Network Firewall Design Guide

Open Source Firewall Design, Configuration, and Operation with VyOS

Prepared By:

Ray Soucy
Senior Cyber Security Engineer
University of Maine System

Last Updated: 2016-05-05 for VyOS release **1.1.7** DRAFT

This page intentionally left blank.

Contents

This page intentionally left blank.

Introduction

This guide covers the use of VyOS as a network firewall. For more information on VyOS, visit the VyOS project website at vyos.net.

VyOS is a community-driven Linux-based network operating system which provides network routing, VPN, and firewall functionality.

When learning VyOS it is important to keep in mind that while built on a foundation of Linux and other Free and Open Source Software (FOSS) projects, such as Quagga, VyOS provides its own shell, installation, and configuration management system, affecting all aspects of installation, configuration, and operation of the system. Attempting to administering the system as a traditional Linux server will almost always result in operational problems and indeterministic system state, and is the most common problem that new users encounter.

It is best to approach VyOS as you would a new network router or firewall with its own CLI and not as a Linux server.

Installation

VyOS uses an image-based install process. This allows multiple releases of VyOS to coexist on a system, allowing for non-destructive upgrades and the ability to revert changes to the OS. Each system image maintains a unique copy of its configuration repository, and the image to use at boot can be set via the CLI or manually selected on the system console at boot. Note that VyOS is not a true in-memory image, and changes to the image filesystem will be preserved upon reboot.

Configuration

VyOS implements a unified, human-readable, configuration file for all system configuration. To achieve this, VyOS makes use of a custom shell which allows for system configuration and operation, and a custom initialization process which generates Linux system and service-specific configuration files programmatically upon service start. This allows VyOS configuration to be backed up as a single file, which can easily be revision controlled, and also allows for a VyOS system to be quickly replicated to improve time to service restoration in the event of hardware failure, or to facilitate large deployments through the use of configuration templates. It also has the consequence of removing any configuration directives manually defined by the user outside the configuration system.

The VyOS configuration system also implements two-stage configuration, allowing changes to be queued before applying them or choosing to discard them, and internal revision control allowing comparison or rollback against previous configurations. Note that at this time, rollback requires a system reboot, which is a major design flaw left over from Vyatta that the VyOS project is working to address in future releases.

Operation

The custom command shell implemented for VyOS provides in-line help and command completion and provides a user interface that will be familiar to network and security engineers who have used proprietary network router or firewall solutions.

Combined, the design of VyOS addresses some of the most common problems with the use of Linux as a network operating system:

- The ability to maintain a predictable system state
- The ability to quickly replicate a system in the event of failure
- The ability to deploy and maintain a large number of systems using consistent configuration practices
- The ability to rollback changes or software upgrades in the event of a problem

Viability

A common concern with the use of VyOS as a network router or firewall is that it is software-based and lacks the performance necessary to be competitive with proprietary hardware solutions. While this can be true for high-performance carrier networks, using connections up to 40 or 100 Gigabit, VyOS on modern commodity hardware is more than competitive for multi- Gigabit throughput, often out-performing commercial solutions advertised for these same levels of throughput.

VyOS on modest hardware can approach 10 Gigabit levels of performance for large packets, and maintain multi-Gigabit performance for small packets.

As an example, the following test results show VyOS performance using an Intel i7-3770 with 8 x Intel I350-T2 Gigabit Ethernet interfaces (4 x dual-port cards):

Packet Length (bytes)	**Packets Per Second**	**Throughput (kbps)**
64	4017857.1480	2057142.8600
128	3885135.1140	3978378.3570
256	3079710.1400	6407246.3670
512 *	1879699.2800	7699248.2510
1024	957854.4000	7846743.2450
1518	650195.0400	7895968.5660

* 512 byte packet length is comparable to real world traffic mix.

Here we can see for 512 byte packets and larger, VyOS on commodity hardware is able to achieve near- line-rate performance, and at smaller packet lengths is a 4M+ PPS solution.

VyOS also provides a high level of system stability, being based on Linux, and also supports high-availability configurations using VRRP and connection tracking state synchronization.

In terms of security, VyOS builds on the most peer-reviewed packet filter in the world, the Linux netfilter project, and being based on Debian Linux, leverages the work of the Debian Security Team to track and patch system vulnerabilities.

As an example use case, within the University of Maine System VyOS has been in production since release May of 2014, details on this experience can be found at the end of this document in Appendix B.

Overall, VyOS provides a mature, stable, and and maintainable platform for use as a network firewall. It has a growing community of users and developers, and has consistently provided timely releases to address security patches and bug fixes. It’s important to understand the limitations of VyOS in the area of web filtering, application awareness, SSL inspection, and threat detection commonly found in commercial all-in-one solutions. VyOS provides only a stateful network firewall. The other significant limitation for VyOS is the lack of documentation and commercial support. This may be a deal breaker for many organizations, but those with staff able to review source code and participate in the community, VyOS can be a very attractive option.

This page intentionally left blank.

Design Concepts

The term **Firewall** can be used to describe a number of different network filtering methods. Generally, these are broken into the following categories:

- Access Control Lists
- Stateful Packet Inspection
- Deep Packet Inspection and Application-Specific

VyOS implements a **Stateful Packet Inspection** firewall, the category most often associated with the term, but can also be used to implement **Access Control Lists**.

Common to each method, is the concept of **5-tuple** matching. The 5-tuple represents the association of the following data of an IP network packet:

1. Protocol
2. Source Address
3. Destination Address
4. Source Port
5. Destination Port

For example, an HTTP connection may be described as: protocol TCP, source address 10.1.0.100, destination address 10.2.0.200, source port random, and destination port 80.

Take note that the source port is a random value in normal communication. This random source port is referred to as the **ephemeral port**. Because a client may need to create multiple network connections, when an operating system needs to open a new connection, it randomly selects an unused source port from a range of valid ephemeral port numbers. This range is often different depending on the operating system, but many have adopted the standard range of 49,152 to 65,535 (65,535 being the maximum value of a 16-bit port number, or 2^{16}).

Access Control Lists

Because there is no method to know the source port a client will use in advance, the random source port problem is typically avoided in an access control list by only matching the destination port. To permit our HTTP packet, we might match: protocol TCP, source address 10.1.0.100, destination address 10.2.0.200, source port any, and destination port 80.

We could go a step further and attempt to manage a source port in the official ephemeral port range defined by IANA of 49,152 through 65,535 but because this range is operating system dependent, it would break devices such as Windows XP and earlier systems which use 1,024 through 5,000, or Linux systems which typically use 32,768 through 61,000 but can be easily configured by the system administrator to be any range of port numbers.

An Access Control List, or ACL, is basic form of filtering that evaluates only the 5-tuple of a connection, and often permits any source port, reducing the match criteria to 4. The advantage of an access control list is that it can be implemented purely in hardware because of its simplicity, which means predictable performance and low latency.

There is little advantage in implementing a stateless ACL on VyOS as opposed to a stateful firewall as both are implemented in software; however, the hardware ACL plays an important role in its ability to filter traffic before reaching a stateful firewall: to help mitigate denial of service attacks which may exhaust resources for more complex software-based filtering, or provide low-latency filtering for latency-sensitive traffic such as network storage. A network switch capable of implementing hardware ACLs is strongly recommended in any security design, even when a stateful firewall is employed.

Stateful Packet Inspection

The Stateful Packet Inspection firewall builds on the ACL by tracking connection state.

This should not be confused with TCP state. While a modern SPI firewall, like VyOS, does enforce TCP state, connection tracking is maintained by the firewall itself and not flags within the packet. This is an important distinction, and necessary to avoid malicious packets with falsified state flags to be inadvertently permitted.

In terms of connection tracking in Linux, each connection is identified as:

- NEW
- RELATED
- ESTABLISHED
- INVALID

Internally, there are also transitionary states used within the system:

- CLOSED
- UNREPLIED
- ASSURED

This connection state information added can be thought of as a “6th tuple” for matching traffic; this metadata exists only within the memory of the firewall itself and not the packet, however, and is derived through observation, or inspection, of network packet headers flowing through the firewall.

SPI solves the random source port problem found with access control lists in allowing any source port only when matching NEW connections and placing them in an UNREPLIED state. After bidirectional traffic is observed (e.g. the server response), the traffic is marked as ASSURED and can be matched with an ESTABLISHED state. This matching of the full 5-tuple and connection state enforces that traffic which does not match a NEW state, for example in the case of TCP having appropriate TCP state flags, is not inadvertently permitted, and that a different source using the same source port is not seen as the same connection.

The RELATED state is a special state similar to ESTABLISHED which is used to match additional traffic which does not match the basic 5-tuple of an existing connection but is expected by the firewall due the protocol being used. The majority of traffic which falls into the RELATED state category is attributed to protocols which call for client and server traffic to use different port numbers, such as FTP, or expected ICMP error messages from a host with an ESTABLISHED connection. This matching is generally implemented through the use of optional connection tracking “helper” modules; the most popular example of a protocol which requires a helper module is FTP, which uses different ports for data transfer and control.

Once a connection is marked as CLOSED (either through observing the TCP handshake, or a timeout for the connection) it will be cleared from the connection tracking table and the client will need to begin with a NEW connection to re-establish communication.

Finally, the INVALID state is used to identify traffic which may match the 5-tuple but violates the rules of TCP, for example, attempting to send traffic after a connection has been marked closed.

In application, the basic model used for defining policy using an SPI firewall is:

- DROP traffic marked INVALID
- ACCEPT traffic marked as ESTABLISHED or RELATED
- ACCEPT traffic marked as NEW for a specific resource (e.g. TCP port 80 to 10.2.0.200 from 10.1.0.100, or connections initiated from a trusted system)
- DROP everything else

With most firewalls, traffic is evaluated against rules sequentially until there is a match and an action performed (generally permit or deny). Because of this, the matching of INVALID, ESTABLISHED, and RELATED traffic is performed early on in the policy to avoid additional processing for the majority of traffic and improve efficiency. It is useful, however, to maintain the ability to insert rules to drop malicious traffic before this state check, so that established connections can be filtered. We will demonstrate this in our example configuration.

With Linux, there are multiple methods to deny traffic. The most common being DROP or REJECT. REJECT is used to respond with an ICMP error message and should be avoided for externally-facing policy as it can be leveraged in denial of service attacks, it can be useful for internal filtering to provide a troubleshooting tool to system administrators. DROP will silently discard the packet, which is appropriate for the majority of network filtering.

The key advantage of the SPI firewall is that it allows traffic to be described in simple terms of 4-tuple matching, like an ACL, but solves the challenge of the 5th tuple dynamically and mitigates packet header manipulation attacks. For state-oriented protocols such as TCP, an SPI firewall can enforce the transition of connection state for the protocol, while for stateless protocols, such as UDP or ICMP, internal timers are used to determine when a connection has ended.

For better understanding, consider the following example. Since we know that our web server is at 10.2.0.200 and that HTTP uses TCP port 80, we are able to create a rule to permit any NEW traffic matching that description to the server, but have the firewall then enforce the source address, source port, and connection state, all of which are unkown to us until a connection is attempted.

As a final note, the most important filtering component that SPI adds is a sense of traffic **direction**. This is achieved by identifying and distinguishing the **traffic initiator**, typically by knowing the designated port used for specific services.

This concept of direction will be exploited throughout this design guide.

Deep Packet Inspection and Intrusion Detection

DPI generally builds on SPI with the additional step of matching the packet payload or contents, rather than simply matching the header. This can be as basic as signature matching, or as advanced as enforcing application protocols (such as knowing and enforcing the correct sequence of commands) and visibility of encrypted data (commonly referred to as SSL inspection or protocol decryption).

Because DPI is extremely CPU-intensive and complex enough to make hardware implementation impractical, DPI at scale is generally achieved through the use of an external system which performs passive analysis of a copy of network traffic, either replicated by a device, or using a passive tap to duplicate the signal.

While VyOS provides functionality to mirror network traffic to an external device, such as an IDS, it is recommended that this function is performed using either a hardware switch, or passive tap outside the firewall to avoid the introduction of network bottlenecks when possible as this replication is done in software.

As an aside, it should be noted that while many commercially available security appliances include IDS and DPI functionality, that enabling these typically introduces significant performance degradation and can be exploited as a denial of service attack vector.   Because of this it is important to be aware of expected performance with all functionality enabled when comparing solutions. For SMB applications, using an all-in-one solution can be attractive in providing a single pane of management, but at scale it is far more common to use discrete infrastructure components for each task (e.g. firewall vs. VPN vs. IDS vs. content filtering). This is not only to maintain separation of concerns, but also typically necessary to maintain expected network performance.

Network Segmentation

A common mistake when deploying network firewalls is to place all internal systems on a single network and depend upon the firewall to filter outside traffic only.

Because the use of SPI firewalls has become standard practice, attackers now expect this and seek to compromise systems from the inside out to then use as a pivot point which can be used to access other internal systems directly. The most common attack vector today is the web browser, and getting an end-user to execute malicious code on an internal system (actively or passively through an exploit), rather than remotely attempting to penetrate a firewall.

Because of this, the need for segmentation of internal systems is critical in the ability to reduce the attack surface of an organization.

Today there are two levels of segmentation commonly employed:

- VLAN (Virtual LAN)
- Security Zone

A **VLAN**, or virtual LAN, can be thought of as a virtual network switch. Each VLAN maintains its own list of ports and isolates traffic such that traffic from one VLAN can not be accessed by another VLAN without the use of a network router, or in our case, a firewall acting as the gateway. Typically, each VLAN is associated with an individual IP network. VLANs replace the need for multiple physical network switches to achieve the goal of network isolation.

A **Security Zone** is an abstraction that exists for the sake of network security policy and can represent a group of host systems, networks, or even users and applications with emerging technology.

In most network designs each VLAN is mapped to a security zone in a 1:1 relationship; however it is worth noting that a single security zone can make use of multiple VLANs, in the event that VLAN segmentation is necessary for network scaling (e.g. limiting a network to 500 hosts to reduce broadcast traffic).

For the sake of this design guide, we will be using a mapping of 1:1, and the use of the term zone and VLAN can be used interchangeably.

The process of abstracting networks as members of security zones, and assigning policy based on relationships between zones, is commonly referred to as a **Zone-Based Firewall**.

Note that VyOS supports both interface-based and zone-based firewall configuration methods, and that a zone-based design can be implemented using interface-based configuration.

Some practical examples of network segmentation include separate zones for:

- Clients
- Internal Servers
- External Servers
- Wireless
- Guest Wireless
- Printers and Projectors
- Phones and Teleconferencing
- Building Sensors and Controls

For example, you may wish to put policy in place that permits clients, servers, and wireless to initiate communication to network printers, but not allow printers to initiate connections to any of those respective networks. You might also want to limit guest wireless from communicating with any other internal network and only provide them with external access.

Depending on the scale and complexity of the network, there may even be several zones representing client systems (e.g. isolation between departments).

Each zone represents a border and the opportunity to control network traffic through filtering. It should also be noted that each zone can exist on a single firewall, or independent firewalls and the use of multiple firewalls is an emerging trend in maintaining separation of concerns and reducing performance bottlenecks through dedicating resources for high traffic networks.

Also note that for very simple inter-zone relationships (such as forbid zone A and B from talking with each other), a hardware ACL on a network switch may be more effective than a firewall if performance is a concern and stateful inspection is not necessary.

Private Addressing

A series of IP address ranges have been designated to be non-routed and available for internal use by RFC 1918. These are commonly referred to by **private addressing** or sometimes **1918 space**.

The following networks are designated:

Network	**Mask**	**Prefix Length**	**First Address**	**Last Address**
10.0.0.0	255.0.0.0	8	10.0.0.1	10.255.255.254
172.16.0.0	255.240.0.0	12	172.16.0.1	172.31.255.254
192.168.0.0	255.255.0.0	16	192.168.0.1	192.168.255.254

The term non-routed is with respect to the Internet. In practice, these addresses should never be reachable over an Internet connection. Internet Service Providers typically enforce this by **Null Routing** and other forms of filtering such that you should never see a connection from the Internet sourced using these networks, and you should never be able to reach another system on the Internet if you source your traffic from them.

Historically, systems which did not require direct Internet access were assigned private addresses. In order to reach other systems on the Internet they would make use of a proxy server which existed within a DMZ or demilitarized zone which used public (routed) addressing. Today, this practice is less common and has instead been replaced with NAT and the use of stateful firewalls.

Network Address Translation

NAT, or Network Address Translation, is a tool often mistaken as a security measure. There are several methods of NAT, but the one most commonly understood is more accurately called **PAT** or Port Address Translation. For all intents and purposes this guide, and many others, will use NAT and PAT interchangeably. PAT is a method for using a single IP address to provide network access to many hosts. This requires a network device that is able to keep track of connections and their state, and rewrite IP packet headers as they traverse a network border in both directions.

A consequence of PAT is that state tracking is implemented, and that no unsolicited traffic to an IP address used for PAT is accepted since the router has no means of knowing which of many internal addresses the traffic would be destined to.

In other words:

A NAT router requires a firewall, but a firewall does not require NAT. This is an important distinction, and it should be noted that there is no security benefit to using NAT over a SPI firewall without NAT.

Because IP addresses are a limited resource, many Internet Service Providers have limited customers to a single address. This created a necessity for NAT, and the side effect of a simplistic stateful packet inspection firewall being a standard component of modern connectivity.

Note that this guide makes the distinction between NAT and a firewall to underscore the differences between securing IPv4 and IPv6, the next generation of Internet Protocol, which does not use NAT, but rather provides enough addressing for any customer to maintain a virtually unlimited number of devices per network.

For most implementations NAT is configured using two rule types:

- Source NAT (or SNAT)
- Destination NAT (or DNAT)

Source NAT is used to translate the source address of a client (initiator) to a different IP address (typically private to public).

Destination NAT is used to translate the destination address from a client (initiator) to a different IP address (typically public to private). When restricted to a specific port and protocol, this is commonly referred to as a **Port-Forward**.

When rules for both SNAT and DNAT are used together for a direct mapping of a single external address to a single internal address, this is commonly referred to as **1:1 NAT**.

In a typical scenario where a network has a single public IP address, DNAT is commonly used in the form of port-forward to direct incoming traffic for a specific service (such as HTTP) to an internal (private) system. Multiple port-forwards can be used on a single address to direct different services such as HTTP and FTP to different internal hosts.

The use of NAT breaks connectivity for applications that may require dynamic incoming connections. Application developers have accommodated for this by forcing all client-to-client traffic through a common server, or "tricking" a stateful firewall into thinking the connection was established from the inside out by creating a traffic for for the desired 5-tuple by exploiting the stateless nature of UDP (referred to as UDP hole punching). This method also requires a server for coordinating discovery of other clients, however. Other efforts to work-around this problem include a protocol by which client systems could request a dynamic port-forward be configured by the firewall, known as Universal Plug-and-Play or its successor NAT-PMP, which works for TCP connections, but required the port number be randomly assigned by the firewall and non-predictable (still requiring a server to coordinate the communication).

Another common issue with the use of NAT is the "split brain" problem: that systems on the same network segment become confused by the translation process if referencing one another using a mapped address. The most common instance of this is hosting a web server using a port-forward, and DNS pointing to the public IP address of the server:

1. The client makes a request to the firewall
2. The firewall translates the destination address to the private address of the server
3. Because the traffic is not leaving the network no translation on the source address is performed
4. The server responds to the unmodified private address of the client, using its private address
5. The client is expecting the response to be sourced from the public address, and discards the traffic

There are two methods to work around this problem, both of them have drawbacks:

1. Configure DNS servers to respond differently to internal systems vs. external systems
2. Implement special rules to NAT traffic between internal host systems

Option 1 is preferred in this scenario, but does require maintaining a DNS server that is configured to support **Split DNS** and is often a function a router can not perform itself due to limited configuration options for DNS services.

Option 2 requires no special configuration, but has the drawback of forcing local traffic through the firewall, as well as making all internal requests appear as they're being sourced by the router. This makes it difficult to troubleshoot problems as there is often no method to distinguish traffic from different internal clients. This option is commonly referred to as **NAT Reflection** or **Hairpin NAT**.

To avoid these problems, the use of NAT should be limited to traffic flowing between the internal and external network only, and not between internal networks.

Demilitarized Zones

A popular design to avoid the need for Split DNS or Hairpin NAT was to establish a three-legged firewall model, which broadly defined 3 security zones of:

- Inside (or Internal)
- Outside (or External)

- DMZ (for Demilitarized Zone)

This design calls for a third network for public-facing servers to be placed on using public IP addressing, but with visibility (through local routing) of private internal systems **without** the use of NAT between the Inside and DMZ. This allows for systems within the DMZ to be accessed from the external network, because public addressing is in use, but also allows systems within the DMZ to access systems on the internal network, such as a database, because the local router is aware of the private networks.

Typically, the DMZ is strengthened with a stateful firewall which has a default policy to allow any internal system to initiate communication to DMZ systems, but disallow DMZ systems to initiate communication with internal systems unless an exception has been made (for example, the database server).

Using this design, there is no need for Split DNS or NAT Reflection, since there is no NAT between internal clients and external servers. The design of the DMZ pre-dates NAT and was originally implemented by having internal client systems on private addressing make use of a proxy server within the DMZ for external connectivity.

Unfortunately, the DMZ model requires an allocation of multiple public IP addresses which is increasingly difficult as less and less IP address space is available.

As an aside, it has become common to misuse the term DMZ to describe a zone dedicated to servers which make use of port-forwards to map external traffic. While they effectively achieve a similar goal, it is a technically incorrect use of the term. The lack of understanding of the proper definition of the DMZ often leads to confusion when attempting to communicate security designs. For the sake of this guide, DMZ will always refer to a public IP network with visibility of private network hosts.

Understanding the use and limitations of NAT, and the distinction of the DMZ is important for understanding security concepts that will be applied for IPv6.

Virtual Private Networks

A Virtual Private Network or VPN is the combination of network tunneling and encryption, often with authentication.

In terms of security architecture, the VPN is an important tool for remote access in that it facilitates the extension of trusted address space (which can be protected against spoofing) outside the network, and can in turn be used to build firewall policy for access to resources that may not be appropriate to open to the Internet at-large.

VPN can be broken down into two categories:

- Site-to-Site VPN
- Client VPN

Site-to-Site VPN is typically built between network routers or firewalls to interconnect private networks at two locations over an untrusted connection. The most common method of site-to-site VPN is IPsec tunneling, typically this is employed using fixed public addressing between two points.

Client VPN solutions are used to provide individual users with secure remote access to private networks from untrusted connections, typically without knowledge of what source addressing will be used.

A common mistake when implementing VPN service is to assume all clients are trusted for full access to the internal network, this has many of the same segmentation problems as clients and servers co-existing.

Instead, security engineers should seek to terminate VPN client traffic in a dedicated zone, and build appropriate firewall policy to limit VPN access to only what is required.

There are many VPN solutions available. Unfortunately traditional protocols commonly associated with VPN connectivity use protocols other than TCP or UDP, or require fixed source port numbers, and either do not function behind NAT, or can only support one connection per public IP when behind NAT.

Because of this, the concept of SSL VPN services has become popular, which make use of standard TCP or UDP protocols and generally work well in NAT environments.

An open source example of this would be OpenVPN. While OpenVPN is supported by VyOS, it’s configuration is beyond the scope of this design guide and building OpenVPN as a dedicated system rather than a service of the firewall is strongly recommended for both separation of concerns and performance reasons.

Example Network

This section provides an overview of the example network that we will reference in the configuration portion of this design guide.

Network Addressing and VLANs

The following networks and VLANs will be used in this example:

VLAN ID	Name	Network	Description
2	net-mgt	10.1.0.0/24	Network Management
100	server	10.1.100.0/24	Servers
101	printer	10.1.102.0/24	Printers
200	access-wired	10.1.200.0/24	Wired Clients
202	access-wireless	10.1.202.0/24	Wireless Clients
204	guest	10.1.204.0/24	Guest Wireless
300	dmz	203.0.113.0/24	DMZ

Network Diagram

Devices, Clients, and Servers

Hostname	IP	Description
firewall.example	--	VyOS Firewall and Router
switch.example	10.1.0.10	Network Switch
ap.example	10.1.0.11	Wireless Access Point
printer.example	10.1.102.10	Network Printer
pc.example	10.1.200.10	Client PC
laptop.example	10.1.202.10	Client Laptop
web-server.example	203.0.113.10	Public Web Server
db-server.example	10.1.100.10	Database Server

VyOS Configuration

Appendix A - Case for Free and Open Source Infrastructure

Appendix B - Experience Running VyOS in Production