Using VyOS as a Firewall

Disclaimer: This guide will provide a technical deep-dive into VyOS as a firewall and assumes basic knowledge of networking, firewalls, Linux and Netfilter, as well as VyOS CLI and configuration basics. This guide was written in hopes that it will be useful to others and makes no claim of responsibility for security incidents related to advice in this document. USE AT YOUR OWN RISK.

To learn more about VyOS, visit the project website at **vyos.net**.

PROLOGUE

***"For us, open source isn't just a business model;
it's smart engineering practice."***

– Bruce Schneier

Argument 1: Open Source as a requirement for Security Infrastructure

In 2013, Edward Snowden, an Infrastructure Analyst working for Booz Allen Hamilton, a US defense contractor, and assigned to the NSA, provided the public with classified information detailing the extent of NSA programs used to compromise security infrastructure, including the use of backdoors on major network firewall platforms. The list of platforms specifically cited include offerings from Cisco, Juniper, and Huawei.

A major concern following these revelations, was not only the ability to trust the integrity of security infrastructure, but also in the perceived disregard or arrogance demonstrated by the NSA in the assumption that these tools or details on how to exploit them would not fall into the purview of bad actors.

As a direct result of these events there is a growing sense within the security community that access to source code is a requirement for security.

Argument 2: Addressing Performance Concerns

In the past there were performance limitations that prevented the use of software-based solutions for network infrastructure.

Today provides much more flexibility. Using modern hardware, Linux-based systems are able to provide a viable solution for throughput needs in the multi- Gigabit realm, easily reaching the 2M pps target for line-rate Gigabit at 64-byte packets.

This performance is quickly growing thanks to efforts like Intel® DPDK which pushes Linux-based systems into the multi- Ten Gigabit area for packet processing (100M pps and beyond). While there are no open source implementations of DPDK-accelerated Linux today, it does show promise for scaling Linux-based network infrastructure, and proprietary Vyatta vRouter 5600 from Brocade® purports to do just that through their vPlane™ technology, though still in early stages of implementation, it does provide encouragement as a growth option until the FOSS community can catch up.

Argument 3: Security through Consistency

As VyOS is designed to run on x86 architecture, it presents one of the first firewall solutions that allows for hardware to truly be right-sized independent of OS or CLI. VyOS can not only be run on a wide range of physical hardware, but also as a virtual instance allowing for NFV (network function virtualization) applications.

The use of one OS and one CLI across firewall solutions within an organization significantly reduces the complexity of the creation and audit of policy and mitigates the human error in policy creation as a result of having to master multiple configuration syntaxes and platforms.

The majority of security incidents can be linked back to configuration error. The human-readable and verbose configuration syntax of VyOS presents clear policy definition that can easily be understood by security professionals without specific knowledge of the platform.

The image-based install process of VyOS allows for new releases to be installed in a non-destructive way. While many solutions share a configuration that may be altered through the upgrade process, VyOS maintains a separate copy of its configuration for each image, allowing for changes to be reverted without the risk of backward-incompatibility. Finally the unified, human-readable and text-based, configuration file of VyOS allows for configuration to be backed up using standard revision control systems.

Operationally these design choices promote the maintenance and operation of VyOS to significantly improve the overall security posture of an organization.

Argument 4: Total Cost of Ownership

The CapEx component of VyOS is a clear winner. The OS is zero-cost and the hardware can be right-sized to application requirements.

A common fear associated with the use of Linux for network and security infrastructure is the out of control OpEx component due to the number of unknowns. The answer to this is that VyOS is not simply a Linux distribution with network and firewall functionality, but a modern network operating system with a consistent CLI and configuration syntax, and clear upgrade process. As a result the OpEx component of VyOS can be considered equivalent to a Cisco or Juniper solution.

Part 1: Dive Deep, VyOS Internals and Linux Netfilter

The firewall for VyOS is powered by Linux Netfilter (more commonly known by it’s user-space utility “iptables”). Netfilter is one of the most widely adopted and peer-reviewed firewall implementations in the world.

Firewall policy in VyOS can be applied using two methods: Per-Interface or Zone Policy. This guide will use the more common per-interface policy.

For per-interface, policy can be applied to an interface in three directions, “in”, “out”, and “local”. For those familiar with Linux Netfilter it is important not to confuse “in” for the “INPUT” chain of iptables. The “in” policy is instead applied to traffic received being forwarded through the interface, representing the “FORWARD” chain in iptables specific to the interface, e.g. “-i” for incoming. Similarly, the “out” direction is also matched using the “FORWARD” chain but using “-o” for outgoing instead. Finally, the “local” policy actually represents the “INPUT” chain with the “-i” flag set to the interface being configured.

Firewall rules can reference group objects (implemented using the “ipsets” sub-project of Netfilter). The group types supported are “network”, “address”, and “port”. The “network” group object is implemented as a “hash:net” ipset. The “address” group object as “hash:ip”. And the “port” group object as “bitmap:port”.

Each named policy is implemented as a chain in iptables. Consider the following example:

```
set firewall name INSIDE-LOCAL default-action 'drop'
```

Generates the following iptables commands:

```
-N INSIDE-LOCAL
-A INSIDE-LOCAL -m comment --comment "INSIDE-LOCAL-10000 default-action drop" -j DROP
```

And applying the policy to an interface “eth0”:

```
set interfaces bonding eth0 firewall local name 'INSIDE-LOCAL'
```

The following:

```
-A VYATTA_FW_LOCAL_HOOK -i eth0 -j INSIDE-LOCAL
```

One thing to note is that the “accept” action in a firewall rule will be implemented as “RETURN” which jumps back to the parent chain for further processing.

VyOS implements the following policy by default:

```
-N VYATTA_FW_IN_HOOK
-N VYATTA_FW_LOCAL_HOOK
-N VYATTA_FW_OUT_HOOK
-N VYATTA_POST_FW_FWD_HOOK
```

```
-N VYATTA_POST_FW_IN_HOOK
-N VYATTA_POST_FW_OUT_HOOK
-N VYATTA_PRE_FW_FWD_HOOK
-N VYATTA_PRE_FW_IN_HOOK
-N VYATTA_PRE_FW_OUT_HOOK

-A VYATTA_POST_FW_FWD_HOOK -j ACCEPT
-A VYATTA_POST_FW_IN_HOOK -j ACCEPT
-A VYATTA_POST_FW_OUT_HOOK -j ACCEPT
-A VYATTA_PRE_FW_FWD_HOOK -j RETURN
-A VYATTA_PRE_FW_IN_HOOK -j RETURN
-A VYATTA_PRE_FW_OUT_HOOK -j RETURN

-A INPUT -j VYATTA_PRE_FW_IN_HOOK
-A INPUT -j VYATTA_FW_LOCAL_HOOK
-A INPUT -j VYATTA_POST_FW_IN_HOOK
-A FORWARD -j VYATTA_PRE_FW_FWD_HOOK
-A FORWARD -j VYATTA_FW_IN_HOOK
-A FORWARD -j VYATTA_FW_OUT_HOOK
-A FORWARD -j VYATTA_POST_FW_FWD_HOOK
-A OUTPUT -j VYATTA_PRE_FW_OUT_HOOK
-A OUTPUT -j VYATTA_POST_FW_OUT_HOOK
```

The “pre” and “post” hooks exist for internal policy rules (such as the final accept) while the majority of filtering is applied using “VYATTA_FW_LOCAL_HOOK” for “local”, “VYATTA_FW_IN_HOOK” for “in”, and “VYATTA_FW_OUT_HOOK” for “out”.

This model of using the “RETURN” target in place of “ACCEPT” to allow further processing if needed restricts the ability for VyOS to support firewall rules referencing an additional policy as a target (or “jumping” to another chain).

To those experienced with Netfilter, this may seem overly restrictive, but it does have a side-effect of enforcing simple, easy-to-read, policy which improves overall security posture by reducing human error or oversight.

An important aspect of VyOS as a firewall which differentiates itself from other Linux-based options is that firewall changes are dynamic in nature.

A typical Linux-based firewall will flush all chains and tables then rebuild policy to implement changes. This not only has the undesired effect of zeroing counters, but can introduce a race-condition for firewalls where the delay between removing and reinserting rules may apply inconsistent or incomplete policy to traffic and potentially trigger connection tracking to mark a flow as invalid.

To avoid this, VyOS only modifies rules that are modified in the configuration system. This means that changes to even large policy rules are non-destructive and will not alter traffic flow matched by other rules. This is especially true if used for stateful packet inspection and a policy to accept established traffic is in use allowing policy changes to be made in production.

A consequence of this model is that manual configuration of iptables can place the system into an inconsistent state where the expected system state from a configuration standpoint does not match the actual Netfilter policy in place. For this reason **you should never invoke Netfilter tools such as “iptables” or “ipset” directly.**

Part 2: Configuration By Example

We learned in the previous section that policy is defined as a named set of firewall rules and applied to a network interface for a direction (“in”, “out”, or “local”).

In this section we’ll take a look at a basic firewall configuration to build a typical firewall configuration.

A standard three zone firewall is assumed “outside” or “WAN” on interface eth0, “inside” or “LAN” on eth1, and a “DMZ” on eth2.

- Traffic for the LAN will use private IP addressing and NAT.
- Traffic for the DMZ will use public IP addressing.
- Traffic from the LAN to the DMZ will be permitted by default.
- Traffic from the DMZ to the LAN will be dropped by default.
- Traffic from the WAN will be dropped by default.

The IP networks used will be:

- WAN: 198.51.100.0/29
- LAN: 10.1.0.0/24
- DMZ: 203.0.113.0/24

Our basic starting interface configuration will be as follows:

```
set interfaces ethernet eth0 description 'WAN'
set interfaces ethernet eth0 address '198.51.100.6/29'
set interfaces ethernet eth1 description 'LAN'
set interfaces ethernet eth1 address '10.1.0.1/24'
set interfaces ethernet eth2 description 'DMZ'
set interfaces ethernet eth2 address '203.0.113.1/24'

set protocols static route 0.0.0.0/0 next-hop '198.51.100.1'
```

Our first goal will be to create local policy for traffic destined to the firewall itself.

There are two ways we can handle stateful packet inspection. The first and easier method is to set the global firewall state policy:

```
set firewall state-policy established action 'accept'
set firewall state-policy related action 'accept'
set firewall state-policy invalid action 'drop'
```

This will be implemented using the “pre” hook chains discussed in the previous section and applied before any user-defined rules are processed. The three states are matched against the Linux “conntrack” table which is used to implement stateful packet inspection in Netfilter.

The second option is to define state rules in each policy. The advantage of doing so is the ability to filter before the state check. For example you may wish to add logging of matched packets for a particular state for troubleshooting or you may wish to blacklist traffic before its connection state is evaluated.

We will opt to use manual method in this example.

Before we create our firewall policy, there are a few group objects we will want in place so they can be referenced:

- A network group for each of the 3 networks (WAN, LAN, and DMZ).
- A network group for systems authorized to manage the firewall (SSH and SNMP).
- A network group of blacklisted hosts or networks.

```
set firewall group network-group NET-BLACKLISTED network '1.1.1.1/32'
set firewall group network-group NET-DMZ network '203.0.113.0/24'
set firewall group network-group NET-LAN network '10.1.0.0/24'
set firewall group network-group NET-MANAGEMENT network '10.1.0.10/32'
set firewall group network-group NET-MANAGEMENT network '10.1.0.11/32'
set firewall group network-group NET-WAN network '198.51.100.0/24'
```

Note that we can reference individual IP addresses as part of a network group object by using a prefix-length of 32.

Next we will create a named local policy for each interface:

```
set firewall name WAN-LOCAL default-action 'drop'
set firewall name WAN-LOCAL rule 1000 action 'drop'
set firewall name WAN-LOCAL rule 1000 source group network-group 'NET-BLACKLISTED'
set firewall name WAN-LOCAL rule 1010 action 'accept'
set firewall name WAN-LOCAL rule 1010 state established 'enable'
set firewall name WAN-LOCAL rule 1010 state related 'enable'
set firewall name WAN-LOCAL rule 1011 action 'drop'
set firewall name WAN-LOCAL rule 1011 state invalid 'enable'
set firewall name WAN-LOCAL rule 1020 action 'accept'
set firewall name WAN-LOCAL rule 1020 icmp type-name 'echo-request'
set firewall name WAN-LOCAL rule 1020 protocol 'icmp'
set firewall name WAN-LOCAL rule 1020 state new 'enable'
set firewall name WAN-LOCAL rule 1100 action 'drop'
set firewall name WAN-LOCAL rule 1100 destination port '22'
set firewall name WAN-LOCAL rule 1100 protocol 'tcp'
set firewall name WAN-LOCAL rule 1100 recent count '4'
set firewall name WAN-LOCAL rule 1100 recent time '60'
set firewall name WAN-LOCAL rule 1100 source group network-group 'NET-MANAGEMENT'
set firewall name WAN-LOCAL rule 1100 state new 'enable'
set firewall name WAN-LOCAL rule 1101 action 'accept'
set firewall name WAN-LOCAL rule 1101 destination port '22'
set firewall name WAN-LOCAL rule 1101 protocol 'tcp'
set firewall name WAN-LOCAL rule 1101 source group network-group 'NET-MANAGEMENT'
set firewall name WAN-LOCAL rule 1101 state new 'enable'
set firewall name WAN-LOCAL rule 1110 action 'accept'
set firewall name WAN-LOCAL rule 1110 destination port '161'
set firewall name WAN-LOCAL rule 1110 protocol 'udp'
set firewall name WAN-LOCAL rule 1110 source group network-group 'NET-MANAGEMENT'
set firewall name WAN-LOCAL rule 1110 state new 'enable'

set firewall name LAN-LOCAL default-action 'drop'
set firewall name LAN-LOCAL rule 1000 action 'drop'
set firewall name LAN-LOCAL rule 1000 source group network-group 'NET-BLACKLISTED'
set firewall name LAN-LOCAL rule 1010 action 'accept'
set firewall name LAN-LOCAL rule 1010 state established 'enable'
set firewall name LAN-LOCAL rule 1010 state related 'enable'
set firewall name LAN-LOCAL rule 1011 action 'drop'
set firewall name LAN-LOCAL rule 1011 state invalid 'enable'
set firewall name LAN-LOCAL rule 1020 action 'accept'
set firewall name LAN-LOCAL rule 1020 icmp type-name 'echo-request'
set firewall name LAN-LOCAL rule 1020 protocol 'icmp'
set firewall name LAN-LOCAL rule 1020 state new 'enable'
set firewall name LAN-LOCAL rule 1030 action 'accept'
set firewall name LAN-LOCAL rule 1030 destination port '67'
set firewall name LAN-LOCAL rule 1030 protocol 'udp'
set firewall name LAN-LOCAL rule 1030 state new 'enable'
set firewall name LAN-LOCAL rule 1040 action 'accept'
set firewall name LAN-LOCAL rule 1040 destination port '53'
set firewall name LAN-LOCAL rule 1040 protocol 'tcp_udp'
set firewall name LAN-LOCAL rule 1040 state new 'enable'
set firewall name LAN-LOCAL rule 1050 action 'accept'
set firewall name LAN-LOCAL rule 1050 destination port '123'
set firewall name LAN-LOCAL rule 1050 protocol 'udp'
set firewall name LAN-LOCAL rule 1050 state new 'enable'
set firewall name LAN-LOCAL rule 1100 action 'drop'
set firewall name LAN-LOCAL rule 1100 destination port '22'
set firewall name LAN-LOCAL rule 1100 protocol 'tcp'
set firewall name LAN-LOCAL rule 1100 recent count '4'
set firewall name LAN-LOCAL rule 1100 recent time '60'
set firewall name LAN-LOCAL rule 1100 source group network-group 'NET-MANAGEMENT'
set firewall name LAN-LOCAL rule 1100 state new 'enable'
set firewall name LAN-LOCAL rule 1101 action 'accept'
set firewall name LAN-LOCAL rule 1101 destination port '22'
set firewall name LAN-LOCAL rule 1101 protocol 'tcp'
set firewall name LAN-LOCAL rule 1101 source group network-group 'NET-MANAGEMENT'
set firewall name LAN-LOCAL rule 1101 state new 'enable'
set firewall name LAN-LOCAL rule 1110 action 'accept'
set firewall name LAN-LOCAL rule 1110 destination port '161'
set firewall name LAN-LOCAL rule 1110 protocol 'udp'
set firewall name LAN-LOCAL rule 1110 source group network-group 'NET-MANAGEMENT'
set firewall name LAN-LOCAL rule 1110 state new 'enable'

set firewall name DMZ-LOCAL default-action 'drop'
set firewall name DMZ-LOCAL rule 1000 action 'drop'
set firewall name DMZ-LOCAL rule 1000 source group network-group 'NET-BLACKLISTED'
set firewall name DMZ-LOCAL rule 1010 action 'accept'
set firewall name DMZ-LOCAL rule 1010 state established 'enable'
set firewall name DMZ-LOCAL rule 1010 state related 'enable'
set firewall name DMZ-LOCAL rule 1011 action 'drop'
set firewall name DMZ-LOCAL rule 1011 state invalid 'enable'
set firewall name DMZ-LOCAL rule 1020 action 'accept'
set firewall name DMZ-LOCAL rule 1020 icmp type-name 'echo-request'
set firewall name DMZ-LOCAL rule 1020 protocol 'icmp'
set firewall name DMZ-LOCAL rule 1020 state new 'enable'
set firewall name DMZ-LOCAL rule 1030 action 'accept'
set firewall name DMZ-LOCAL rule 1030 destination port '67'
set firewall name DMZ-LOCAL rule 1030 protocol 'udp'
set firewall name DMZ-LOCAL rule 1030 state new 'enable'
set firewall name DMZ-LOCAL rule 1040 action 'accept'
set firewall name DMZ-LOCAL rule 1040 destination port '53'
set firewall name DMZ-LOCAL rule 1040 protocol 'tcp_udp'
set firewall name DMZ-LOCAL rule 1040 state new 'enable'
set firewall name DMZ-LOCAL rule 1050 action 'accept'
set firewall name DMZ-LOCAL rule 1050 destination port '123'
set firewall name DMZ-LOCAL rule 1050 protocol 'udp'
set firewall name DMZ-LOCAL rule 1050 state new 'enable'
set firewall name DMZ-LOCAL rule 1100 action 'drop'
set firewall name DMZ-LOCAL rule 1100 destination port '22'
set firewall name DMZ-LOCAL rule 1100 protocol 'tcp'
set firewall name DMZ-LOCAL rule 1100 recent count '4'
set firewall name DMZ-LOCAL rule 1100 recent time '60'
set firewall name DMZ-LOCAL rule 1100 source group network-group 'NET-MANAGEMENT'
set firewall name DMZ-LOCAL rule 1100 state new 'enable'
set firewall name DMZ-LOCAL rule 1101 action 'accept'
set firewall name DMZ-LOCAL rule 1101 destination port '22'
set firewall name DMZ-LOCAL rule 1101 protocol 'tcp'
set firewall name DMZ-LOCAL rule 1101 source group network-group 'NET-MANAGEMENT'
set firewall name DMZ-LOCAL rule 1101 state new 'enable'
set firewall name DMZ-LOCAL rule 1110 action 'accept'
set firewall name DMZ-LOCAL rule 1110 destination port '161'
set firewall name DMZ-LOCAL rule 1110 protocol 'udp'
set firewall name DMZ-LOCAL rule 1110 source group network-group 'NET-MANAGEMENT'
set firewall name DMZ-LOCAL rule 1110 state new 'enable'
```

The above policy is similar for each interface:

1. Drop blacklisted traffic
2. State Policy (accept established,related; drop invalid)
3. Accept ICMP echo-request (ping)
4. Accept DHCP request
5. Accept DNS requests
6. Accept NTP requests
7. Limit SSH connections to 3 per minute per IP address and accept SSH from management networks
8. Accept SNMP from management networks

The difference for the WAN interface is that we exclude exceptions to permit DHCP, DNS, and NTP traffic since we only provide these services to the LAN and DMZ.

Next we apply the policy to each interface:

```
set interfaces ethernet eth0 firewall local name 'WAN-LOCAL'
set interfaces ethernet eth1 firewall local name 'LAN-LOCAL'
set interfaces ethernet eth2 firewall local name 'DMZ-LOCAL'
```

At this point the firewall itself is secure and allows management access from 10.1.0.10 and 10.1.0.11 (defined in our NET-MANAGEMENT network group object) and the 1.1.1.1 address is blacklisted to the firewall.

We still need to filter traffic between networks, however. For filtering traffic through the firewall it's recommended to filter in the "in" and "out" direction for local networks (e.g. LAN and DMZ) but not filter on the uplink interface (WAN). A common mistake is to filter on both the WAN and LAN interface which creates redundant policy. As the need for additional internal interfaces grow over time so does the size and complexity of a WAN policy.

First we'll create the default "incoming" policy. This will match outgoing traffic from the client perspective. For our LAN we will permit all outgoing traffic by default. For our DMZ we will deny all outgoing traffic by default.

```
set firewall name LAN-IN default-action 'drop'
set firewall name LAN-IN rule 1000 action 'drop'
set firewall name LAN-IN rule 1000 source group network-group 'NET-BLACKLISTED'
set firewall name LAN-IN rule 1001 action 'drop'
set firewall name LAN-IN rule 1001 destination group network-group 'NET-BLACKLISTED'
set firewall name LAN-IN rule 1010 action 'accept'
set firewall name LAN-IN rule 1010 state established 'enable'
set firewall name LAN-IN rule 1010 state related 'enable'
set firewall name LAN-IN rule 1011 action 'drop'
set firewall name LAN-IN rule 1011 state invalid 'enable'
set firewall name LAN-IN rule 9000 action 'accept'
set firewall name LAN-IN rule 9000 source group network-group 'NET-LAN'
set firewall name LAN-IN rule 9000 state new 'enable'

set firewall name DMZ-IN default-action 'drop'
set firewall name DMZ-IN rule 1000 action 'drop'
set firewall name DMZ-IN rule 1000 source group network-group 'NET-BLACKLISTED'
set firewall name DMZ-IN rule 1001 action 'drop'
set firewall name DMZ-IN rule 1001 destination group network-group 'NET-BLACKLISTED'
set firewall name DMZ-IN rule 1010 action 'accept'
set firewall name DMZ-IN rule 1010 state established 'enable'
set firewall name DMZ-IN rule 1010 state related 'enable'
set firewall name DMZ-IN rule 1011 action 'drop'
set firewall name DMZ-IN rule 1011 state invalid 'enable'
```

Note that we drop blacklisted traffic in both directions (as a source and as a destination). This allows us to use the blacklist group object to filter internal systems as well as external systems.

For the DMZ policy we simply omit the rule to permit traffic sourced from the local network.

Apply these policies to their interfaces:

```
set interfaces ethernet eth1 firewall in name 'LAN-IN'
set interfaces ethernet eth2 firewall in name 'DMZ-IN'
```

Next we'll look to create the "outgoing" policy for the LAN and DMZ. This will be inbound traffic from the client perspective. Here we will create a policy to permit all traffic to the DMZ from the LAN network as well as a rule to permit traffic to a web server at 203.0.113.100.

```
set firewall name LAN-OUT default-action 'drop'
set firewall name LAN-OUT rule 1000 action 'drop'
set firewall name LAN-OUT rule 1000 source group network-group 'NET-BLACKLISTED'
set firewall name LAN-OUT rule 1001 action 'drop'
set firewall name LAN-OUT rule 1001 destination group network-group 'NET-BLACKLISTED'
set firewall name LAN-OUT rule 1010 action 'accept'
set firewall name LAN-OUT rule 1010 state established 'enable'
set firewall name LAN-OUT rule 1010 state related 'enable'
set firewall name LAN-OUT rule 1011 action 'drop'
set firewall name LAN-OUT rule 1011 state invalid 'enable'
set firewall name LAN-OUT rule 1020 action 'accept'
set firewall name LAN-OUT rule 1020 icmp type-name 'echo-request'
set firewall name LAN-OUT rule 1020 protocol 'icmp'
set firewall name LAN-OUT rule 1020 state new 'enable'

set firewall name DMZ-OUT default-action 'drop'
set firewall name DMZ-OUT rule 1000 action 'drop'
set firewall name DMZ-OUT rule 1000 source group network-group 'NET-BLACKLISTED'
set firewall name DMZ-OUT rule 1001 action 'drop'
set firewall name DMZ-OUT rule 1001 destination group network-group 'NET-BLACKLISTED'
set firewall name DMZ-OUT rule 1010 action 'accept'
set firewall name DMZ-OUT rule 1010 state established 'enable'
set firewall name DMZ-OUT rule 1010 state related 'enable'
set firewall name DMZ-OUT rule 1011 action 'drop'
set firewall name DMZ-OUT rule 1011 state invalid 'enable'
set firewall name DMZ-OUT rule 1020 action 'accept'
set firewall name DMZ-OUT rule 1020 icmp type-name 'echo-request'
set firewall name DMZ-OUT rule 1020 protocol 'icmp'
set firewall name DMZ-OUT rule 1020 state new 'enable'
set firewall name DMZ-OUT rule 1100 action 'accept'
set firewall name DMZ-OUT rule 1100 source group network-group 'NET-LAN'
set firewall name DMZ-OUT rule 1100 state new 'enable'
set firewall name DMZ-OUT rule 4000 action 'accept'
set firewall name DMZ-OUT rule 4000 destination address '203.0.113.100'
set firewall name DMZ-OUT rule 4000 destination port '80,443'
set firewall name DMZ-OUT rule 4000 protocol 'tcp'
set firewall name DMZ-OUT rule 4000 state new 'enable'
```

Finally we apply these to their interfaces:

```
set interfaces ethernet eth1 firewall out name 'LAN-OUT'
set interfaces ethernet eth2 firewall out name 'DMZ-OUT'
```

Traffic filtering is now in place. For better accounting, there are some logging options that can be enabled. The first is to log configuration changes (recommended):

```
set firewall config-trap 'enable'
```

The next option is to log traffic that reaches the default rule of a named policy (in our case the default drop). Take care as this can generate quite a bit of logging. A remote syslog server is strongly encouraged.

```
set firewall name LAN-IN 'enable-default-log'
set firewall name LAN-OUT 'enable-default-log'
set firewall name DMZ-IN 'enable-default-log'
set firewall name DMZ-OUT 'enable-default-log'
```

For any individual rule, logging can also be enabled. In this example we want to log traffic dropped due to an invalid conntrack state:

```
set firewall name LAN-IN rule 1011 log 'enable'
set firewall name LAN-OUT rule 1011 log 'enable'
set firewall name DMZ-IN rule 1011 log 'enable'
set firewall name DMZ-OUT rule 1011 log 'enable'
```

Finally, the following global configuration defaults are recommended:

```
set firewall all-ping 'enable'
set firewall broadcast-ping 'disable'
set firewall ipv6-receive-redirects 'disable'
set firewall ipv6-src-route 'disable'
set firewall ip-src-route 'disable'
set firewall log-martians 'enable'
set firewall receive-redirects 'disable'
set firewall send-redirects 'disable'
set firewall source-validation 'disable'
set firewall syn-cookies 'enable'
```

The last step is to configure NAT, in this case we will NAT LAN traffic to the IP of our WAN interface.

```
set nat source rule 1000 outbound-interface 'eth0'
set nat source rule 1000 source address '10.1.0.0/24'
set nat source rule 1000 translation address '198.51.100.6'
```

Note that more detailed NAT configuration (including port-forwards or 1:1 NAT) is beyond the scope of this guide.

Part 3: High Availability and Scaling Considerations

In the previous section we discussed how to configure VyOS as a firewall. In this section we will build on the previous configuration to configure a pair of VyOS firewalls for High Availability.

VyOS supports two methods of HA, Clustering and Manual VRRP. Clustering was a relatively new feature introduced in Vyatta before the VyOS project was started and as a result is not yet feature complete. Most noticeably a "configuration-sync" mechanism is absent. As a result, the current preferred method of HA in VyOS is the use of VRRP and conntrack-sync (connection state table sharing), and will be the method covered by this guide.

Assuming you have a pair of VyOS systems, each with 4 ethernet ports, and the configuration used for the previous example, our first step will be to establish a peer-link to the secondary unit.

The peer-link only exists between VyOS systems and should be a direct connection. As such, we will use a bogon IP address for the link network of 192.0.0.1/30 for the primary, and 192.0.0.2/30 for the secondary as to avoid addressing conflicts with RFC 1918private address space.

Note that use of this addressing is in violation of RFC 5736 and currently allocated by RFC 7335 and may not be suitable for your environment. Alternatively any RFC 1918 prefix can be used.

On the primary:

```
set interfaces ethernet eth3 description 'PEER-LINK'
set interfaces ethernet eth3 address '192.0.0.1/30'
```

Secondary:

```
set interfaces ethernet eth3 description 'PEER-LINK'
set interfaces ethernet eth3 address '192.0.0.2/30'
```

Next we will configure the conntrack-sync service to make use of this link to share the connection state table. This will provide the secondary firewall with the ability to know if traffic was previously established or not in the event of failover.

On both Primary and Secondary:

```
set service conntrack-sync event-listen-queue-size '8'
set service conntrack-sync failover-mechanism vrrp sync-group 'CLUSTER-1'
set service conntrack-sync interface 'eth3'
set service conntrack-sync mcast-group '225.0.0.50'
set service conntrack-sync sync-queue-size '1'
```

Finally, we will re-configure each network interface (WAN, LAN, DMZ) to use VRRP for failover. Note that VRRP will require 3 IP addresses per network; one for each firewall and a shared "virtual" IP.

On the Primary:

```
set interfaces ethernet eth0 address '198.51.100.4/29'
set interfaces ethernet eth0 vrrp vrrp-group 10 advertise-interval '1'
set interfaces ethernet eth0 vrrp vrrp-group 10 preempt 'true'
set interfaces ethernet eth0 vrrp vrrp-group 10 preempt-delay '300'
set interfaces ethernet eth0 vrrp vrrp-group 10 priority '128'
set interfaces ethernet eth0 vrrp vrrp-group 10 'rfc3768-compatibility'
set interfaces ethernet eth0 vrrp vrrp-group 10 sync-group 'CLUSTER-1'
set interfaces ethernet eth0 vrrp vrrp-group 10 virtual-address '198.51.100.6/29'

set interfaces ethernet eth1 address '10.1.0.2/24'
set interfaces ethernet eth1 vrrp vrrp-group 11 advertise-interval '1'
set interfaces ethernet eth1 vrrp vrrp-group 11 preempt 'true'
set interfaces ethernet eth1 vrrp vrrp-group 11 preempt-delay '300'
set interfaces ethernet eth1 vrrp vrrp-group 11 priority '128'
set interfaces ethernet eth1 vrrp vrrp-group 11 'rfc3768-compatibility'
set interfaces ethernet eth1 vrrp vrrp-group 11 sync-group 'CLUSTER-1'
set interfaces ethernet eth1 vrrp vrrp-group 11 virtual-address '10.1.0.1/24'

set interfaces ethernet eth2 address '203.0.113.2/24'
set interfaces ethernet eth2 vrrp vrrp-group 12 advertise-interval '1'
set interfaces ethernet eth2 vrrp vrrp-group 12 preempt 'true'
set interfaces ethernet eth2 vrrp vrrp-group 12 preempt-delay '300'
set interfaces ethernet eth2 vrrp vrrp-group 12 priority '128'
set interfaces ethernet eth2 vrrp vrrp-group 12 'rfc3768-compatibility'
set interfaces ethernet eth2 vrrp vrrp-group 12 sync-group 'CLUSTER-1'
set interfaces ethernet eth2 vrrp vrrp-group 12 virtual-address '203.0.113.1/24'
```

On the Secondary:

```
set interfaces ethernet eth0 address '198.51.100.5/29'
set interfaces ethernet eth0 vrrp vrrp-group 10 advertise-interval '1'
set interfaces ethernet eth0 vrrp vrrp-group 10 preempt 'true'
set interfaces ethernet eth0 vrrp vrrp-group 10 preempt-delay '300'
set interfaces ethernet eth0 vrrp vrrp-group 10 priority '64'
set interfaces ethernet eth0 vrrp vrrp-group 10 'rfc3768-compatibility'
set interfaces ethernet eth0 vrrp vrrp-group 10 sync-group 'CLUSTER-1'
set interfaces ethernet eth0 vrrp vrrp-group 10 virtual-address '198.51.100.6/29'

set interfaces ethernet eth1 address '10.1.0.3/24'
set interfaces ethernet eth1 vrrp vrrp-group 11 advertise-interval '1'
set interfaces ethernet eth1 vrrp vrrp-group 11 preempt 'true'
set interfaces ethernet eth1 vrrp vrrp-group 11 preempt-delay '300'
set interfaces ethernet eth1 vrrp vrrp-group 11 priority '64'
set interfaces ethernet eth1 vrrp vrrp-group 11 'rfc3768-compatibility'
set interfaces ethernet eth1 vrrp vrrp-group 11 sync-group 'CLUSTER-1'
set interfaces ethernet eth1 vrrp vrrp-group 11 virtual-address '10.1.0.1/24'
set interfaces ethernet eth2 address '203.0.113.3/24'
set interfaces ethernet eth2 vrrp vrrp-group 12 advertise-interval '1'
set interfaces ethernet eth2 vrrp vrrp-group 12 preempt 'true'
set interfaces ethernet eth2 vrrp vrrp-group 12 preempt-delay '300'
set interfaces ethernet eth2 vrrp vrrp-group 12 priority '64'
set interfaces ethernet eth2 vrrp vrrp-group 12 'rfc3768-compatibility'
set interfaces ethernet eth2 vrrp vrrp-group 12 sync-group 'CLUSTER-1'
set interfaces ethernet eth2 vrrp vrrp-group 12 virtual-address '203.0.113.1/24'
```

The configuration above will delay recovery of the primary by 5 minutes (300 seconds) to allow for re-building of the conntrack state table.

For larger systems you may want to configure the system ARP table to support more address space.

```
set system ip arp table-size '32768'
```

You should also consider configuration of conntrack if the system will be responsible for a large number of flows.

```
set system conntrack expect-table-size '8192'
set system conntrack hash-size '131072'
set system conntrack table-size '1048576'
```

Finally, you may find it helpful to disable certain conntrack helper modules known to be problematic:

```
set system conntrack modules nfs 'disable'
set system conntrack modules sip 'disable'
set system conntrack modules sqlnet 'disable'
```

At this point you should have a highly available pair, allowing for in-service upgrades and maintenance. Take care to make sure you replicate firewall configuration on each firewall so that they're consistent in the event of failover.