Practical OPNsense

Enterprise firewalls build on open source

Markus Stubbig

Contents

Preface

OPNsense started its life as a bitchy little sister of pfSense who wanted to be superior: better code, better security, better licensing, better targets – and even better open source then its siblings!

With these grandstanding words OPNsense separated from pfSense in 2014. The OPNsense developers started with a spring-cleaning of the pfSense source code. They presented the first version of OPNsense at the beginning of 2015: they tidied up all the code and added a modern web GUI without changing the functionality.

After all the effort, did OPNsense actually make the cut and find friends? If so, who are they? As it turns out, well-structured and documented source code, as seen in OPNsense, is apparently a significant attribute for an open-source firewall! And several celebrities from the security world have complimented OPNsense, first and foremost of these being the chief developer of monowall.

Probably every pfSense administrator has taken a brief look at OPNsense and reviewed its differences. The OPNsense web interface appears in a *responsive design*, while the known features from pfSense are accessible only from swiveling menus. This improvement adds to the already positive impression of OPNsense.

This book will show you how to operate OPNsense and the many features which are all possible with this open-source firewall.

Enjoy reading and trying things out – and be ready for wonderful surprises (and even a bit of cursing).

Overview

Part 1, *For Beginners*, sets up the network environment with physical devices or on a virtual platform. All machines get an operating system and a quick configuration, followed by essential functions, like routing and IPv6.

In part 2, *For Intermediates*, the firewalls fulfill some pressing tasks, which must be present in every network. As a packet filter and address translator, they will connect and isolate the attached subnets.

Part 3, *For Experts*, dives into enterprise-grade topics and establishes site-to-site VPN tunnels and firewall clusters for high-availability. An in-depth look inside the data flow provides good-old NetFlow. And the included proxy server can even sniff inside SSL connections.

Outside the closed lab environment OPNsense acts in part 4, *For Hackers*, as DSL router, load-balancer for multiple Internet links and even as Sheriff for data trespassers.

Part 5, *For Admins*, provides many small hints that make daily work with the firewall more fluent and straightforward. After that, OPNsense uploads its configuration file to the cloud and stores it revision-safe on a DropBox or Google Drive. Finally, let’s check out the programming interface of OPNsense.

Resources

https://opnsense.org
The homepage of OPNsense offers a good start into the topic and links to the official documentation, to the forum, and the download area.

https://github.com/opnsense
The source code is hosted at GitHub, where anybody can review the code and its development process. It also offers the build tools and tutorials on how to compile the code yourself.

https://docs.opnsense.org/
OPNsense for reading: manuals for user and developer, how-to documents with many screenshots and step-by-step tutorials. Almost as comprehensive as a full book.

https://forum.opnsense.org/
The forum is the first place to find small tutorials, discussions, and support from the community. The language is not limited to English, and many posts are in German.

Conventions

`Constant Width regular` shows the output of a command.

`Typewriter font` is used for configuration and keywords, and must be typed exactly as shown.

Constant Width bold shows commands that expect some kind of output.

`Accentuations` indicate unique words or lines within a command or its output.

```
a-very-long-command-string --with --many \
  --many "options"
```

Commands with many arguments can take more space than fits into one line. For a clear overview, these commands are printed in several lines indented by two spaces. At the end of the line is a backslash to indicate that the command continues on the next line.

Legal

Introduction

OPNsense is an open-source network operating system for routers and firewalls. It is based on FreeBSD UNIX and contains such applications as Squid, pf, StrongSwan, and OpenVPN with a consistent web interface. OPNsense runs on physical hardware, as a virtual machine, or in the cloud. Although it offers a wide range of functions, it has not yet become a well-known brand. Even though it hits it out of the ballpark for its functionality and usability. OPNsense combines the charm of UNIX with the functional range of a professional firewall at a very low budget.

OPNsense is:

Evolving. And that’s said in a positive light because it means there is room to grow. In addition, implementing features is sometimes out of the ordinary: the provider-centric QinQ-tagging and several DNS services are included, but IPv6 does need focus.

Open Source. The advantage of an open-source solution isn’t always its price. There is no license cost involved, but its use requires time and resources from an IT department to set up; and when it is finally set up, the software may be poorly documented and not provide vendor support.

Thus, the main advantage of open source is the ability to detect unwanted code. At the time of writing, it is rumored that the NSA will force vendors to install backdoors in their security software. For a consumer who installs a firewall system, that is almost impossible to discover. But it is a big

drawback when these firewalls are used in your own network.

On the contrary, security experts can review open-source products and have a good chance of finding malicious code. Furthermore, it is challenging for vendors to install a backdoor in the source code if it is available for everybody to read and analyze.

Try before Buy. You can (and should) evaluate OPNsense thoroughly before spending money on infrastructure. That's the same idea as with a shareware application. Who is happy about limited functionality or a demo license that expires after 30 days?

In this context, *try* means evaluating with sample scenarios and *buy* stands for deployment in the local environment.

Hardware-independent. OPNsense is a software that requires some sort of hardware or virtual infrastructure. Since there are many options, all of which are acceptable, choosing the right one isn't easy. In terms of requirements and desired characteristics, for example, which piece of electronics is needed to saturate a 34 Mbps link with a VPN tunnel doing strong encryption? In the past, software-based network solutions could not keep up with the performance of a hardware device. The main reason for this was the terrible cooperation between software drivers and underlying hardware. The choice of network adapters, mainboards, CPUs, and memory is virtually unlimited, which makes it impossible for the software to get the maximum performance out of every combination.

Nowadays, regular servers or embedded systems have surprisingly good performance. Even with a non-optimized software and small packet sizes, it is possible to break the bandwidth level of 100 Mbps.

The Dutch corporation *Deciso* [1] and its *Netboard A10* tackled the question about which hardware component is best for the job. Optimization, adaptation and marketing have led to a respectable firewall appliance.

UNIX. Within OPNsense is a customized FreeBSD. Access to the operating system is possible but protected by a password. Login is permitted from the console menu or by an SSH connection. This flexibility allows you to customize, enhance, or install additional tools. Be careful though, these changes might also lead to unstable behavior.

Best Of. Although OPNsense doesn't reinvent the wheel, it does implement many services from the UNIX and Linux domain. The software has attained rock-solid stability after years of development. The web proxy comes from Squid, the SSH server is a subsidiary of OpenSSH and the firewall ruleset is the packet filtering engine *pf* from BSD.

Is use of OPNsense, an open-source software, theft of intellectual property? Not at all! It simply proves that open source works. When license terms are met, it is perfectly legal to integrate 3rd party software. It is highly recommended, especially in the security world, that application developers do not invent another crypto algorithm; but rather use stable free libraries.

History

The history of OPNsense is coupled with monowall and pfSense. At the beginning of 2003, *monowall* started as a firewall which used FreeBSD as its operating system. One year later, *pfSense* forked monowall with the goal of being better. This approach worked well and in 2006 pfSense out-

performed its predecessor regarding functionality and popularity.

The concept behind pfSense and its development was successful. In the following years, they released one version after the other.

The rivalry apparently ended in January 2014 when monowall published its ultimate stable version. The project announced its end later in February 2015 and ceased development on its firewall software.

Later in 2014, the US enterprise *Electric Sheep Fencing LLC* offered commercial support for pfSense and finally took over the firewall distribution. This acquisition resulted in a license change, which made it difficult for developers to get the source code.

This political change to pfSense and the downfall of monowall was the main reason that some Dutch and German developers started their firewall distribution as a fork of pfSense. Their aims were code quality, security, transparency, and tight integration with the community. The title *OPNsense* was intended as a reminder to its origin as pfSense.

The first version of OPNsense came out in January 2015 and was pfSense code dressed in a nice suit. Then the developers started working below the surface and replaced version after version pfSense code by their own code. At present (2019) both firewall distributions have approximately 10% common code lines.

OPNsense publishes new releases on a precise semiannual basis. This regular interval makes update schedules easy, and the community likes this strategy. Indeed, critical security patches have been released in the meantime, when necessary, so there is no need to wait for the next major release.

OPNsense and pfSense compete with similar goals to win trust of its users. Which distribution will win has not been decided yet: OPNsense with a fresh start and clear goals or pfSense with long-time confidence, stability, and a reputable name.

Part I

For Beginners

Chapter 1

Lab Network

An OPNsense firewall would be useless without a surrounding network to protect it. In practical terms, it is best to set up a separate lab network. Within this framework, it is safe to experiment with the firewall and its features without affecting any productive services.

All topics within this book have a practical background. Each chapter begins with the basics to establish an understanding or to refresh dusty knowledge. The examples and exercises are meant to be played with and rebuilt.

All chapters are based on the exact same network diagram, which represents a small corporate network with two remote sites and redundant wide area network (WAN) connections. Depending on the complexity of the topic, it might be enough to use only a small part of the lab network. Some chapters might have a different setup or an additional device. In that case, this will be explained in detail at the beginning of the section.

Resources

Using the exact same setup for the lab network removes the need to modify the infrastructure between chapters, i.e. there is no need to re-cable or modify the virtual

environment. This saves time and prevents error. And after a few chapters, the lab network will become more and more familiar, since the names of firewalls, clients, network adapters, and IP addresses remain unchanged. The complete network diagram is printed in Figure 1.1. The following chapters will use only part of this network for demonstration purposes.

There is no need to have physical access to the devices. It is even possible to operate the lab remotely.

The choice for hardware components depends on the expected throughput. More memory, CPU cores, and disk space will lead to a more powerful firewall that can handle higher bandwidth and more user sessions.

The official specifications for memory and disk capacity take into account bandwidth and activated features. The minimal requirements are low enough to run the full lab on a laptop computer or on cheap hardware. For example, an OPNsense firewall requires 512 MB memory and a 4 gigabyte hard disk.

Despite the ability to run a lab on minimal requirements, more CPU cores, memory space, and disk space are always encouraged. Table 1.1 gives the specifications for different configurations. The numbers are based on official documentation [2].

Specification	**Processor**	**CPU cores**	**Memory**	**Disk size**
Minimum	500 MHz	1	512 MB	4 GB
Reasonable	1 GHz	2	1 GB	40 GB
Recommended	1,5 GHz	2-4	4 GB	120 GB

Table 1.1: Hardware requirements for OPNsense

Some chapters use isolated networks, others need Internet access. The path to the Internet is always through the firewall in the core network. The firewall expects Internet access behind its network adapter *em0*. This is done with a NAT adapter in a virtual environment. In a physical setup, connect the network adapter to the DSL router. Any scenario that leads to the Internet is welcome.

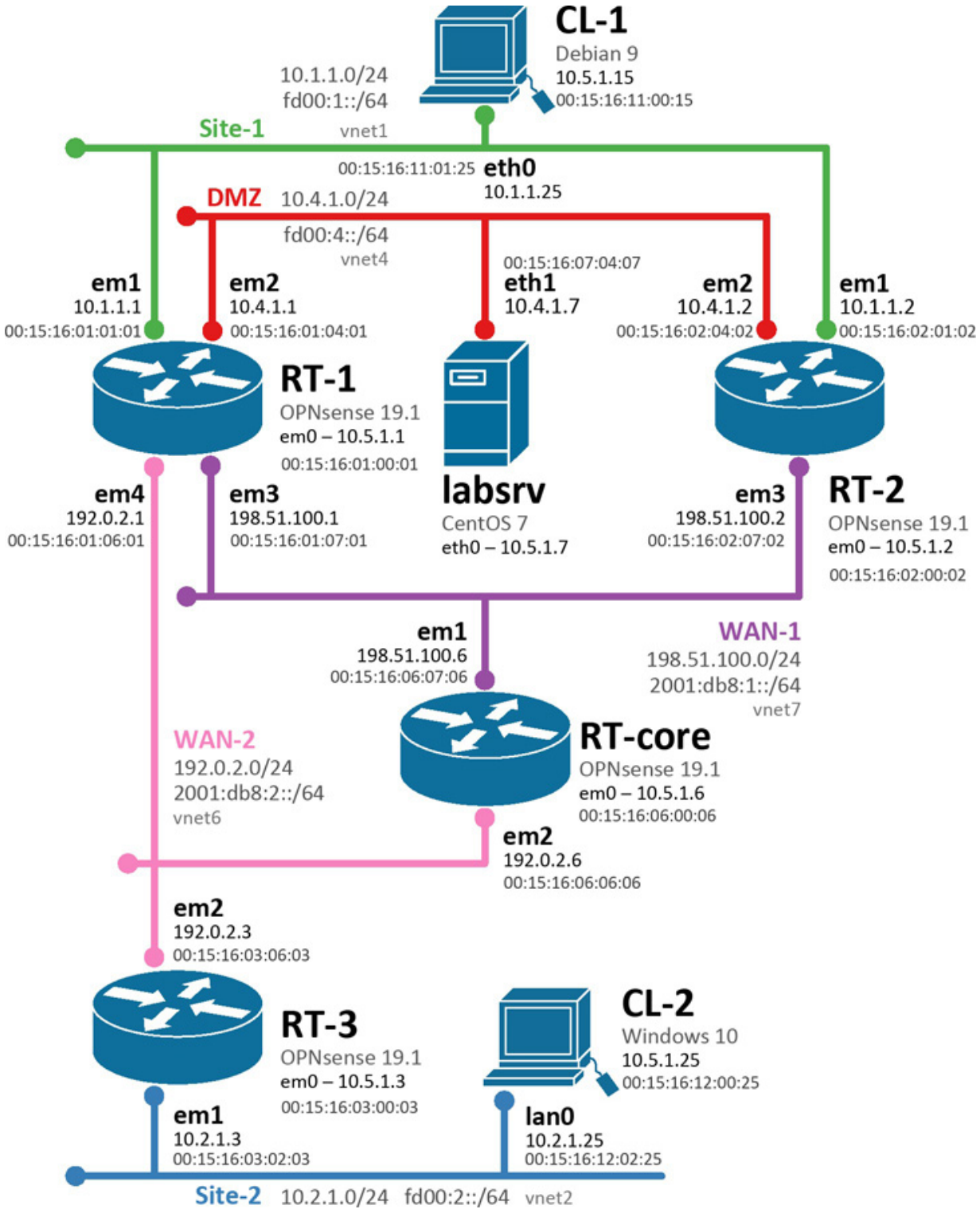

Figure 1.1: The lab network is a template for all chapters

Virtualization

It is possible to fully virtualize all devices used in the lab network. Each firewall becomes a separate virtual machine (VM) with virtual network cables connecting neighboring VMs. The interconnecting networks between the VMs are VMnetX (at VMware) and vboxnetX (at VirtualBox). A physical network adapter in the host system is only required when mixing the lab with real gear.

Firewall	**Interface**	**VMnet/vboxnet**	**IPv4**	**IPv6**
RT-1	em0 em1 em2 em3 em4	Management VMnet1 VMnet4 VMnet7 VMnet6	10.5.1.1 10.1.1.1 10.4.1.1 198.51.100.1 192.0.2.1	fd00:5::1 fd00:1::1 fd00:41 2001:db8:1::1 2001:db8:2::1
RT-2	em0 em1 em2 em3	Management VMnet1 VMnet4 VMnet7	10.5.1.2 10.1.1.2 10.4.1.2 198.51.100.2	fd00:5::2 fd00:1::2 fd00:4::2 2001:db8:1::2
RT-3	em0 em1 em2	Management VMnet2 VMnet6	10.5.1.3 10.2.1.3 192.0.2.3	fd00:5::3 fd00:2::3 2001:db8:2::3
RT-core	em0 em1 em2	Management VMnet7 VMnet6	10.5.1.6 198.51.100.6 192.0.2.6	fd00:56 2001:db8:1::6 2001:db8:2::6
labsrv	eth0 eth1	Management VMnet4	10.5.1.7 10.4.1.7	fd00:5::7 fd00:4::7

Table 1.2: All firewalls with network adapters and VMnet/vboxnet

Table 1.2 lists which interface belongs to which virtual network. Though not technically required, the network adapters of the VMs can use a predefined MAC address to help interpret the output. This technique makes it easy to recognize a device when reading a command output or when comparing it with the samples in the book.

All labs are tested and validated with VMware Workstation 14, VMware ESXi 6 and VirtualBox 5.0.

Hardware

OPNsense performs well on all devices with i386- or x86_64-type architecture. Even the brand of the network adapter is not that important since the lab setup is intended to demonstrate features and not to provide the best possible performance. Check the FreeBSD hardware compatibility list [3] to see compatible gear.

Networks

The network architecture between the firewalls is based on Ethernet. Every subnet is its own broadcast domain. It is therefore important not to mix cables of different segments. Two methods are available to correctly separate the subnets: by switches or by VLANs.

Separate by switches

Each network segment uses a separate network switch or hub. The switches are not interconnected.

A smaller 5-port device is sufficient since the subnets are rather small. Any brand and model should be sufficient for the job.

Separate by VLANs

All cables are connected to a single switch. Cables and switch ports that belong to the same network segment are members in a common virtual LAN (VLAN). For example, all switch ports leading to the light gray core network will be a member of VLAN 6.

The switch device must have enough ports to provide connectivity for all the firewalls. The switch is not required

to provide routing between VLANs. A regular VLAN capable layer 2 switch is adequate.

It is even possible to mix both modes. For instance, both WAN segments could connect to one switch and the site subnets could connect to another switch. The requirements correspond to the method *Separate by VLANs*.

Firewall

The OPNsense firewall uses the current stable version 19.1 as a 64-bit image. If different versions or additional devices are included, then the lab is modified by replacing the device in question.

Each firewall has one additional network adapter for management access. That way, a SSH client will still reach the firewall even if some configuration change has failed and the regular interfaces become inaccessible. If the hardware does not provide an extra interface for device management, it is acceptable to skip this option.

The lab firewalls are serially numbered. The device number is echoed in the IPv4, IPv6, and MAC addresses. This allows easy device identification in a command output listing.

The name of each network adapter is printed next to the router symbol. The full IPv4 address is added below. Information about IPv4 subnet and IPv6 prefix is presented at the network line icon.

Addressing

The subnetworks of the imaginary remote sites use private IPv4 addresses and unique local IPv6 addresses. Each site

has a client computer, which is used only to validate a feature or generate traffic. The required command set is limited to ping, traceroute, netstat and a web browser. Even the choice of the operating system is irrelevant – the demo lab picks Debian Linux and Windows due to their popularity.

The area between the sites is the core network. Devices in this network use the address ranges 192.0.2.0/24 and 198.51.100.0/24, which are reserved for documentation (RFC 5737).

The addressing of IPv6 also uses two unequal prefixes to visually simplify the differences: prefix fd00::/16 is used in site networks and prefix 2001:db8::/32 is used in the core network.

The address ranges are intended exactly for this purpose and do not collide with public prefixes. Furthermore, the addressing is kept straightforward. All ranges are structured uniformly and have only “regular” netmasks of /24 (IPv4) or a prefix of /64 (IPv6).

Table 1.3 summarizes the IPv4 and IPv6 ranges attached to the `VMnet` networks. Additional addresses (e.g. for PPPoE, tunnel, CARP) are derived from the same ranges.

vnet	**Purpose**	**IPv4**	**IPv6**
VMnet1	Site 1	10.1.1.0/24	fd00:1::/64
VMnet2	Site 2	10.2.1.0/24	fd00:2::/64
VMnet4	DMZ	10.4.1.0/24	fd00:4::/64
VMnet5	Management	10.5.1.0/24	fd00:5::/64
VMnet6	VPN	10.6.0.0/16	fd00:6::/64
VMnet7	WAN light gray	192.0.2.0/24	2001:db8:2::/64
	WAN dark gray	198.51.100.0/24	2001:db8:1::/64
	VPN	203.0.113.0/24	2001:db8:3::/64

Table 1.3: All virtual networks with IP ranges

Lab Server

The lab server provides infrastructure services. It can run on physical hardware or as a virtual machine. If the OPNsense firewall is evaluated for a client/ server protocol, the lab server will be the counterpart. It can accept requests from the firewalls on NTP, DNS, Syslog, FTP/TFTP, NetFlow and HTTP. The deployed lab server runs on CentOS 7.

Utilization

Each chapter uses a subset of the full lab network. Lesser devices provide better control, simpler examples and briefer command output. This limitation leads to a better overview. Feel free to insert additional firewalls to dive deeper into features.

Chapter 2

Platform

The next step is all about setting up the lab components. It begins with the creation or purchase of the equipment, followed by installation and finally networking.

As mentioned in Chapter 1, the lab can run on physical hardware or find its home entirely in a virtual environment. This makes a big difference in the structure, but is irrelevant for the example scenarios in the following chapters.

The installation procedure is the same for all methods: it first begins by creating the virtual networks, which are separated either by a virtual switch or a port group. The next step is to set up the virtual machines (VMs) and finally the new VMs put their network adapters into the local VM networks.

The choice of virtualization software depends on your personal preferences. The following explanations apply to VMware ESXi, Workstation and Player, and VirtualBox.

This chapter cannot substitute as a reference manual for VMware or VirtualBox! The installation of the VMs requires a basic knowledge of the respective products. The descriptions only cover the installation of the new VM and not why the individual steps are necessary or advantageous.