

# Forum:[Hack Forums](#)

Thread: [Linux Protection Script \(VPN/TOR Script for OpSec\)](#)

**Burger** - June 1, 2024, 5:27 pm

I have recently found a script that I created awhile back that utilizes Private Internet Access (PIA VPN) and Tor to create a secure network for your server or personal computer running Linux. This was created on Debian and has been used with Deb / Ubuntu!

Take a peek and try it out pretty simple code and easy to configure, message me if you have any problems!

I have never charged for my scripts / help however I would love to get the byte awards and make some flips today so please give me a Quick Love and or a donation of bytes if you use this!

```
Code #!/bin/bash
# Line 27- Change bangbang to hostname
# Tor must be installed - apt-get install tor
# PIA must be installed - install-pia.sh -> Manual Download will be needed easy to install!
```

```
PIA_HOSTNAMES="sweden france ca germany swiss us-california us-seattle us-texas nl
uk-london us-east us-midwest us-florida israel ro hk us-west uk-southampton"
VPN_IPS=""
```

```
# Flush all rules
sudo iptables -F
sudo iptables -X
sudo iptables -t nat -F
sudo iptables -t nat -X
sudo iptables -t mangle -F
sudo iptables -t mangle -X
sudo iptables -P INPUT ACCEPT
sudo iptables -P OUTPUT ACCEPT
sudo iptables -P FORWARD ACCEPT
sudo echo "nameserver 8.8.8.8" > /etc/resolv.conf
echo "Flushed."
```

```
if [ "$1" == "flush" ]; then
    exit 0
fi
```

```
echo "Preparing resolv.conf"
sudo echo 1 > /proc/sys/net/ipv4/ip_forward
sudo echo -ne "domain BANGBANG\nsearch BANGBANG\nnameserver 8.8.8.8\n" >>
/etc/resolv.conf
sudo echo -e "127.0.0.1\tlocalhost" > /etc/hosts
sudo echo -e "127.0.0.1\tsecure" >> /etc/hosts
```

```
echo "Resolving all PIA hostnames"
for HOSTNAME in $PIA_HOSTNAMES
```

```

do
  DIG_OUTPUT=$(dig "$HOSTNAME".privateinternetaccess.com +short)
  for IADDR in $DIG_OUTPUT
  do
    VPN_IPS=$(echo "$VPN_IPS $IADDR/32")
    sudo echo -e "$IADDR\t$HOSTNAME.privateinternetaccess.com" >> /etc/hosts
  done
done
VPN_IPS=$(echo "$VPN_IPS" | sed 's/^ */')

# Drop everything by default
echo "Dropping all traffic"
sudo iptables -P INPUT DROP
sudo iptables -P OUTPUT DROP
sudo iptables -P FORWARD DROP

# Whitelist all VPN IPs
echo "Whitelisting PIA IP addresses on eth0"
for IP in $VPN_IPS
do
  sudo iptables -A INPUT -s $IP -i eth0 -j ACCEPT
  sudo iptables -A OUTPUT -d $IP -o eth0 -j ACCEPT
done

# Allow the tun0 device
echo "Allowing only 'debian-tor' to communicate via tun0"
sudo iptables -A INPUT -i tun0 -m state --state RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A OUTPUT -o tun0 -m owner --uid-owner debian-tor -j ACCEPT

echo "Allow communication with Google's DNS servers"
sudo iptables -A INPUT -s 8.8.8.8 -j ACCEPT
sudo iptables -A OUTPUT -d 8.8.8.8 -j ACCEPT

# Allow local interface
echo "Enable communicating with loopback interface"
sudo iptables -A INPUT -i lo -j ACCEPT
sudo iptables -A OUTPUT -o lo -j ACCEPT

# Allow outgoing and incoming NTP traffic to/from NIST.
echo "Enable NTP traffic to NIST"
sudo iptables -A OUTPUT -p udp -m iprange --dst-range 129.6.15.28-129.6.15.30 --dport 123 -j ACCEPT
sudo iptables -A INPUT -p udp -m iprange --src-range 129.6.15.28-129.6.15.30 --sport 123 -j ACCEPT

```

**Nomad.Null** - June 1, 2024, 6:56 pm

Stopping by to throw in a vouch and a code review for [Burger](#)

As a courtesy, I went and validated that this script is, more or less, safe to run. It simply modifies the IPtables rules to be suitable for forwarding traffic over an existing PIA network followed by a TOR daemon, and successfully getting that traffic back over TCP sessions.

The code doesn't hide anything malicious, but it's a good idea to understand it for the sake of

preventing troubleshooting problems.

Although, I would strongly recommend doing a backup of your IPtables before proceeding, as this will add a LOT of rules that will be difficult to undo without restoring.

Here is how to do both actions as a safety tip.

Backup

Code iptables-save >> /etc/iptables.rules

Restore

Code iptables-restore << /etc/iptables.rules

**Burger** - June 1, 2024, 7:09 pm

(Jun 1, 2024 06:56 PM)Nomad.Null Wrote: [\[CITE\]](#) Stopping by to throw in a vouch and a code review for [Burger](#)

As a courtesy, I went and validated that this script is, more or less, safe to run. It simply modifies the IPtables rules to be suitable for forwarding traffic over an existing PIA network followed by a TOR daemon, and successfully getting that traffic back over TCP sessions.

The code doesn't hide anything malicious, but it's a good idea to understand it for the sake of preventing troubleshooting problems.

Although, I would strongly recommend doing a backup of your IPtables before proceeding, as this will add a LOT of rules that will be difficult to undo without restoring.

Here is how to do both actions as a safety tip.

Backup

Code iptables-save >> /etc/iptables.rules

Restore

Code iptables-restore << /etc/iptables.rules

Thanks for the review my man, I appreciate you taking your time and giving an overview to the newer users who might not understand what is happening with the bash script.

This is just a tool I used to use to keep myself abit more secure :)

**LEGENDAVIE** - June 2, 2024, 3:50 am

It's a nice basic script but there are couple problems that would compromise the entire purpose of the script:

1. You use Google DNS(8.8.8.8) which is being logged, not very effective if you're after anonimosity. Maybe use PIA's instead.
2. There's no rules to block traffic in case of an unexpected VPN drop. Maybe add a kill switch:

Code sudo iptables -A OUTPUT ! -o tun0 -m owner --uid-owner debian-tor -j DROP  
sudo iptables -A OUTPUT ! -o tun0 -p udp --dport 53 -j DROP

**BitcoinLord9** - June 2, 2024, 4:42 am

Is this script compatible only with PIA, or can it also be used with other VPNs like N-o-r-d and IPVanish?

**Burger** - June 2, 2024, 10:24 am

(Jun 2, 2024 03:50 AM)LEGENDAVIE Wrote: [\[CITE\]](#) It's a nice basic script but there are couple problems that would compromise the entire purpose of the script:

1. You use Google DNS(8.8.8.8) which is being logged, not very effective if you're after anonimosity. Maybe use PIA's instead.
2. There's no rules to block traffic in case of an unexpected VPN drop. Maybe add a kill switch:

```
Code sudo iptables -A OUTPUT ! -o tun0 -m owner --uid-owner debian-tor -j DROP
sudo iptables -A OUTPUT ! -o tun0 -p udp --dport 53 -j DROP
```

Yeah I developed this probably 8+ years ago if not longer so there is some pretty outdated stuff attached to it and I agree I wouldnt use Googles Public DNS servers anymore.

Thanks for the review and honest feedback!

(Jun 2, 2024 04:42 AM)BitcoinLord9 Wrote: [\[CITE\]](#) Is this script compatible only with PIA, or can it also be used with other VPNs like N-o-r-d and IPVanish?

You could repurpose this script to work with another VPN Provider. If you aren't to tech savvy then ChatGPT could probably help you make some changes to the code and get it to work!

Note: I don't recommend relying on ChatGPT to do any major coding for you however It is a great tool for beginners to help with Debug an Functionality.

**Dawidosek** - June 2, 2024, 4:02 pm

(Jun 1, 2024 05:27 PM)Burger Wrote: [\[CITE\]](#) I have recently found a script that I created awhile back that utilizes Private Internet Access (PIA VPN) and Tor to create a secure network for your server or personal computer running Linux. This was created on Debian and has been used with Deb / Ubuntu!

Take a peek and try it out pretty simple code and easy to configure, message me if you have any problems!

I have never charged for my scripts / help however I would love to get the byte awards and make some flips today so please give me a Quick Love and or a donation of bytes if you use this!

```
Code #!/bin/bash
# Line 27- Change bangbang to hostname
# Tor must be installed - apt-get install tor
# PIA must be installed - install-pia.sh -> Manual Download will be needed easy to install!
```

```
PIA_HOSTNAMES="sweden france ca germany swiss us-california us-seattle us-texas nl
uk-london us-east us-midwest us-florida israel ro hk us-west uk-southampton"
VPN_IPS=""
```

```
# Flush all rules
sudo iptables -F
sudo iptables -X
sudo iptables -t nat -F
sudo iptables -t nat -X
sudo iptables -t mangle -F
sudo iptables -t mangle -X
sudo iptables -P INPUT ACCEPT
sudo iptables -P OUTPUT ACCEPT
sudo iptables -P FORWARD ACCEPT
```

```
sudo echo "nameserver 8.8.8.8" > /etc/resolv.conf
echo "Flushed."
```

```
if [ "$1" == "flush" ]; then
    exit 0
fi
```

```
echo "Preparing resolv.conf"
sudo echo 1 > /proc/sys/net/ipv4/ip_forward
sudo echo -ne "domain BANGBANG\nsearch BANGBANG\nnameserver 8.8.8.8\n" >>
/etc/resolv.conf
sudo echo -e "127.0.0.1\tlocalhost" > /etc/hosts
sudo echo -e "127.0.0.1\tsecure" >> /etc/hosts
```

```
echo "Resolving all PIA hostnames"
for HOSTNAME in $PIA_HOSTNAMES
do
    DIG_OUTPUT=$(dig "$HOSTNAME".privateinternetaccess.com +short)
    for IADDR in $DIG_OUTPUT
    do
        VPN_IPS=$(echo "$VPN_IPS $IADDR/32")
        sudo echo -e "$IADDR\t$HOSTNAME.privateinternetaccess.com" >> /etc/hosts
    done
done
VPN_IPS=$(echo "$VPN_IPS" | sed 's/^ */')
```

```
# Drop everything by default
echo "Dropping all traffic"
sudo iptables -P INPUT DROP
sudo iptables -P OUTPUT DROP
sudo iptables -P FORWARD DROP
```

```
# Whitelist all VPN IPs
echo "Whitelisting PIA IP addresses on eth0"
for IP in $VPN_IPS
do
    sudo iptables -A INPUT -s $IP -i eth0 -j ACCEPT
    sudo iptables -A OUTPUT -d $IP -o eth0 -j ACCEPT
done
```

```
# Allow the tun0 device
echo "Allowing only 'debian-tor' to communicate via tun0"
sudo iptables -A INPUT -i tun0 -m state --state RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A OUTPUT -o tun0 -m owner --uid-owner debian-tor -j ACCEPT
```

```
echo "Allow communication with Google's DNS servers"
sudo iptables -A INPUT -s 8.8.8.8 -j ACCEPT
sudo iptables -A OUTPUT -d 8.8.8.8 -j ACCEPT
```

```
# Allow local interface
echo "Enable communicating with loopback interface"
sudo iptables -A INPUT -i lo -j ACCEPT
sudo iptables -A OUTPUT -o lo -j ACCEPT
```

```
# Allow outgoing and incoming NTP traffic to/from NIST.  
echo "Enable NTP traffic to NIST"  
sudo iptables -A OUTPUT -p udp -m iprange --dst-range 129.6.15.28-129.6.15.30 --dport 123 -j  
ACCEPT  
sudo iptables -A INPUT -p udp -m iprange --src-range 129.6.15.28-129.6.15.30 --sport 123 -j  
ACCEPT  
Best
```