        /etc/init.d/openvpn restart
                fi
else
                if pgrep systemd-journal; then
                        systemctl restart openvpn@server.service
                        systemctl enable openvpn@server.service
                else
                        service openvpn restart
                        chkconfig openvpn on
                fi
```

```
		fi
		# If the server is behind a NAT, use the correct IP address
		if [[ "$PUBLICIP" != "" ]]; then
			IP=$PUBLICIP
		fi
		# client-common.txt is created so we have a template to add further users later
		echo "client
dev tun
proto $PROTOCOL
sndbuf 0
rcvbuf 0
remote $IP $PORT
resolv-retry infinite
nobind
persist-key
persist-tun
remote-cert-tls server
auth SHA512
cipher AES-256-CBC
setenv opt block-outside-dns
key-direction 1
verb 3" > /etc/openvpn/client-common.txt
		# Generates the custom client.ovpn
		newclient "$CLIENT"
		echo
		echo "Finished!"
		echo
		echo "Your client configuration is available at:" ~/"$CLIENT.ovpn"
		echo "If you want to add more clients, you simply need to run this script again!"
fi
```

IRLP firewall.txt file contents

(you can copy everything below to a file called: firewall.txt , and then upload it to your server. This would replace step number 13 above. This here is mainly a reference guide for the contents of what is in the firewall.txt file, or should be in it.)

```
#!/bin/sh

### BEGIN INIT INFO
# Provides:         firewall
# Required-Start:  $syslog
# Required-Stop:   $syslog
# Default-Start:   2 3 4 5
# Default-Stop:    0 1 6
# Short-Description: Start Firewall Services
### END INIT INFO

PATH=/sbin:/bin:/usr/sbin:/usr/bin

. /lib/lsb/init-functions

NAME=firewall
IPTABLES=/sbin/iptables

test -x $IPTABLES || exit 5

# Define The External Interface Name
EXTERNAL_INTERFACE="venet+"

# Define The Internal (VPN) Interface Name
INTERNAL_INTERFACE="tun0"

# Define The Loopback Interface Name
LOOPBACK_INTERFACE="lo"

# Define TCP Ports which connections are permitted inbound
TCP_ACCEPT="22 1194 2222"
```

```
#
# vsftpd is set to use "63000:64000"

# Common TCP Ports
# 21/ftp
# 22/ssh
# 23/telnet
# 25/smtp
# 53/dns
# 80/http
# 110/pop3
# 143/imap
# 443/https
# 5666/nrpe (Nagios)

# Define UDP Ports which connections are permitted inbound
UDP_ACCEPT="53 1194"

# Common UDP Ports
# 53/dns
# 161/snmp

# Define ICMP Types/codes which are permitted inbound
ICMP_ACCEPT="3/0 3/4 3/10 11/0 11/1"
#ICMP_ACCEPT=""

# Common ICMP Types
# 3/0 = Port Unreachable
# 3/4 = Fragmentation needed
# 3/10 = Communication Administratively Prohibited
# 8/0 = Echo Request
# 11/0 = Time to Live exceeded in transit
# 11/1 = Fragment Reassembly Time Exceeded

# Broadcast Types to Block (broadcast, multicast, unicast)
#BCAST_BLOCK="broadcast multicast"
# Does not work on OpenVZ?
```

```
BCAST_BLOCK=""

# Trusted sources which will have NO filtering applied.
# Please add here with caution!
TRUSTED_SOURCE=""

#NRPE="0/0"

# Service specific ACLs
SSH_CLIENTS="0/0"
#RSYNC_CLIENTS="166.84.0.28"

IRLP=10.8.0.2
IRLPTCP="222 15425:15428"
IRLPUDP="2074:2094"

if [ -r /etc/default/$NAME ]; then
        . /etc/default/$NAME
fi

case $1 in
        start)
                log_daemon_msg "Starting Firewall Services"
                #

                # First accept anything that is already in the state table
                $IPTABLES -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
                $IPTABLES -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

                # Set Default Policies to drop
                $IPTABLES -P INPUT DROP
                $IPTABLES -P OUTPUT DROP
                $IPTABLES -P FORWARD DROP
                #

                # Allow everything on loopback
                $IPTABLES -A INPUT -i $LOOPBACK_INTERFACE -j ACCEPT
```

```
		$IPTABLES -A OUTPUT -o $LOOPBACK_INTERFACE -j ACCEPT


		# Allow listed TCP services in

		if [ "${TCP_ACCEPT}" != "" ]

		then

		for tcp in ${TCP_ACCEPT}

		do

		$IPTABLES -A INPUT -i $EXTERNAL_INTERFACE -p tcp -s 0/0 --sport 1024:65535 -d 0/0 --dport
$tcp -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT

		$IPTABLES -A OUTPUT -o $EXTERNAL_INTERFACE -p tcp -s 0/0 --sport $tcp -d 0/0 --dport
1024:65535 -m state --state ESTABLISHED,RELATED -j ACCEPT

		done

		fi


	# Allow listed UDP services in

	if [ "${UDP_ACCEPT}" != "" ]

	then

	for udp in ${UDP_ACCEPT}

	do

	$IPTABLES -A INPUT -i $EXTERNAL_INTERFACE -p udp -s 0/0 --sport 1024:65535 -d 0/0 --dport $udp -m
state --state NEW,ESTABLISHED,RELATED -j ACCEPT

	$IPTABLES -A OUTPUT -o $EXTERNAL_INTERFACE -p udp -s 0/0 --sport $udp -d 0/0 --dport 1024:65535 -
m state --state ESTABLISHED,RELATED -j ACCEPT

		done

		fi


	# Allow listed ICMP Types/Codes in

		if [ "${ICMP_ACCEPT}" != "" ]

		then

		for icmp in ${ICMP_ACCEPT}

		do


		$IPTABLES -A INPUT -i $EXTERNAL_INTERFACE -p icmp -s 0/0 --icmp-type $icmp -d 0/0 -m state
--state NEW,ESTABLISHED,RELATED -j ACCEPT


		done

		fi


		# Block Broadcast, Multicast and Unicast traffic from being logged
```

```
		if [ "${BCAST_BLOCK}" != "" ]
		then
		for bcast in ${BCAST_BLOCK}
		do

		echo $IPTABLES -A INPUT -i $EXTERNAL_INTERFACE -m pkttype --pkt-type $bcast -j DROP

		done
		fi



	# Allow listed sources in
	if [ "${TRUSTED_SOURCE}" != "" ]
	then
	for trusted in ${TRUSTED_SOURCE}
	do
	$IPTABLES -A INPUT -i $EXTERNAL_INTERFACE -s $trusted -d 0/0 -m state --state
NEW,ESTABLISHED,RELATED -j ACCEPT
	done
	fi
		#--------#
	# SSH/22 #
		#--------#
		if [ "${SSH_CLIENTS}" != "" ]
		then
		for ssh in ${SSH_CLIENTS}
		do
		$IPTABLES -A INPUT -i $EXTERNAL_INTERFACE -p tcp -s $ssh -d 0/0 --dport 22 -m state --state
NEW,ESTABLISHED -j ACCEPT
		done
		fi

		#-----------#
		# RSYNC/873 #
		#-----------#
		if [ "${RSYNC_CLIENTS}" != "" ]
		then
		for rsync in ${RSYNC_CLIENTS}
```

```
                do

                IPTABLES -A INPUT -i $EXTERNAL_INTERFACE -p tcp -s $rsync -d 0/0 --dport 873 -m state --state
NEW,ESTABLISHED -j ACCEPT

                done

                fi




        if [ "${NRPE}" != "" ]

        then

        for nrpe in ${NRPE}

        do

        $IPTABLES -A INPUT -p tcp -i $EXTERNAL_INTERFACE -s $nrpe --sport 1024:65535 -d 0/0 --dport 5666 -
m state --state NEW,ESTABLISHED -j ACCEPT

        $IPTABLES -A OUTPUT -p tcp -o $EXTERNAL_INTERFACE -s 0/0 --sport 5666 -d $nrpe --dport 1024:65535
-m state --state ESTABLISHED -j ACCEPT

        done

        fi



            # Allow IRLP traffic

            if [ "${IRLPTCP}" != "" ]

            then

            for irlptcp in ${IRLPTCP}

            do

            $IPTABLES -t nat -A PREROUTING -p tcp -i $EXTERNAL_INTERFACE -s 0/0 --sport 1024:65535 -d 0/0
--dport $irlptcp -j DNAT --to-dest $IRLP

            $IPTABLES -A FORWARD -p tcp -i $EXTERNAL_INTERFACE -s 0/0 --sport 1024:65535 -o
$INTERNAL_INTERFACE -d $IRLP --dport $irlptcp -m state --state NEW,ESTABLISHED -j ACCEPT

            $IPTABLES -A FORWARD -p tcp -i $INTERNAL_INTERFACE -s $IRLP --sport $irlptcp -o
$EXTERNAL_INTERFACE -d 0/0 --dport 1024:65535 -m state --state ESTABLISHED -j ACCEPT

            $IPTABLES -t nat -A POSTROUTING -p tcp -o $EXTERNAL_INTERFACE -s $IRLP --sport $irlptcp -d 0/0
--dport 1024:65535 -j MASQUERADE

            done

            fi



            if [ "${IRLPUDP}" != "" ]

            then

            for irlpudp in ${IRLPUDP}

            do

            $IPTABLES -t nat -A PREROUTING -p udp -i $EXTERNAL_INTERFACE -s 0/0 --sport 1024:65535 -d
0/0 --dport $irlpudp -j DNAT --to-dest $IRLP
```

```
        $IPTABLES -A FORWARD -p udp -i $EXTERNAL_INTERFACE -s 0/0 --sport 1024:65535 -o
$INTERNAL_INTERFACE -d $IRLP --dport $irlpudp -m state --state NEW,ESTABLISHED -j ACCEPT

        $IPTABLES -A FORWARD -p udp -i $INTERNAL_INTERFACE -s $IRLP --sport $irlpudp -o
$EXTERNAL_INTERFACE -d 0/0 --dport 1024:65535 -m state --state ESTABLISHED -j ACCEPT

        $IPTABLES -t nat -A POSTROUTING -p udp -o $EXTERNAL_INTERFACE -s $IRLP --sport $irlpudp -d
0/0 --dport 1024:65535 -j MASQUERADE

        done

        fi


            # Allow all outbound from Internal Interface

            $IPTABLES -A FORWARD -i $INTERNAL_INTERFACE -o $EXTERNAL_INTERFACE -d 0/0 -m state --
state NEW,ESTABLISHED,RELATED -j ACCEPT

            $IPTABLES -A FORWARD -i $EXTERNAL_INTERFACE -s 0/0 -o $INTERNAL_INTERFACE -m state --
state ESTABLISHED,RELATED -j ACCEPT

            $IPTABLES -t nat -A POSTROUTING -o $EXTERNAL_INTERFACE -m state --state
NEW,ESTABLISHED,RELATED -j MASQUERADE


            # Permit all new and established traffic out

            $IPTABLES -A OUTPUT -o $EXTERNAL_INTERFACE -s 0/0 -d 0/0 -m state --state
NEW,ESTABLISHED,RELATED -j ACCEPT

            $IPTABLES -A INPUT -i $EXTERNAL_INTERFACE -s 0/0 -d 0/0 -m state --state
ESTABLISHED,RELATED -j ACCEPT


            # Logging and dropping everything else

            # TCP

            $IPTABLES -A INPUT -i $EXTERNAL_INTERFACE -p tcp -j LOG --log-prefix "IPT TCP In: "

            $IPTABLES -A INPUT -i $EXTERNAL_INTERFACE -p tcp -j DROP

            # UDP

            $IPTABLES -A INPUT -i $EXTERNAL_INTERFACE -p udp -j LOG --log-prefix "IPT UDP In: "

            $IPTABLES -A INPUT -i $EXTERNAL_INTERFACE -p udp -j DROP

            # ICMP

            $IPTABLES -A INPUT -i $EXTERNAL_INTERFACE -p icmp -j LOG --log-prefix "IPT ICMP In: "

            $IPTABLES -A INPUT -i $EXTERNAL_INTERFACE -p icmp -j DROP

            # Everything else

            $IPTABLES -A INPUT -i $EXTERNAL_INTERFACE -j LOG --log-prefix "IPT Misc In: "

            $IPTABLES -A INPUT -i $EXTERNAL_INTERFACE -j DROP


            log_end_msg $?

            ;;

    stop)
```

```
                log_daemon_msg "Stopping Firewall Services"

                # Setting default policies to accept
                $IPTABLES -P INPUT ACCEPT
                $IPTABLES -P OUTPUT ACCEPT
                $IPTABLES -P FORWARD ACCEPT

                # Flush all rules
                $IPTABLES -F
                $IPTABLES -t nat -F

                log_end_msg $?
                ;;
        restart)
                $0 stop && $0 start
                ;;
        *)
                echo "Usage: $0 {start|stop|restart}"
                exit 2
                ;;
esac
```