

# Firewall Setup

## Install Shorewall

To manage `nftables/iptables` I decided to go with [Shorewall](#) since it is easy to configure and very mature. At some point I may look into switching to [FireHol](#) since it looks even simpler to configure but I wanted something I knew I'd be able to make do everything I needed.

I started by installing *shorewall* as my firewall, *shorewall-doc* which includes examples, and *shorewall-init* which can lockdown the system at boot before *Shorewall* has had a chance to configure the firewall.

```
# apt install shorewall shorewall-doc shorewall-init
```

Then I update the *shorewall* configuration to reflect that I'm using *ulogd2* for logging and that I want IPv4 forwarding enabled when *shorewall* starts.

```
# /etc/shorewall/shorewall.conf
- LOG_LEVEL="info"
+ LOG_LEVEL="NFLOG(1,0,1)"
...
- LOGFILE=/var/log/messages
+ LOGFILE=/var/log/firewall.log
...
- IP_FORWARDING=Keep
+ IP_FORWARDING=Yes
```

All my configuration files are adapted from the examples that *shorewall-doc* makes available under `/usr/share/doc/shorewall/examples`.

Setting up the zones is pretty self-explanatory. The only addition I made is I have a `warp` zone which I will use later when I am setting up my VPN.

```
# /etc/shorewall/zones
+ #-----
+ # For information about entries in this file, type "man shorewall-zones"
```

```

+ #
+ # See http://shorewall.net/manpages/shorewall-zones.html for more information
+
#####
#####
+ #ZONE  TYPE  OPTIONS          IN          OUT
+ #                OPTIONS          OPTIONS
+ fw     firewall
+ wan    ipv4
+ lan    ipv4
+ dmz    ipv4
+ warp   ipv4

```

Setting up the interfaces and assigning them zones is also pretty self-explanatory.

```

# /etc/shorewall/interfaces
+ #-----
+ # For information about entries in this file, type "man shorewall-interfaces"
+ #
+ # See http://shorewall.net/manpages/shorewall-interfaces.html for more information
+
#####
#####
+ ?FORMAT 2
+
#####
#####
+ #ZONE[]INTERFACE  OPTIONS
+ wan[]WAN_IF[]tcpflags,dhcp,nosmurfs,routefilter,logmartians,sourceroute=0,physical=eth0
+ lan[]LAN_IF[]tcpflags,dhcp,nosmurfs,routefilter,logmartians,physical=eth1
+ dmz[]DMZ_IF[]tcpflags,dhcp,nosmurfs,routefilter,logmartians,physical=eth1.8
+ warp[]WARP_IF[]tcpflags,dhcp,nosmurfs,routefilter,logmartians,physical=eth1.9

```

My real `/etc/shorewall/policy` file is less liberal than what is shown below (`lan` being allowed to access whatever it wants) but I wanted to show a reasonably secure policy that allowed me to have a very simple `/etc/shorewall/rules` config below.

```

# /etc/shorewall/policy
+ #-----
+ # For information about entries in this file, type "man shorewall-policy"
+ #

```

```

+ # See http://shorewall.net/manpages/shorewall-policy.html for more information
+
#####
#####
+ #SOURCE[]DEST[]POLICY[]LOGLEVEL[]RATE  CONNLIMIT
+
+ $FW[]all[]ACCEPT
+ lan[]all[]ACCEPT
+ dmz[]$FW,wan[]ACCEPT
+ warp[]$FW[]ACCEPT
+
+ wan[]all[]DROP[]$LOG_LEVEL
+ # THE FOLLOWING POLICY MUST BE LAST
+ all[]all[]REJECT[]$LOG_LEVEL

```

Because my example policy is pretty open, my rules in this example are pretty sparse.

```

# /etc/shorewall/rules
+ #-----
+ # For information about entries in this file, type "man shorewall-rules"
+ #
+ # See http://shorewall.net/manpages/shorewall-rules.html for more information
+
#####
#####
#####
+ #ACTION      SOURCE      DEST      PROTO  DEST  SOURCE      ORIGINAL  RATE      USER/
MARK  CONNLIMIT    TIME      HEADERS  SWITCH  HELPER
+ #              PORT   PORT(S)   DEST      LIMIT    GROUP
+ ?SECTION ALL
+ ?SECTION ESTABLISHED
+ ?SECTION RELATED
+ ?SECTION INVALID
+ ?SECTION UNTRACKED
+ ?SECTION NEW
+
+ #      Don't allow connection pickup from the net
+ Invalid(DROP)  wan      all      tcp
+
+ DNS(ACCEPT)   all!wan,warp  $FW

```

```
+ DNS(ACCEPT)  $FW,dmz      lan:10.0.1.2
+
+ Web(ACCEPT)   dmz         $FW
+ Web(DNAT)     wan         dmz:10.0.8.2
```

Lastly is the magic that allows private addresses to access the Internet by masquerading them all as my one public IPv4 address I am assigned. The following just says all traffic heading out of

`WAN_IF` (`eth0`) coming from a private IP range should be [masqueraded](#).

```
# /etc/shorewall/snat
+ #-----
+ # For information about entries in this file, type "man shorewall-snat"
+ #
+ # See http://shorewall.net/manpages/shorewall-snat.html for more information
+
#####
#####
#####
+ #ACTION          SOURCE          DEST          PROTO  PORT  IPSEC  MARK  USER
SWITCHORIGDEST PROBABILITY
+ MASQUERADE        10.0.0.0/8,\
+                   169.254.0.0/16,\
+                   172.16.0.0/12,\
+                   192.168.0.0/16      WAN_IF
```

Now that I have everything configured it might be wise to run `shorewall check` just to make sure I didn't have any typos.

I hooked *shorewall* into the boot process to make sure the system is secure during boot by enabling *shorewall-init.service* and *shorewall.service*. First I told *shorewall-init* that it needs to account for *shorewall* when it runs.

```
# /etc/default/shorewall-init
- PRODUCTS=""
+ PRODUCTS="shorewall"
```

Then I simply told those services to start at boot.

```
# systemctl enable shorewall
# systemctl enable shorewall-init
```

# Modify Interfaces

Now that *Shorewall* will secure everything at bootup it is safe to update `/etc/networking/interfaces` and add their IPv4 addresses.

```
# /etc/networking/interfaces
auto eth1
- iface eth1 inet manual
+ iface eth1 inet static
+     address 10.0.1.1/21

auto eth1.8
- iface eth1.8 inet manual
+ iface eth1.8 inet static
+     vlan-raw-device eth1
+     address 10.0.8.1/24

auto eth1.9
- iface eth1.9 inet manual
+ iface eth1.9 inet static
+     vlan-raw-device eth1
+     address 10.0.9.1/24
```

Now if I reboot the system all my interfaces will come up configured and the system will be protected by *nftables/iptables* configured by *Shorewall*.

Be sure to sanity check the configuration so *Shorewall* doesn't block SSH access if that is needed.

```
# reboot
```

---

Revision #5

Created 31 March 2021 02:29:20 by dustin@swigg.net

Updated 8 April 2021 12:50:13 by dustin@swigg.net