free CD inside 7 Must-have applications 4 Discount coupons inside!

starterkit

hakin9

practical protection

hakin9 starterkit 2/2007 (2)

Practical IT Security Solutions for Newbies

Issue 200702 (2) Vol.1 No. 2 14.99USD 14.99AUD Bimonthly ISSN 1896-9801

Knock Knock Knockin'

On Firewall's Door

Exclusively
for **hakin9** starterkit

Discount coupons inside!

Dekart's Private Disk Multificator
50% off

WaterMark Factory
50% off

Enigma Lite Desktop Edition
50% off

NetConceal Anonymizer
20% off

Must-have applications on the CD

- **Comodo Firewall Pro** (2.4 version)
- **m0n0wall Firewall**
- **NetConceal AntiHistory** (full commercial version worth $24.90)
- **NetConceal Anonymizer** (trial version)
- **MicroWorld e-Scan ISS** (120-day trial version)
- **Watermark Factory** (45-day trial version)
- **Enigma Lite Desktop Edition** (90-day trial version)

PLUS

- Introduction to Firewalls
- Essentials of Firewalling, iptables and Firewalls' Capabilities
- Highly-Redundant Network Firewall: pf + CARP
- Web Application Firewall – mod_security for Apache
- Linux Netfilter – Packet Mangling and Applications
- Easy Firewalling with IPCop

02
0 74470 22432 7

dekart
MAKE IT SECURE

Looking for a reliable encryption solution?

Private Disk Multifactor

Easy and secure backups

NIST certified AES 256-bit encryption

Data wiping

Disk firewall

PRIVATEDISK MULTIFACTOR

Windows 64-bit compatible

50% discount

Get your own copy at half the price by visiting *dekart.com/offers/pdmf/*

Coupon code: “hakin9”

Multiple factors of authentication

Something you **know**	+	Something you **have**	+	Something you **are**
password or PIN		USB key,smart card...		voice,fingerprint,eye...

... or developing one of your own?

Private Disk SDK

Extend your applications using the SDK's encryption mechanism for securing your workflow, implementing strong NIST-certified encryption into your applications without requiring specialized security expertise. Dekart Private Disk SDK includes everything you need to quickly and easily build security into new or existing publishing, healthcare, legal, financial, and other applications. With its unique *Disk firewall* feature the developers can ensure that their applications have an exclusive access to the protected disk, thus providing the highest possible level of security. The entire range of features available in Private Disk can be added to your custom application with ease.

The package includes sample projects of different complexity, the samples *'just work!'* so that you can fix your own fully-functional application in just a little bit more than 5 minutes!

The SDK is a set of dynamic linked libraries and system drivers that can be used in multiple programming languages.

Check out Password Carrier

Dekart Password Carrier automatically collects and securely stores the passwords and private details you type when you log on to web-sites or use Windows applications. Next time you need to provide your user credentials, Password Carrier does that for you. Phishing protection, keylogger protection and strong password generation are just a few of the facilities offered by our product.

Password Carrier works with a broad range of portable devices. Besides USB flash disks, it can be used with flash memory cards (such as SD, MMC, CF or MS), digital audio players or digital cameras. To summarize, Password Carrier works with any memory device which is detected as a removable drive when connected to a Windows PC.

Upgrade offer

Upgrade to the latest version with a 30% discount by visiting *dekart.com/offers/pc/*

Coupon code: "hakin9"

Firewalls ABC

Firewalls are believed to be a first line of defense, and hence it should be a topic that each serious computer user knows, like the back of their hand. Firewalls are standard equipment for each Internet connection nowadays. They are a necessary part of every organization's IT security system. Home users who connect to commercial Internet service providers are also using firewall appliances to protect their connections.

A firewall is a device configured to deny, permit or proxy data connections which is set and configured by the organization's security policy. The above definition sounds rather simple, but it covers millions of terms, types, technique and procedures. Thanks to *hakin9* starterkit – Firewalls Edition you will learn some crucial things on firewalls – hardware or software based firewalls, network and personal ones, stateful and stateless firewalls, network layer firewalls, iptables, ISO/OSI stack levels, and much more...

Together with our authors we created a guide to firewalls to present the most important aspects of this great branch of IT security. We prepared a couple of general articles that will help you to enter the complex world of firewalling. The other articles are devoted to certain sections like:

- Web Application Firewall – mod_security for Apache,
- A Highly-Redundant Network Firewall: pf + CARP,
- An Introduction to Firewall Rulebases.

Apart from the technical writing, *hakin9* starterkit contains a lambent column by Matt Yonkman and a CD with interesting surprises prepared exclusively for our dear readers. We hope that you will take advantage of the discounts we negotiated for you with Dekart, WaterMark Factory, GlobalTrust and NetConceal.

We do realize that a firewalls topic is so broad that we could have a separate monthly magazine devoted to it. In this issue, however, we wanted to provide a set of essential information that would project the importance of firewalls and encourage every reader to penetrate it on their own. Let this edition of *hakin9* starterkit be your primer and initiate your adventure with IT security management.

It is just a second edition of *hakin9* starterkit. We are curious indeed about, what you think of our magazine, what comments or advise you would give us, etc. Open to your suggestions, we invite you to contributing to the next issues.

Magdalena Błaszczyk
magdalena.blaszczyk@hakin9.org

CD Contents 06

What's new in the latest *hakin9.live* version (3.2.1-aur., updated versions of Aircrack and Kismet) and what must-have applications you will find (*NetConceal Anti-History & Anonymizer, Enigma Lite Desktop Edition, MicroWorld e-Scan Internet Security Suite, Watermark Factory, Comodo Firewall Pro, m0n0wall*).

hakin9 starterkit **Practical IT Security Solutions for Newbies**

Editor in Chief: Ewa Dudzic *ewa.dudzic@software.com.pl*
Editor: Magdalena Błaszczyk *magdalena.blaszczyk@hakin9.org*
Contributing Editor: Shyaam Sundhar R. S.
DTP Director: Marcin Pieśniewski *marcin.piesniewski@software.com.pl*
Art Director: Agnieszka Marchocka *agnes@software.com.pl*
CD: Rafał Kwaśny
Proofreaders: N. Potter, D. F. Leer, M. Szuba, Kelley Dawson
Top betatesters: Wendel Guglielmetti Henrique, Justin Seitz, Peter Hüwe, Damian Szewczyk, Peter Harmsen, Kevin Bewley, Steve Lape

President: Monika Godlewska monikag@*software.com.pl*
Senior Consultant/Publisher: Paweł Marciniak *pawel@software.com.pl*
National Sales Manager: Monika Godlewska *monikag@software.com.pl*
Production Director: Marta Kurpiewska *marta@software.com.pl*
Marketing Director: Ewa Dudzic *ewa.dudzic@software.com.pl*
Advertising Sales: Magdalena Błaszczyk *magdalena.blaszczyk@hakin9.org*
Subscription: *subscription@software.com.pl*
Prepress technician: Marcin Pieśniewski *marcin.piesniewski@software.com.pl*

Publisher: Software Media LLC
(on Software Publishing House licence *www.software.com.pl/en*)
Postal adderss:
Software Media LLC
1461 A First Avenue, # 360
New York, NY 10021-2209
USA
Tel: 004822 8871010
www.hakin9.org/en

Software LLC is looking for partners from all over the World.
If you are interested in cooperating with us,
please contact us by e-mail: *cooperation@software.com.pl*

Print: 101 Studio, Firma Tęgi
Printed in Poland

Distributed in the USA by: Source Interlink Fulfillment Division, 27500 Riverview Centre Boulevard, Suite 400, Bonita Springs, FL 34134 Tel: 239-949-4450.

Distributed in Australia by: Gordon and Gotch, Australia Pty Ltd.
Level 2, 9 Roadborough Road, Locked Bag 527, NSW 2086, Sydney, Australia
Tel: + 61 2 9972 8800

CD Contents

hakin9 starterkit magazine comes with a CD full of exciting surprises:

It contains *hakin9.live* (h9l) version 3.2.1-aur, which, apart from selected firewall related tutorials, contains special editions of most interesting commercial applications prepared exclusively for our readers.

With this *hakin9* starterkit *CD* you can choose between booting *hakin9.live* and m0n0wall!

hakin9.live is a well-known bootable Linux distribution crammed with useful utilities and tutorials. To start using *hakin9.live* simply boot your computer from the CD. After booting, you can log into system using the *hakin9* term for user, the password is no needed. *h9l* version 3.2.1-aur is based on the Aurox 12.0 distribution.

The system runs the 2.6.17 kernel with some patches and features improved hardware detection and network configuration.

The default graphical environment is currently based on KDE 3.5.5. It looks very nice and is highly configurable and has very modest hardware requirements. As usually, you can find the Aurox Installer on *h9l* 3.2.1-aur. After launching it on the disk, you can install additional programs using the yum command.

Within this *hakin9.live* CD you get almost 30 updated package versions of the *hakin9.live* programs including Aircrack (keys cracking program) and Kismet (wireless network detector, sniffer, and intrusion detection system). Materials on *h9l CD* are selected in appropriate directories:

- doc – indexes in HTML format,
- tut – tutorials,
- apps – special versions of commercial applications.

Tutorials assume that we are using *hakin9.live,* which helps avoid such problems as different compiler versions, wrong configuration file paths or specific program options for a given system.

The current *hakin9.live* version consists of especially selected tutorials covering firewall-related topics.

Especially for *hakin9* starterkit readers we enclose some interesting applications that will enhance your IT security maintenance:

NetConceal AntiHistory – a full commercial version of an application that erases history and tracks of your activity. Clears web browser's history, cookies, temporary files, index.dat files, recent document lists, recently typed information, search history and many other dangerous tracks of your activity. Includes manual and scheduled operation, uses advanced data erasing techniques and supports vast majority of systems and applications out of the box. VERY easy to use: just install and click 'Start'. Retail price $24.90

NetConceal Anonymizer – a trial version of anonymity shield that hides your real IP address and other private information (your location, your ISP etc) by redirecting all of your network activity through special Internet computers, known as Proxy Servers. It uses special SOCKS protocol to communicate with proxy servers instead of communicating directly to target Internet resources.

Enigma Lite Desktop Edition – 90-day trial version of a tool that offers a modular and scalable solution to secure document storage and transfer; it meets not only the requirements of single users, by protecting their desktop or mobile computer, but also those of great organization local and geographic networks.

Enigma allows you to implement the strongest security policies and to ensure the protection and the privacy of files, e-mails, databases and of the whole document flow.

MicroWorld e-Scan Internet Security Suite – 120-day trial version of a comprehensive security solution designed and developed to protect personal computers from threats like: Viruses, Spyware, Adware, Malware, Keyloggers, Hackers, Spammers, Privacy related issues, objectionable content and many more.

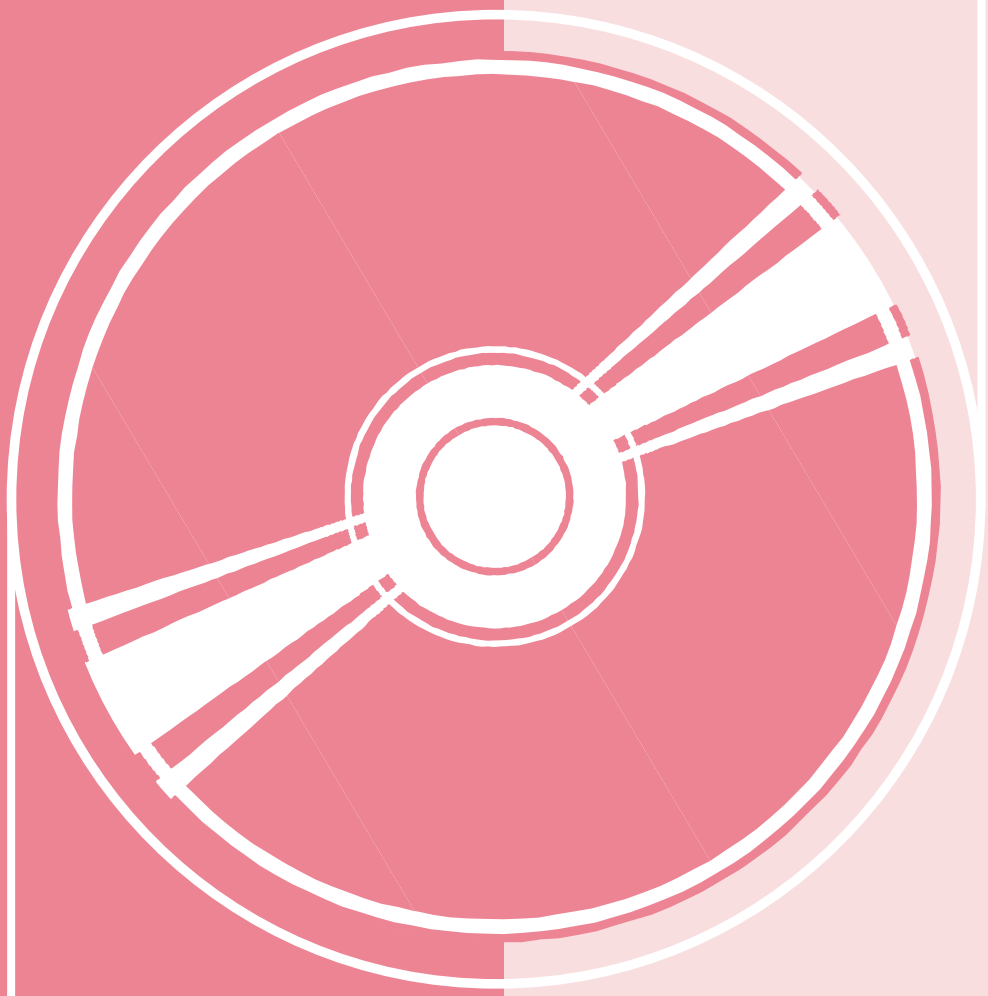
After 30 days of using the trial you will be ask to register in order to get 90 days more for free.

Watermark Factory – 45-day trial version of a application allowing to add text or image watermark to any picture. Protect your copyrights or simply add comments to any picture. This useful program has beautiful and easy to use interface. You will be able to process thousands of files in a few seconds.

***Comodo Firewall Pro* (2.4 version)** – helps you to understand what is going on by analyzing each alert and providing you an intuitive, easily understandable Security Considerations section with each question it asks. Constantly monitors and defends your PC from Internet attacks, has free maintenance upgrades, lets you gain complete control over which programs are allowed Internet access as well as stay protected against new threats via automatic online updates.

m0n0wall Firewall – is an embedded firewall software package that, when used together with an embedded PC, provides all the important features of commercial firewall boxes (including ease of use) at a fraction of the price (free software). ●

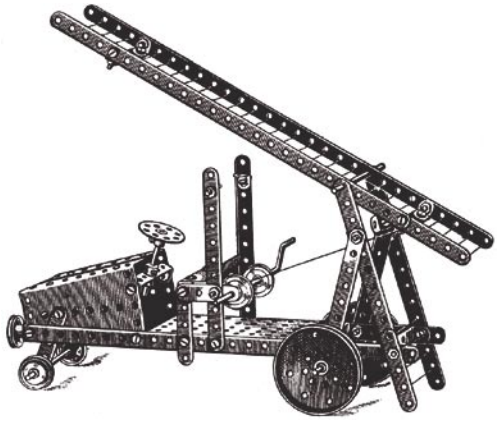
If you have encounter any problems with this CD, write to: *cd@software.com.pl*



If the CD contents can't be accessed and the disc isn't physically damaged, try to run it in at least two CD drives.

Introduction to Firewalls: From ISO/OSI to DMZ

Michele Orrù



I've been using, configuring and administrating firewalls for at least 3 years, in which I've seen and tested the most used of those like Iptables, BSD's PF, Ipfilter, Checkpoint, common firewalls in embedded devices, and almost all software firewalls.

After all of my studies, I arrived at the conclusion that a firewall is mainly a concept, not a product (hardware or software). It is more like security that is a process, not a product.

Additionally, this is a guide for newbies, so that in this article I will try to explain all the information that is needed to understand firewalls.

So... we can say in a less abstract manner than before, that usually a firewall is something (a device) that manages and controls the connection with something else.

Those things could be single workstations, servers, or more complex LAN or WAN.

However, every firewall has common features and functionalities, similar to those which different firewall vendors give to us, but usually we can divide firewalls into different typologies.

So how can you choose which firewall is more useful and convenient for the business and the security of your enterprise, or for the information that you have in your home or office?

Usually, if you are the security manager who works for a large company, and in your everyday schedule you have to evaluate this in something called Risk Determination, which is the last phase of a long and expensive procedure called Enterprise Security and Risk Management, which helps you to define the most important Assets, and the correlated Threats and Vulnerabilities.

After all these evaluations, you will be able to choose the most appropriate solution that satisfies the security level that you need for the assets of the company.

What you should know...

- TCP/IP basics,
- classic IP packet structure,
- computer networks basics.

What you will learn...

- firewall generic architecture,
- how the rule parsing works, and how those rules are managed,
- how to choose the best firewall solution for your situation.

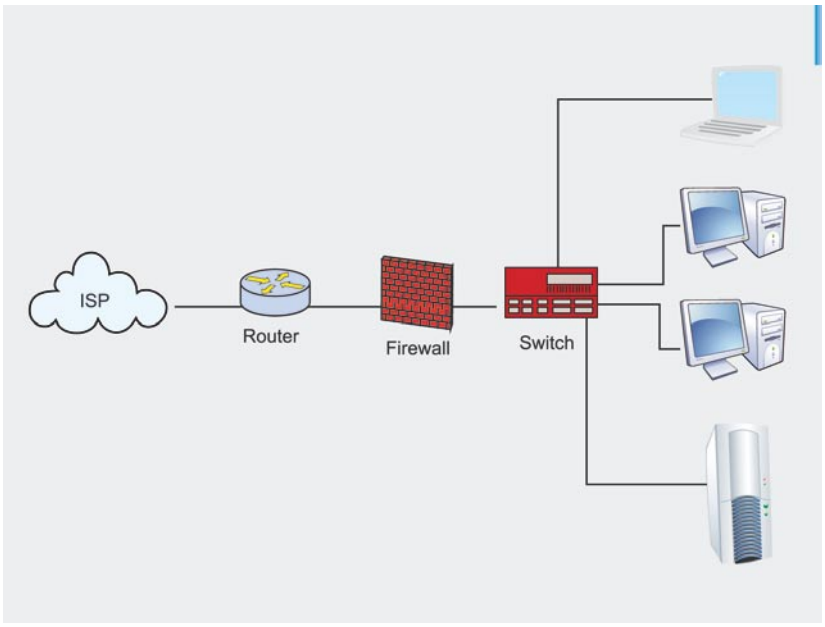


Figure 1. *A simple situation with a firewall that manages a small LAN*

Software vs Hardware Firewalls

A software firewall is, as the name implies, a program that you install and configure in a single computer, and that usually is only one of many processes running on your machine.

Examples of these solutions are Zone Alarm, Norton Internet Security Suite, McAfee, BitDefender, Panda, etc. There are exceptions, such as Checkpoint.

This proprietary good solution is a huge software that you install in a dedicated machine that in theory will manage multiple connections and a lot of traffic.

All of these vendors usually combine their products with antivirus, antispyware, phishing control

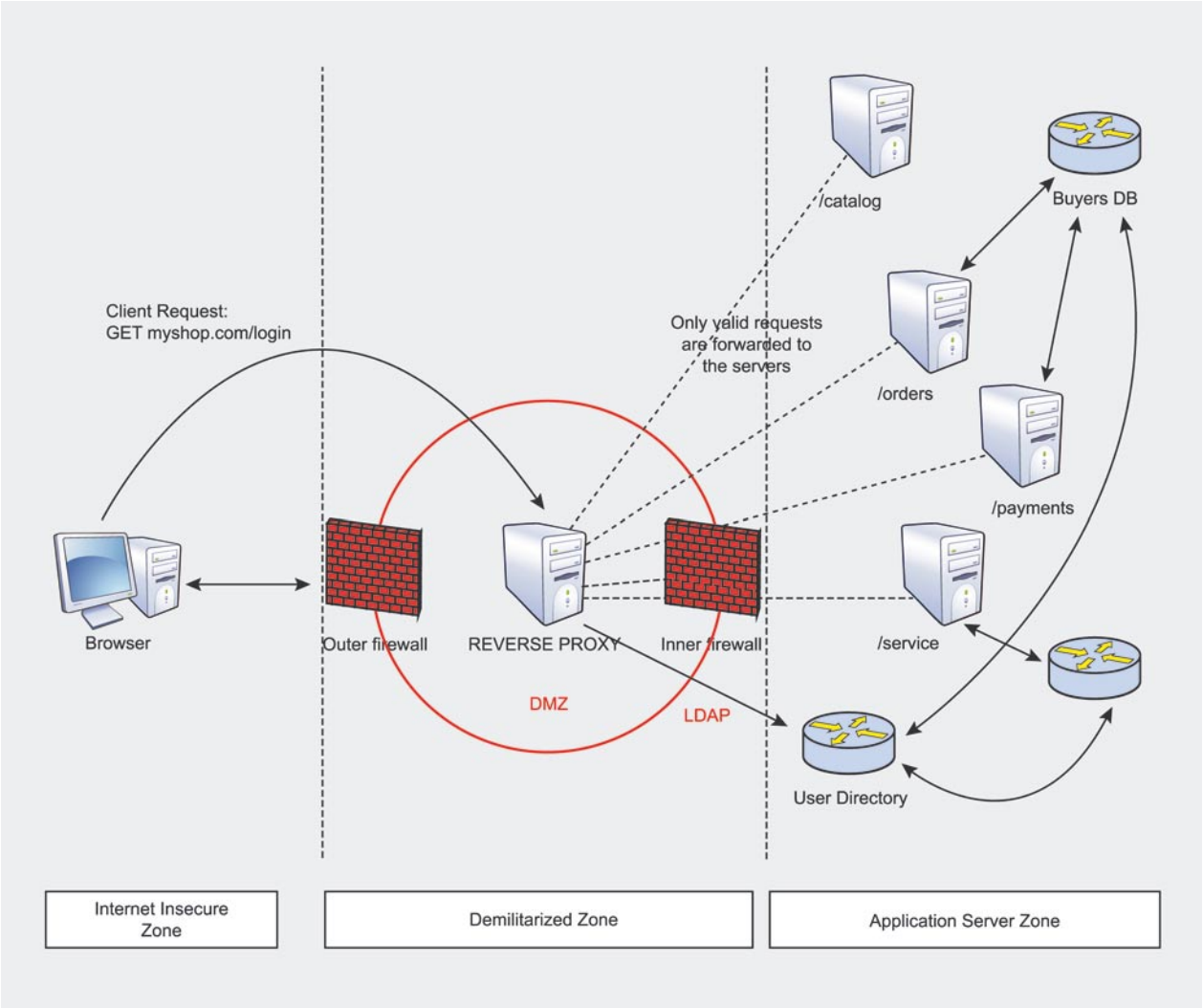


Figure 2. *A complex enterprise situation with business-critical applications. A multi-layer/in-depth protection is the best approach to ensure high availability and security*

and other quite useful bundled programs (from the economique technique called Bundling) in a big-suite called xxx-Internet Security. Software firewalls are not as expensive as hardware solutions, and are quite secure if well configured, and of course if you trust your vendor. Of course they are not the best solution if you need speed, high-availability, and if your firewall needs to manage a lot of connections at the same time, and if you need high values of throughput (in the order of GigaBits).

Also, a hardware firewall is a soft-ware, of course, but explicitely dedicated to do only a few tasks, and on an extremely hardened operating system. The most known examples are the Cisco PIX firewalls, that are dedicated hardware solutions that run the Cisco IOS operating system with some other proprietary programs like a firewall, a VPN manager, and other useful things. These devices are so performant that the throughput is in the order of GigaBits, and are dedicated to do only one thing, or a few tasks. So the selected hardware is optimized to do only those things: for example Cisco and 3Com devices use ARM processors, more performant than other architectures when you need to route a lot of packets. Of course you can create your own hardware firewall solution, but you need to evaluate the time, the knowledge and the cost. The vendors have of course already evaluated this, and they give to you an optimized and performant solution. Surely you can, as I usually do, create your own firewall solutions: it's exciting, cheaper and good to increase your knowledge. You have to use trusted and secure operating systems, like *Open/FreeBSD* with PF firewall – maybe as a kernel module instead of a usermode program – or Linux-based distros like Debian and Gentoo, enforced with Grsecurity kernel patches and Iptables as firewall.

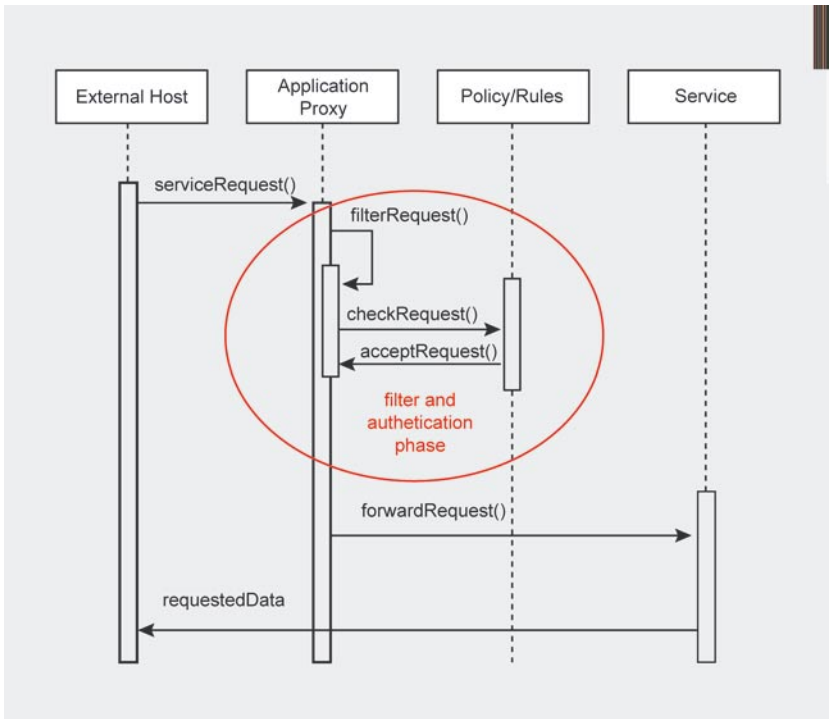


Figure 3. *Sequence diagram that shows how an external host request is filtered, checked and, if allowed, finally accepted*

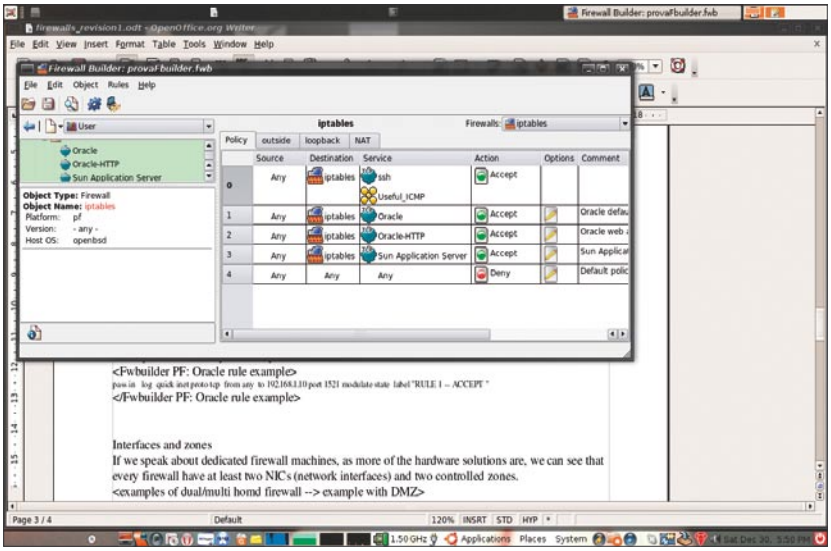


Figure 4. *Firewall Builder in action. The simple and effective GUI is the best way to manage complex and huge rulesets minimizing the errors*

ISO-OSI stack level

Another way to classify firewall solutions is based on which level of the ISO-OSI stack they work.

We have at least three different types of firewalls using this categorization:

Stateless Packet filter firewalls: they define a basic filtering function at the IP layer based on packet inspection, typically of network addresses. They work on the level 3-4 of the ISO-OSI stack. With this type of firewall you can, for example, filter – accepting or rejecting- individual IP packets based on IP: port tuples. The big problem is that they cannot manage the state of these packets. Usually with a simple client/server communication the sent/received packets are a lot: a stateless firewall cannot know if

'e Scan™ AntiVirus & Content Security

eScan

The brighter side of computing

- MWL Technology
- AntiVirus
- Adware / Spyware Removal
- E-mail Scanning
- Anti Phishing
- Web Scanning / Parental Control
- Anti Spam with NILP
- Privacy Protection / Web Washer
- Central Management Control
- Hardware & Software Asset Management
- Pop-Up Blocker
- Firewall
- Automatic Hourly Updates
- 24 x 7 Free Technical Support

(NILP - Non Intrusive Learning Patterns)

Microsoft Partners Antivirus

Microsoft Partners E-Mail Content Security

MicroWorld Technologies Inc.
33045 Hamilton Court East, Suite 105
Farmington Hills, MI 48334-3385, USA
Tel: (248) 848 9081/9084
Fax: (248) 848 9085

MicroWorld Technologies GmbH
Leopoldstr. 244
D-80807 München
Tel: +49 (89) 20 80 39 - 223
Fax: +49 (89) 20 80 39 - 226

MICROWORLD

We add confidence to computing

sales@mwti.net
www.mwti.net

a packet from incoming traffic to the server is part of one previous communication with the same client, or if it's a new request from a different client.

Stateful Packet filter firewalls: they have the same funtionalities of a packet filter firewall, but they also manage the state of the packets, which describes when incoming packets are part of a new connection, or part of a continuing communication whose connection was approved previously.

All the stateful firewalls work at the levels 3-4 of the ISO-OSI stack.

Stateful analysis occurs for TCP, UDP and ICMP packets. For connectionless protocols such as UDP, it works mainly with values and packet tuples (address:port, like 10.0.0.1:8080). With protocols like TCP, which manage their state, a sequence number is used with a sliding window representing the connection. The sequence number is important in preventing dangerous attacks like session hijacking, which are possible if the attacker uses some programs (the old Juggernaut, Ettercap, an so on) that are able to guess the next number of the session sequence and inject the attacker into the communication. This technique is not as simple as I've described, but if the hacker is able to deeply understand the situation in which he's doing hacking (the network, the systems, the security level), he could be able to reduce the firewall effectiveness.

Some advanced firewalls, like BSD's PF (originally distributed with OpenBSD), have the capability to modulate the state of those packets, affecting TCP sequence numbers translating those to strongly random increments, which prevent session hijacking and passive OS detection with tools like pOf.

Application Proxy firewalls: with this type of firewall every request for a service is authenticated and checked. We can verify that the requests are authentic, and filter out some payload attacks like, for example, a wrong command for a service, wrong type parameters in service calls, directory traversal in HTTP requests, privileged resources or programs requests done without the needed rights.

They usually provide a higher level of security than packet filters, because they inspect the complete packet including the headers and data segments. The main problem is that it is difficult to find application proxies for all applications: for the most used (like HTTP, FTP, SMTP) we can manage that, and also with OpenSource and Free applications. But if we are using proprietary solutions, this is more difficult and expensive.

Examples of this type of firewall include Squid (HTTP proxy), mod_security (Apache module), Socks protocol, and some Postfix filters.

General Firewall Structure

Usually a firewall has at least three components (the examples below are about the BSD's PF firewall):

- the engine (PF main process): the main program that manages the main firewall functions, like filtering packets and applying defined policies,
- the rules file (pf.conf): the file where we write all the rules and our policies that the firewall will use,
- the rule *parser/shower* (pfctl): the program that parses and afterwards can show all the *rules/policies* present in the rules file. Usually in the most advanced firewalls this program rearranges the rules, so the firewall doesn't

Listing 1. *PF and Iptables rules*

```
PF:
pass in   log  quick inet proto tcp  from any  to 192.168.1.10
 port 1521 modulate state  label "RULE 1 -- ACCEPT "

IPtables:
$IPTABLES -N RULE_1
$IPTABLES -A OUTPUT -p tcp -m tcp  -d 192.168.1.10
 --dport 1521  -m state --state NEW  -j RULE_1
$IPTABLES -A INPUT -p tcp -m tcp  -d 192.168.1.10
-dport 1521  -m state --state NEW  -j RULE_1
$IPTABLES -A RULE_1  -j LOG  --log-level info
 --log-prefix "RULE 1 -- ACCEPT "
--log-tcp-options  --log-ip-options
$IPTABLES -A RULE_1  -j ACCEPT
```

Default Policies

Experienced firewall administrators know well how important the default policies are if their networks are under attack. The less-expert admins might think: why? Well, try to imagine a situation in which your firewall has to process a lot of traffic and packets, and the rules are in the order of one hundred or more, maybe because the services that you want to make disponibile are a lot and they use a lot of ports (application servers, for example). Now try to imagine that you are seeing a lot of incoming malicious packets from an unknown source address, on a privileged service (*port number* < *1024*) that you don't want to put in listening mode (for example *identd* or *telnet*), and that of course is not mentioned in your rules file. Your firewall will start to parse all the defined rules, without finding a good rule to apply on those packets: the final result is that it will use the default policy. Now there is the more interesting part: if this policy is set up to ALLOW INCOMING, all the malicious packets will not be rejected, but rather forwarded to the requested service. Otherwise, if the default policy is set up to DENY INCOMING, the packets will be rejected because the firewall will block them, without knowing what it has to do.

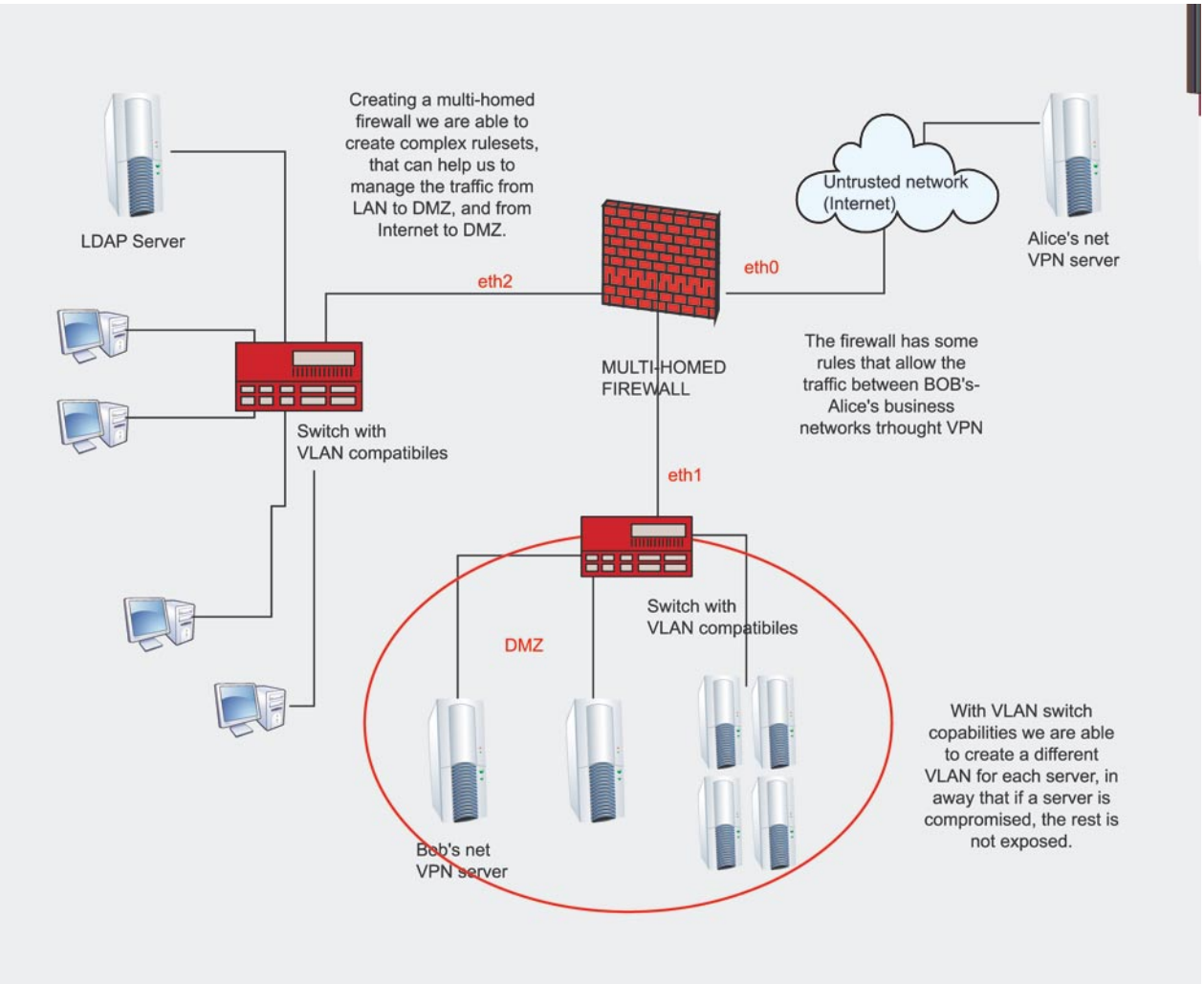


Figure 5. *A multi-homed firewall that manage a LAN and a DMZ, and their connection with the a trusted/untrusted network (in this case the Alice NET as trusted – thought a VPN – and Internet as untrusted)*

About the Author

Michele Orrù, a.k.a euronymous, is an Italian IT student charmed by the security world. He is an *aficionado* of Linux and BSD systems, and in general all that is not Microsoft. He believes in OpenSource software, like software development methodology and business approach. He admins heterogeneous networks with security in mind. Lately his primary interests are J2EE with special considerations about security, like LDAP, PKI, trusted OS-es, XML security and integration with business environments.

References

- [1] Cheswick, Bellovin, Rubin, *Firewall and Internet Security, Second Edition*, Addison-Wesley, 2003,
- [2] Schumacher, Fernandez, Hybertson, Bushmann, Sommerlad, *Security Patterns: Integrating Security and System Engineering*, Wiley, 2006,
- [3] Palmer, Nazario, *Secure Architectures with OpenBSD*, Addison-Wesley, 2004,
- [4] Firewall Builder website, *http://www.fwbuilder.org/*,
- [5] OpenBSD's PF, *http://www.openbsd.org/faq/pf/*.

use a top-down approach on the rules order: the rules parser is able to speed up the filter process, optimizing the position of the controlled rules that the firewall will use.

Rules, rulesets and policies

Writing rules and policies is the main phase of a firewall configuration, and is surely the most important part. All firewall administrators should remember:

- to create rules with the least privilege, like DEFAULT DENY policies for both incoming and outgoing connections,
- to KISS: Keep It Simple, Stupid,
- to write as few rules as possible:
- fewer rules = fewer resources

consumed = easier administration = cheaper total cost.

Writing rules for heterogeneous systems, with different types of firewalls like Cisco PIX, BSD's pf and Linux iptables could be difficult, especially if the network architecture is complex.

To help us in writing rulesets, we can use OpenSource tools like Firewall Builder. This powerful tool has a very useful GUI and can manage a lot of rulesets: the approach is to make your policy/rules configuration, and afterwards choose for which type of firewall/OS to make the final ruleset.

The most used firewalls are supported: PIX, PF, iptables, ipfw, ipfilter, Cisco FWSM.

Some rules examples are the following: you can see the differences between two good OpenSource firewalls, OpenBSD PF and Linux Iptables. The rules of both allow traffic to an Oracle DB server.

(I will not spend time explaining in-depth the syntax of the rules below, because this guide is not a howto for writing rules and policies with Iptables or PF: I recommend the reader view a PF/iptables FAQ that can be easily found on Internet.)

One of the most advanced features of Fwbuilder is the rule compiler: this consists of several elementary building blocks, or *rule processors*. Each rule processor performs an elementary operation on a rule and passes it to the next. The operations that the compiler performs are verification, transformation and optimization.

Interfaces and zones

If we speak about dedicated firewall machines, as most of the hardware solutions are, we can see that every firewall has at least two NICs (network interfaces) and two controlled zones. The most-used solutions are the dual-homed (two NICs) and multi-homed (three or more NICs).

Typically we use a dual-homed firewall in a situation like in Figure 5 (ISP – internal LAN).

DMZ Importance. Why it's so useful in a production system?

Reading the common definitions of a DMZ, maybe you cannot understand how useful this security solution can be. The best way to understand this is with a practical example. Try to imagine a classic network situation with a router, a firewall, a LAN and the servers, one web server that manages the presentation layer of a huge e-commerce aplication, and an application server (that contains also the database) that manages the business part.

We imagine that all the hosts in the LAN are Windows clients, and that the servers are physically in a server room, but logically in the same subnet (for example, if the router IP is 10.0.0.1, one host 10.0.0.8, one of the two servers will have an IP like 10.0.0.78, because the subnet is 10.0.0.0/24 and the netmask 255.255.255.0).

This means that every host can have network access to the servers, can execute pings or more dangerous operations, and can have all the packets forwarded by default to the servers.

Now try to imagine that the not-so-skilled administrator left open the router and on the firewall some ports used by VNC (5800-5900) to have remote GUI control on some of the LAN hosts, for example,one of those dedicated to act as Active Directory server. Now suppose that there is a hacker that for a few days has been scanning more-or-less casual IP ranges, to find some vulnerable VNC daemon that are listening (At *www.milw0rm.org*, you can find the exploit and the program to scan automatically the vulnerable hosts).

Now imagine that this hacker found the router IP and discovered that the VNC server is vulnerable to the exploit that he has. So, the hacker will execute the exploit, obtain remote control on the compromised machine, install a rootkit (yes, also for windows exists kernel rootkits, thanks to Greg Hoglund) with some reverse backdoor (that will be able to bypass almost all of the firewalls, because it's reversed), start the enumerations and scanning activities on all the LAN, find the servers, and discover that, for example, the Linux application server has a kernel version with a remote vulnerability. The game is over, and I can stop here with the example.

Now try to imagine the situation in which there is a DMZ: trough the network implementation choices the skilled administrator has decided to use a dedicated firewall for the DMZ with restrictive rules, like DENY ALL both for the incoming/outgoing traffic.

The servers have been put into the DMZ, especially the application server that is the most business-critical asset because it is the business part of the applocation (it manages private information about customers, like credit card numbers).

The DMZ has been configured explicitly with another subnet, 10.0.1.0/24, and there are not possible routes betwenn the LAN and the DMZ, and also for the opposite. The DMZ administration tasks could be done only by being physically present in the server room (maybe, for remote administration, an SSH solutions through port knocking could be configured). Now let me come back to the previous hacker situation; he has already compromised one host in the LAN, and he's going to compromise all the rest, but when he's enumerating and scanning the network he cannot find anything interesting (except for some hosts where he can install keyloggers, trojans and other malicious programs). This is because the firewall cannot be seen from the LAN, because no routes from the LAN to the DMZ exist. The customers' privacy is preserved by the DMZ, avoiding the possible credit card numbers theft: also the less skilled administrator, after some days of suspicious network traffic (if the hacker is not-so-skilled, of course), will start to think that somethig bad is happening, and maybe he will start to put some IDS or honeypot probes in the net, to find the hacker and denounce him to the authorities.

Usually in a dual-homed solution, especially if there is an internal network with a lot of workstations and we don't have a lot of publicly available IPs, we have to implement NAT capabilities on the firewall: with Network Address Translation we are able to map all the internal machines behind the firewall with the firewall IP, and create one or more different subnets to manage each segment of our LANs. There will be the firewall to manage all the connection and to show its address to the WAN or to other networks (potentially untrusted).

If we have to build or buy a more complex firewall solution (maybe for an enterprise that has a LAN and some servers) the best and safest solution is to separate the internal networks (that you have to treat always like the external ones, as INSECURE) from the server zone, with something called DeMilitarized Zone (DMZ): in this zone we will put all the servers that need isolation from the rest of the net – maybe also in a different subnet – with specific rules in the firewall (and a dedicated interface) that will control all the traffic (and the few internal IPs that can have access to the DMZ).

DMZs solutions are effective only if the firewall that manages them is well configured and hardened: if the firewall is compromised, also the entire DMZ could be compromised. In these situations the best approach is to implement the so called *Security in Depth* (also known as *Defense in Depth*) principle: multi-level security, with different levels of protection for different types of resources (and business values).

Finally, we can assume that the benefits of DMZ solutions are many: the global security is increased; it's multi-level and in-depth; it's transparent to users if well configured. The possible liabilities are an increased total cost, as both for structure and manageability, and possibly less performance and availability. However, a DMZ solution is always preferred in production systems but, as with all security products, liabilities and benefits must be well evaluated and compared.

Conclusion

Like all the concepts/products related in the information security world, so are firewalls not a *panacea*: they cannot help you solve all problems, and complex networks administrated by not- so-skilled admins could become your worst nightmare, with complex rulesets, abstruse configurations and too much permissive/restrictive default policies.

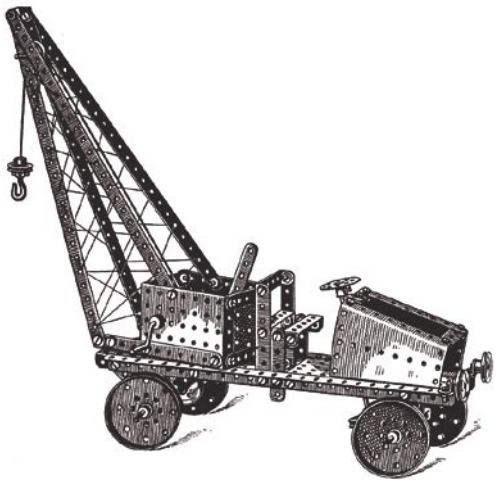
However, if wisely configured and developed, they could become a very useful solution to enforce network/host security, especially if they are combined with IDS/IPS systems with reactive rulesets that interact with the firewall.

Finally, I never mentioned Open and Free vs Closed and Commercial firewall solutions: I'm a convinced Linux/BSD user, and I believe in the power of Open Source software, as a software development methodology and as business approach.

In certain cases – such as big-enterprise systems – you have to consider buying Closed Source solutions, like *Cisco/3Com/Checkpoint* products, or at least configure your personal firewall solutions in this heterogeneous ambient, and so interact with different products. Making this choice between Open/Proprietary solutions depends on your budget, your security needs and the enterprise assets that you have to protect: no doubt a Cisco PIX 5xx firewall is powerful, secure and "Cisco," but costs thousand of dollars. So if you have to protect a LAN of 20-30 hosts with a NAT-ted firewall, maybe also with a little DMZ, a Pentium-4 with 1 GB of RAM, with an optimized FreeBSD installation with PF directly compiled in the kernel is enough (and surely less and less expensive). Of course, you will not have technical support from anybody, but you have the knowledge and your mind, and I think that this is enough! ●

Introduction to Firewall Rulebases

Gr@ve_Rose **(Sean Murray-Ford)**



Firewalls – We have all heard of them in some way or the other and most of us have worked on them as well. Although prevalent to our network security infrastructure, not a lot of us in the IT world have taken the time, or had the opportunity to learn more about the firewalls that we use, why we use them and the best practices for using them.

This article helps the reader to understand more about firewalls, both from the good guy's (security specialist's) perspective as well as a bad guy's (attacker's) perspective. This article will focus on some general principles of setting up and securing a small office or home network – Although not the most exciting task, this will help lay the groundwork for the larger projects which will be covered further down the road.

The first thing to become familiar with is the OSI model, especially the second, third and the fourth layers where most firewalls operate and this is where most of this article will be focused. We also have to understand that, at layer three, there are (theoretical maximum of) 256 protocols and at layer four (when dealing with TCP and UDP) there are 65,536 ports. Both these numbers starts from zero, and not one. Some people forget about layer three and its strengths. For example, what if the incoming as well as the outgoing UDP traffic is blocked in your network. Instead of clogging up your rule base with over sixty thousand ports (which causes your firewall to potentially slow down), you can just block IP/17 and be done with it. In such cases, the packets not only match one specific criteria but will match one layer prior to that, if you specify all ports.

Now that the dry theory is out of our way with some helpful reminders, let us take a look at some of the general practices for your environment. Firewalls are usually defined with *legs* where a *three-legged* firewall is the one that protects three interfaces and, which is generally three independent networks. If you are running a home firewall or a small office firewall, you may just have an *in-and-out* firewall with a public interface and a private interface. Let us start with that example...

What you will learn...

- how to create firewall rulebases,
- what a rulebase looks like in most firewall software,
- how to understand and extrapole client requirements,
- the ever-important *Stealth* and *Clean-up* rules.

What you should know...

- basic IT security knowledge.

Although not very exciting and, what some would consider basic, this is indicative of small offices and home networks throughout the world. Perhaps there are a few other devices such as a networked printer or SAN, behind the firewall as well, but you get the idea. You might think that this would be easy as pie to secure, but here you may be wrong. Before reading up on the article, imagine that your task is to create the rule base on this firewall. Most of you will know *block incoming and allow outgoing traffic* which, for the most part is true for firewalls but what kind of traffic are you letting out? What operating system(s) are being run behind the firewall? Are there any specific user applications that must be allowed through the firewall to be accessed by the partners? What about the users who has to use VPN software to connect to their headquarters? What kind of firewall are you using? These questions should be addressed and answered before you start plunking firewalls into the customer's networks. Let us now consider that a customer has asked us to build them a firewall with the following criteria:

- Users cannot surf the web,
- Users should have access to a web-based application at `1.2.2.1` on port 80 (TCP),
- Users should be able to use SNMP to monitor a dynamic host on the 'Net,
- You, as the network security person/analyst should have remote SSH access to manage the firewall,
- Users should be able to route GRE to random external destinations.

In real life, this would be a horrible situation to work with,doing SNMP queries and setting up GRE tunnels while logging data into a web-based application. Nonetheless, you have been contracted to setup the rules. Let us examine the steps. First, take a look at the criteria and let us break it down:

- Users are not allowed to surf the web, which means no TCP/80 or TCP/443,
- Users should use TCP/80 for a specific application,
- Users should have UDP/161 outbound to a dynamic host outside,
- You should be able to hit the firewall on TCP/22 from outside,
- You should allow IP/47 outbound.

Something is left out on purpose... Try spotting it before moving on.

Most firewalls create rules in a top-down and left-to-right fashion so let us examine what a rule base may look like...

Breaking this down, we can see that the rule (number zero) is the first to be accessed when a packet arrives at the firewall (regardless of interface unless you are running Cisco Pix or any other interface-based firewall). The packet will then be examined against this rule to see if the criteria matches, what the packet is trying to do. In this case, the firewall will check to see if the packet is sourced from the internal network and if it isn't, it will move on

Tabela 1. *The looks of rule base*

Number	Source	Destination	Service	Action	Notes
0	Internal	DMZ	TCP/22	Allow	Insecure

Tabela 2. *A rule base example*

Number	Source	Destination	Service	Action	Notes
0	Internal	1.2.2.1	TCP/80	Allow	Custom Application
1	Internal	Anywhere	UDP/161	Allow	SNMP Monitor
2	Internal	Anywhere	IP/47	Allow	GRE
3	Internal	Anywhere	Anything	Drop	Internal Drop
4	My_Office	Firewall	TCP/22	Allow	Remote Management

Tabela 3. *General guidelines*

Number	Source	Destination	Service	Action	Notes
0	My_Office	Firewall	TCP/22	Allow	Remote Management
1	Anywhere	Firewall	Anything	Drop	Stealth Rule
2	Internal	DNS_ Servers	UDP/53	Allow	Name Resolution
3	Internal	Anywhere	UDP/161	Allow	SNMP Monitor
4	Internal	1.2.2.1	TCP/80	Allow	Custom Application
5	Internal	Anywhere	IP/47	Allow	GRE
6	Anywhere	Anywhere	Anything	Drop	Clean-up

to the next rule, if any. If it is from the internal network, the firewall will examine the destination and see if it is destined for the DMZ. If so, it passes on to the service where, if the service is equal to TCP/22 (SSH), the action is taken which, in this case, is to allow the packet to pass through the firewall.

Knowing this, we can now formulate a plan for developing our rule base. The best way to create a rule base is to start with the specifications and set those up first. Then, once you have them out of the way, create your blanket rules to catch the rest. Let us take a look at what our rule base could look like to accommodate our client.

We have got our custom web-based application setup as our first rule, our SNMP next and followed by with the GRE. Rule number three blocks anything else from the internal network and we have allowed remote management through SSH from the office. We have met all our criteria and can assume that we have done a good job, right?

Wrong. You get a call in the middle of the night stating that the users can't access the SNMP device at `snmp.example.net` and that their firewall is acting slow as well. You get logged into the machine, run *w* and notice that the unit has been rooted and that *sendmail* is running, queuing Spam like it was going out of style. You get into the office, check your logs and notice that there are a lot of UDP/53 drops from the internal network and a multitude of ICMP sweeps, SSH brute force attacks and other items of nastiness which make you cringe just at the thought. What is wrong?

The first thing is with the client's requirements of accessing a dynamic host. If the IP of the host is going to change, have to access DNS to query the current IP of `snmp.example.net` when we access it. This is the tricky part that is intentionally left out – We have to allow UDP/53 (DNS) out from the Internal network.

Secondly, we do not have a *stealth rule*. A *stealth rule* is a rule that hides the firewall from everyone except those, who are supposed to see it. In best practices, you should define your firewall access rules first, add a *stealth rule*, and then add your user rules. This way here, your firewall will appear (almost) invisible to those who are not supposed to have access to it.

The last item in our checklist is a *clean-up* rule. A *clean-up* rule (also known as an Explicit Drop) is to drop everything. This is our *catch-all* so that if a packet makes it all the way through the rule base and doesn't match anything that we have defined, it should be dropped or otherwise discarded. If we do not know about it, we do not want it.

With these general guidelines clearly defined, let us revisit our rule base and see how it should look when configured properly...

Let us examine our new rule base in a bit more detail:

- Rule 0 – Allow access from *My_Office* to the Firewall on TCP/22,
- Rule 1 – Drop anything to the firewall,
- Rule 2 – Allow the Internal network to access the DNS servers for DNS queries,
- Rule 3 – Allow SNMP outbound. There is no destination restriction on this rule, as we do not know what it is or what it will be at any given point of time,
- Rule 4 – All the internal network HTTP access to our partner site where the custom application is being hosted,
- Rule 5 – Allow outbound GRE setup. Once again, there is no destination restriction, as we don't know where the GRE tunnels will be setup at any given point,
- Rule 6 – This is our *clean-up* rule. Anything going in any direction to, from or through the firewall in any shape, way or form is to be dropped.

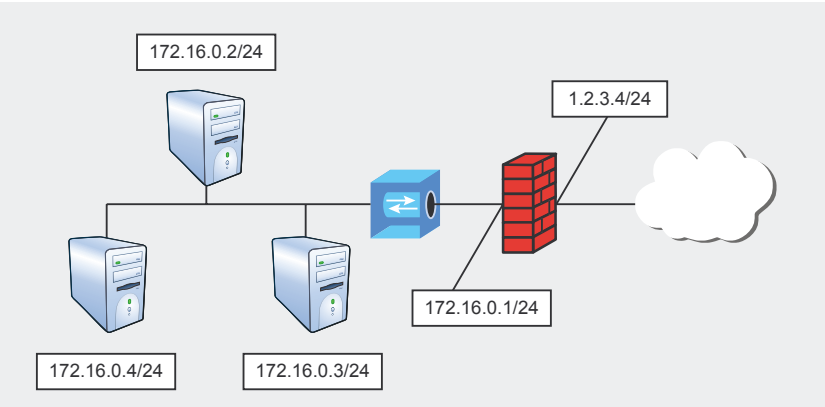


Figure 1. *In-and-Out*

As you can see, this rule base with only two additional rules and a bit of a shuffle, is quite a lot more secure than the previously created rule base. The *clean-up* rule comes in very handy for a multitude of reasons such as, a virus that comes into a computer on the internal network. Without the *clean-up* rule, this virus would be allowed to propagate into the wild, consuming your bandwidth and infecting other computers. One thing, which a lot of people overlook, is that your Internal network(s) aren't clean. They are not as vicious as the external interfaces, which are almost always under some form of attack, but you should never implicitly trust anything, which is why we have our *clean-up* rule – If we do not know about it, we don't want it. ●

About the Author

Gr@ve_Rose (Sean Murray-Ford) has been working in Network Security for over eight years focusing primarily on firewalls, Linux and IPv6. He has created two Linux distributions and published multiple whitepapers and independant documents on security related issues.

your Privacy guard

NetConceal Anonymity Shield

Why lose...

Hide IP Address:
Keep Your Privacy

Be careful - IP address is a unique ID which points exactly to your location, with very high precision, right to your home door.

Also:
- Fake Country of Origin
- Unban From Forums and Newsgroups
- Access Restricted Sites
- Play Games Anonymously
- Anonymous Web, IM, P2P, E-mail.

Less than $20, lifetime, no annual fees!

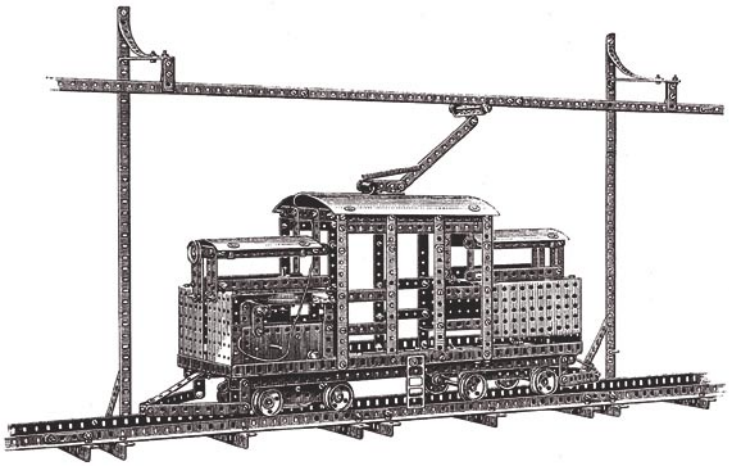
Discount
Coupon code:
XTY635

Order Now For 20% Off*
*offer expires June.30th

Get It Today! ***www.netconceal.com***

Knock Knock Knocking On Firewall's Door

Raul Siles



Firewall technologies today are still a critical component to protect systems and networks. Standard filtering solutions can be complemented with more advanced techniques to secure the services behind a firewall.

This entry level article constitutes a brief introduction to a concept known as port knocking, and specifically to one of its derivatives, Single Packet Authorization (SPA). From a practical perspective, the article provides a step-by-step guide for beginners to deploy fwknop (an open-source SPA implementation) using FC6 Linux, and provides additional links for those interested in getting in-depth knowledge about these technologies.

Introduction: Demystifying Firewall Technologies

In trying to demystify firewalls as the one-and-only solution that solves all security threads, it seems useful to exemplify firewall technologies through the following analogy: *Firewalls are like traffic lights, they simply allow or deny traffic, and once they allow it, the traffic flows through*. A green traffic light indicates traffic is allowed, while a red traffic light indicates traffic is denied. Both scenarios can be directly mapped to firewall rules with an action of allow or deny. Additionally, an amber traffic light indicates traffic is still allowed but with caution; in the firewall world, this can be translated to a rule allowing but logging traffic.

This analogy helps to dismiss firewalls as a unique line of defense for an IT infrastructure, highlighting; the need to apply defense in depth principles. The technologies covered in this article provide additional protection to increase the security provided by a firewall, applying defense in depth to the firewall device itself.

Fortunately, the usage of a firewall as the one and only protection mechanism was a common thought years ago, but today, any security conscious IT professional knows that multiple technologies are required to prevent, detect and respond to all the different attacks going on

What you will learn...

- How to deploy fwknop (an open-source SPA implementation) using FC6 Linux,
- The in-depth knowledge about technologies presented in the article,
- The basics of port knocking concept.

What you should know...

- The standard firewall capabilities.

through IP-based networks. However, the previous analogy should not devaluate the relevance of firewalls, in the same way traffic lights are crucial to regulate and manage traffic in the real world.

Simplifying, firewalls filter traffic based on source IP address and port and destination IP address and port. There are multiple types of firewall technologies, such as packet filters, stateful filters, proxy gateways, application-level firewalls… but, roughly speaking, they simply increase the granularity and conditions that can be specified in a firewall rule to allow or deny traffic. Therefore, each firewall rule is like a tiny traffic light, managing the flow of certain traffic specified by the rule definition.

Firewalls work by following a statically defined ruleset (except for some temporary rules dynamically generated by stateful filters) that can only be modified manually by changing the firewall policy through the firewall configuration file or graphical interface. Based on that, one could wonder if this type of solution, a traffic light that is opened (or closed) 24x7, is required in all the scenarios where a firewall is needed.

Port knocking, and its related technologies, provides new add-ons to the standard firewall capabilities, avoiding the always on traffic light role, and providing access controls similar to the main door barriers in most buildings, where you need to identify yourself prior to getting temporary access to the facilities.

Port Knocking

Port knocking is a network-based client-server communication mechanism that allows creating new rules in a firewall and, therefore, opening specific ports remotely. Its main advantage is that it makes possible to establish network connections through the firewall using ports that (most of the time) are closed.

The standard port knocking mechanism is based on generating (using the port knocking client) a set of connection attempts on a set of predefined closed ports. These connection attempts will be analyzed by the port knocking server, and once the correct sequence is identified, the firewall ruleset is dynamically modified to add a new rule that allows the source host (client) to connect to a target (previously closed) port.

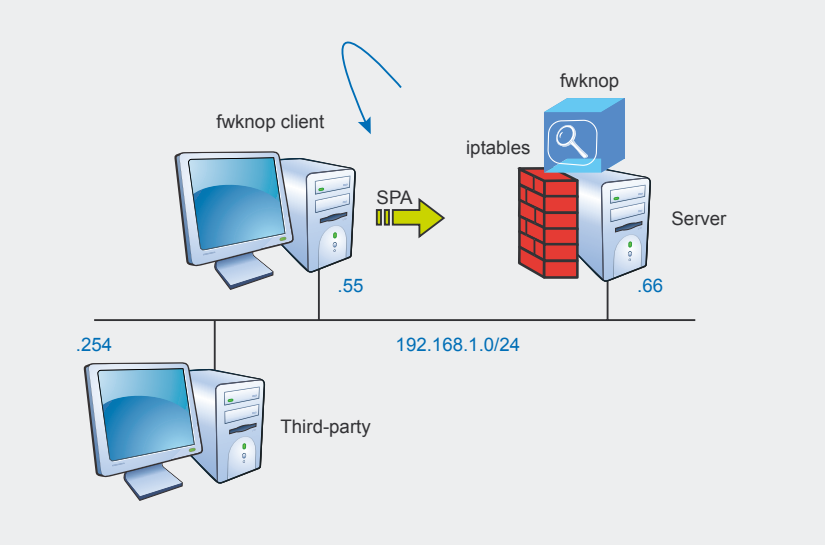


Figure 1. *Basic port knocking architecture*

Some related literature refers to port knocking as an advanced network authentication system, such as the first paper on the topic [1] published in 2003. Some port knocking advanced evolutions, like Single Packet Authorization (SPA), are referred as an authorization mechanism [2]. Although port knocking and SPA both require authentication and authorization, from a general perspective, they mainly provide advanced remote firewall management capabilities and make the services protected by the firewall stealthier. Their main goal is to remotely manage the firewall policy adding (or removing) new filtering rules. Obviously, these sensitive management tasks must be properly authenticated and provide strict authoritative controls to only allow the modification of certain rules.

Port knocking is sometimes negatively identified as a security through obscurity solution, but, it tends to be an additional layer of defense that applies the principle of less privilege to a highly granular level, that is, at the firewall rules level. If a specific firewall rule is not required all the time, what is the advantage of having it always active? The same principle is applied in other security solutions, such as authentication systems that only allow certain users to log in during specific timeframes, such as working hours. Port knocking makes resources available only when they are required, that is, when the user requests them using the port knock sequence.

Finally, port knocking is commonly implemented in two modes. First one is called file mode and is based on configuring a daemon to watch the firewall log files for connection attempts, identify the knock sequence or SPA, and then modify the firewall configuration accordingly. It can also be implemented using network mode, where the process examines packets gathered through live traffic captures using libpcap.

Port Knocking History

The port knocking term was coined by Martin Krzywinski [1] [4]. Its capabilities were extended by Michael Rash in the original fwknop implementation in 2004 [5], combining traditional encrypted port knocking with passive OS fingerprinting. Then, the concept evolved to Single Packet Authorization (SPA), which retains the benefits of port knocking, but has other advantages such as the usage of asymmetric ciphers for encryption, provides non-replayable packets, and a remarkable simplicity and speed

because it only sends a single packet over the network. SPA was coined by MadHat in 2005 [8] and, at the same time, Michael Rash provided the most commonly used, actively maintained and supported implementation nowadays, fwknop [5].

More general details about port knocking and SPA, their benefits, disadvantages and difference can be obtained from the main port knocking portal [3] and initial related articles [4], from the fwknop homepage [5] and from several presentations on the topic between 2004 and 2006 in major security conferences, like Toorcon, Shmoocon, BackHat, DefCon…

Step-by-step guide to deploy SPA using fwknop in FC6

The following section describes step-by-step how to deploy Single Packet Authorization (SPA) using fwknop [5] on Linux, using Fedora Core 6 (FC6) for both the client and the server systems. The tool is going to be deployed to manage iptables (Netfilter) acting as a personal firewall to protect a FC6 system, that is, the firewall only filters traffic entering or leaving this single computer. Fwknop can also be deployed on a multi-homed firewall, allowing or denying traffic for other systems or networks.

Step 1. Downloading fwknop

The first step to deploy fwknop is to download the tool from the official homepage, available at *http://www.cipherdyne.org/fwknop/download/*. The current version available is 1.0 from November 05, 2006. To simplify the installation in FC6, it is recommended to download the RPM package, called fwknop-1.0-1.i386.rpm. Once downloaded to a user directory, for example */home/eric*, its MD5 checksum should be verified to check the software integrity. The value must match the one published in the official Web page:

```
$ md5sum fwknop-1.0-1.i386.rpm
1c4bd61a49dffae3bcacc8ab06b86802
  fwknop-1.0-1.i386.rpm
```

Step 2. Installing fwknop

The next step is to install the fwknop RPM package as root in both, the SPA client and the SPA server:

```
$ su -
Password:
# rpm -iv /home/eric/ -->
fwknop-1.0-1.i386.rpm
Preparing packages for installation.
fwknop-1.0-1
[+] You can edit the EMAIL_ADDRESSES
 variable in
/etc/fwknop /fwknop.conf /etc/fwknop/
fwknop.conf to have email
 alerts sent to an address
   other than root\@localhost
```

Step 3. Pointing to the right libpcap version

The fwknop server (192.168.1.66) requires the packet capture library (libpcap) to capture the network traffic containing the SPA packets. The current version was compiled using the 0.8.3 version of libpcap, but FC6 includes by default the 0.9.4 version. Fwknop can work with the latest version, although it is required to create a symbolic link (just on the server) to point the old version to the new one:

```
# ln -s /usr/lib/libpcap.so.0.9.4 /
usr/lib/libpcap.so.0.8.3
```

Step 4. Reviewing fwknop files and directories

In order to understand how a tool works, it is always good practice to inspect the RPM package contents to list the files the software includes. For those readers interested in all the details, the following command provides the full list of files:

```
$ rpm -q --filesbypkg fwknop-1.0-1 | more
```

To sum up, fwknop saves all its configuration files under the `/etc/fwknop/` directory. The client binary is called `/usr/bin/fwknop` and the server binaries are available at `/usr/sbin/` (search by `*knop*`). All the libraries required by the tool are stored under `/usr/lib/fwknop/`, the documentation (manual pages) are available at `/usr/share/man/man8/fwknop*` and the tool log files are located at `/var/log/fwknop` and `/var/run/fwknop`. The server startup script is called `/etc/rc.d/init.d/fwknop` and enables the service for run-levels 3, 4 and 5:

```
# chkconfig --list | grep fwknop
fwknop    0:off 1:off 2:off 3:on 4:on
5:on 6:off
```

Step 5. Reviewing file and directory permissions

The default root umask that determines the default file permissions in FC6 is 0022, therefore, after installing fwknop some files have permissions too relaxed. It is recommended to restrict them through the chmod command so that they can only be read by root, especially files that contain clear text passwords, such as access.conf:

```
# umask
0022
# ll /etc/fwknop/
total 72
-rw-r--r-- 1 root root  6147
Nov  6 03:54 access.conf
-rw-r--r-- 1 root root   883
Nov  6 03:54 alert.conf
-rw-r--r-- 1 root root  7637
Nov  6 03:54 fwknop.conf
-rw-r--r-- 1 root root  1523
Jan  5 00:11 knopwatchd.conf
-rw-r--r-- 1 root root 26526
Nov  6 03:54 pf.os
# chmod 600 /etc/fwknop/*
```

OS (FC6) Requirements

The examples on this article are based on the Fedora Core 6 (FC6) Linux distribution. A minimal installation was performed on the tests systems (client and server), selecting only the default *Base* software packages from the *Base System* group, recommended for creating small router/firewall boxes.

Additionally, the following packages are required:

```
$ rpm -aq | grep -i libpcap
libpcap-0.9.4-9.fc6
$ rpm -aq | grep -i iptables
iptables-1.3.5-1.2.1
iptables-ipv6-1.3.5-1.2.1
$ rpm -aq | grep perl
perl-5.8.8-10
```

Not only the configuration files permissions must be modified, but also the tool client executable; the fwknop server binaries permissions are restrictive enough (500) by default:

```
# ll /usr/bin/fwknop
-rwxr-xr-x 1 root root 44817 Nov
  6 03:54 /usr/bin/fwknop
# chmod 700 /usr/bin/fwknop
```

Step 6. Configuring the fwknop server: fwknopd

Although port knocking and SPA can be used to protect any service and its associated ports, they are mainly used to protect and enable access to remote management protocols, and especially long running TCP sessions, such as Secure Shell, SSH (TCP port 22).

The article presents two configuration examples. The first one focuses on allowing access to SSH while the second allows the client to manage remotely the rules that should be applied in the iptables firewall.

Fwknop offers grater capabilities than these two introductory examples, so for the seasoned reader, it is recommended to check fwknop documentation and test more advanced scenarios. More information can be obtained from the tool manual pages (fwknop, fwknopd, knopmd and knopwatch). Fwknop configuration files are very well documented and the tool homepage [5] provides extra details in the form of documents, FAQ and related articles.

Step 6.1. Customizing the access.conf file (example1)

The access.conf file defines the access control directives and defines parameters for accepting single-packet authorization messages. The example below allows SPA packets from any source IP address via libpcap.

Once a valid packet is received, fwknop will dynamically reconfigure the local iptables policy to allow temporary access to the SSH daemon from the IP address specified in the SPA packet for 30 seconds:

```
SOURCE: ANY;
OPEN_PORTS: tcp/22;
DATA_COLLECT_MODE: PCAP;
KEY: $uper$ecre7;
FW_ACCESS_TIMEOUT: 30;
```

The default configuration has been modified by adding the two directives marked in *bold*, the service port and the secret key. Additionally, it is possible to open multiple ports at once, as in `OPEN _ PORTS: tcp/22,upd/53;`.

Fwknop offers many advanced options, such as using public/private GPG keys as an alternate authentication encryption method instead of the original symmetric key.

Additionally, instead of allowing access, fwknop can be configured to allow a remote client to send a command to the fwknopd server, which will be executed as root. It is even possible to restrict what commands can be executed through regular expressions or require what must be the username on the system that generates the authorization packet.

Step 6.2. Understanding the pf.os file

In addition, fwknop also implements the traditional port knocking concept and provides parameters for accepting shared or encrypted knock sequences.

It is even capable of enforcing what must be the operating system from which the encrypted sequence originates.

This functionality is implemented through passive OS fingerprinting techniques using a database adapted from the p0f tool [6] (available in the pf.os file).

Step 6.3. Understanding the alert.conf file

The alert.conf file is used to configure how the main two daemons, fwknop and knopwatchd, generate notification alerts.

By default they send alerts using syslog and email (*root@localhost*).

Step 6.4. Customizing the fwknop.conf file

The fwknop.conf file is the main Firewall Knock Operator configuration file and provides lots of different options and controls to manage things, like the integrity of the timestamp between the client and the server (by default their clocks could differ a maximum of 120 seconds). To simplify the article, only the following two parameters have been modified.

On the Net

- [1] Port Knocking – Network Authentication Across Closed Ports. Martin Krzywinski. SysAdmin magazine. June 2003. Volume 12, Number 6, *http://www.samag.com/articles/2003/0306/*,
- [2] Single Packet Authorization with fwknop. Michael Rash.USENIX. February 2006. Volume 31, Number 1, *http://www.usenix.org/publications/login/2006-02/pdfs/rash.pdf* (login required),
- *http://www.cipherdyne.org/fwknop/docs/SPA.html*,
- [3] Port Knocking, *http://www.portknocking.org*,
- *http://portknocking.sourceforge.net*,
- [4] Port Knocking. Martin Krzywinski. Linux Journal. June 2003, *http://www.linuxjournal.com/article/6811*,
- [5] Fwknop: Firewall Knock Operator. Michael Rash, *http://www.cipherdyne.org/fwknop/*,
- [6] p0f: Passive OS Fingerprinting. Michal Zalewski, *http://lcamtuf.coredump.cx/p0f.shtml*,
- [7] nmap: Network Mapper. Fyodor Vaskovich, *http://insecure.org/nmap/*,
- [8] SPA: Single Packet Authorization. MadHat and Simple Nomad. BlackHat Briefings USA 2005, *http://www.blackhat.com/presentations/bh-usa-05/bh-us-05-madhat.pdf*.

By default fwknopd will sniff traffic in promiscuous mode on the default Linux network interface (eth0).

Due to the fact this is not required while protecting a single system, the setting has been modified not to require this interface mode:

```
ENABLE_PCAP_PROMISC       N;
```

Fwknop uses libpcap filters to look for remote SPA management packets for performance reasons, not to require looking to all packets.

It is recommended be changed the default port (UDP port 62201):

```
PCAP_FILTER   udp port 29905;
```

Step 6.5. Understanding the knopwatchd.conf file

The knopwatchd.conf file defines the settings driving the fwknop watch daemon (knopwatchd), a process that monitors the other fwknop processes and makes sure they are running (ala watch dog). The main processes are knopmd, which filters out iptables from other kernel messages for port knocking (not used in SPA mode), knoptm that removes the temporary iptables entries (after the default 30 seconds timeout expires) and fwknopd, the main fwknop server.

Step 7. Resolving conflicts between SELinux and fwknopd

By default, FC6 has SELinux enabled. The restrictions imposed by this security subsystem, especially on the Netfilter hooks, do not allow fwknop to dynamically modify the iptables policy, as indicated by the audit denied logs generated when fwknopd starts.

In order to simplify the setup and make fwknop work in FC6, this subsystem can be disabled by modifying the SELINUX variable inside the `/etc/sysconfig/selinux` configuration file from *enforcing* to *disabled*. It is necessary to reboot the system for this setting to take place. This is not a good security practice but it was required to avoid going deeper on all the complexity associated to SELinux in this article.

Step 8. Setting up the server firewall policy

The article examples use a very restrictive iptables policy, where no input traffic is allowed by default, the default action is to drop it all and only allow traffic related with previous sessions. The firewall policy in FC6 is defined in the */etc/sysconfig/iptables* file and should look like this:

```
# cat /etc/sysconfig/iptables
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POLICY - [0:0]
-A INPUT -j POLICY
-A FORWARD -j POLICY
#
-A POLICY -i lo -j ACCEPT
-A POLICY -m state --state
ESTABLISHED,RELATED -j ACCEPT
-A POLICY -j DROP
COMMIT
```

The new policy can be applied by restarting the iptables service and is displayed through the iptables command, using the *-nL* (or *-vnL*) options. See Listing 1.

The fwknop server can be started in FC6 by rebooting the system or by running the service command. Once it is up, it will add its own dynamic firewall rules to a custom iptables chain called `FWKNOP_INPUT`:

```
# service fwknop start (& stop)
# iptables -nL
Chain INPUT (policy ACCEPT)
target     prot opt
source               destination
FWKNOP_INPUT  all  --
0.0.0.0/0            0.0.0.0/0
```

Listing 1. *Restrective iptables policy*

```
# service iptables restart
# iptables -nL

Chain INPUT (policy ACCEPT)
target     prot opt source               destination
POLICY     all  --  0.0.0.0/0            0.0.0.0/0

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
POLICY     all  --  0.0.0.0/0            0.0.0.0/0

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain POLICY (2 references)
target     prot opt source               destination
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           state
                    RELATED,ESTABLISHED
DROP       all  --  0.0.0.0/0            0.0.0.0/0
```

Listing 2. *Port scanning the target system*

```
# nmap -n -sT -p 1-65535 192.168.1.66

Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2007-01-05 04:56 CET
All 65535 scanned ports on 192.168.1.66 are filtered
MAC Address: 00:01:02:09:52:0F (3com)
Nmap finished: 1 IP address (1 host up) scanned in 1356.765 seconds
#
# nmap -n -sU -p 1-65535 192.168.1.66

Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2007-01-05 05:09 CET
All 65535 scanned ports on 192.168.1.66 are open|filtered
MAC Address: 00:01:02:09:52:0F (3com)
Nmap finished: 1 IP address (1 host up) scanned in 1350.906 seconds
```

```
POLICY     all  --  0.0.0.0/
0            0.0.0.0/0
...
Chain FWKNOP_INPUT
 (1 references)
target     prot opt
source               destination
...
```

Step 9. Testing the server ports before sending any SPA packet

The following nmap [7] commands show that, using the previous firewall policy, there aren't any TCP or UDP ports opened on the server. See Listing 2.

Step 10. Running the fwknop client

The fwknop client system (192.168.1.55) only requires the RPM package to be installed. There is no configuration associated to it. If the client tries to connect to TCP port 22, the connections time out due to the firewall default drop policy:

```
$ ssh 192.168.1.66
ssh: connect to host
 192.168.1.66 port 22:
 Connection timed out

$
```

The client can send an SPA packet specifying the server SPA listening port (UDP port 29905), the server IP address (-k) and the source IP address that must be allowed, in this case the one associated to the SPA packet (-s). Once prompted, the user must type the server shared key (defined in the access.conf file). See Listing 3.

At this point, the valid SPA packet is accepted by fwknopd and it modifies the iptables policy to allow the traffic coming from the client IP address (192.168.1.55) going to TCP port 22 for the next 30 seconds. See Listing 4.

At this point, the SSH connection from the client to the server can be established:

```
$ ssh 192.168.1.66
eric@192.168.1.66's password:
```

Listing 3. *Basic port scanning*

```
$ fwknop --Server-port 29905 -s -k 192.168.1.66
[+] Starting fwknop in client mode.
[+] Enter an encryption key. This key must match a key in the file
    /etc/fwknop/access.conf on the remote system.


Encryption Key:

[+] Building encrypted single-packet authorization (SPA) message...
[+] Packet fields:

        Random data: 6156915380424820
        Username:    eric
        Timestamp:   1168010834
        Version:     1.0
        Action:      1 (access mode)
        Access:      0.0.0.0,none,0
        MD5 sum:     Yrpc6vw8yfRu+EymXASCFA

[+] Sending 150 byte message to 192.168.1.66 over udp/29905...
```

Listing 4. *IPtables rule dynamically added*

```
# iptables -nL
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
FWKNOP_INPUT  all  --  0.0.0.0/0            0.0.0.0/0
POLICY     all  --  0.0.0.0/0            0.0.0.0/0

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
POLICY     all  --  0.0.0.0/0            0.0.0.0/0

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain FWKNOP_INPUT (1 references)
target     prot opt source               destination
ACCEPT     tcp  --  192.168.1.55         0.0.0.0/0           tcp dpt:22

Chain POLICY (2 references)
target     prot opt source               destination
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           state
                    RELATED,ESTABLISHED
DROP       all  --  0.0.0.0/0            0.0.0.0/0
```

```
Last login:
Fri Jan  5 16:17:58 2007 from
192.168.1.254
[eric@cucu ~]$ uname -a
Linux cucu.monsters.com 2.6.18-
1.2868.fc6 #1
 SMP Fri Dec 15 17:31:29
EST 2006 i686 i686 i386 GNU/Linux
```

After 30 seconds the newly created rule will disappear and further access won't be allowed anymore.

However, due to the iptables stateful capabilities, the connections already established will be kept. The currently established sessions can be listed through the following command:

```
# cat /proc/net/ip_conntrack
tcp      6 431671 ESTABLISHED src=
192.168.1.55 dst=192.168.1.66 sport=
35608 dport=22 packets=70 bytes=6304
src=
192.168.1.66 dst=192.168.1.55 sport=
22 dport=35608 packets=52 bytes=
7204 [ASSURED] mark=0 secmark=0 use=1
```

The main server log, /var/log/messages, details all the fwknop

actions. The valid SPA packet is received, the access rule is created and the rule is removed after 30 seconds:

```
Jan  5 16:21:52 cucu fwknopd: received
 valid Rijndael encrypted packet from:
 192.168.1.55, remote user: eric
Jan  5 16:21:52 cucu fwknopd: adding
 FWKNOP_INPUT ACCEPT rule for
 192.168.1.55 -> tcp/22 (30 seconds)
Jan  5 16:22:23 cucu knoptm: removed
 iptables FWKNOP_INPUT ACCEPT
 rule for 192.168.1.55 -> tcp/
22, 30 second timeout exceeded
```

When the fwknop client is behind a NAT device, the communication can still be established by allowing the client to resolve its public IP address through the *www.whatismyip.com* Web service.

The client resolves and notifies the server the public IP address, adding it to the encrypted SPA payload, when the `-w` client option (replacing the `-s` option) is used. Other IP address resolution services can be specified.

Step 11. Managing the source IP address from the fwknop client

Apart from creating a firewall rule that allows access from its own IP address to the fixed target port configured in the server side, the fwknop client can be used to dynamically manage the firewall policy remotely. For example, using the `-a` option it is possible to tell the fwknop server to open a rule for a different source IP address.

The following fwknop command, run from the same client (.55), creates an access rule to access TCP port 22 for a third-party system (.254). See Listing 5.

Step 12. Managing the firewall policy from the fwknop client (example2)

The first example focused on providing access to a fixed server port, TCP/22. However, by modifying the fwknop server configuration, it is possible for the client to remotely create rules for other ports, and thus manage the firewall policy remotely.

The client can specify which ports are opened by fwknopd in the firewall policy if the *OPEN_PORTS* directive in the access.conf file is replaced by the *PERMIT_CLIENT_PORTS*. The *access.conf* file should look like this:

Listing 5. *Advanced port scanning*

```
$ fwknop --Server-port 29905 -a 192.168.1.254 -k 192.168.1.66


[+] Starting fwknop in client mode.
[+] Enter an encryption key. This key must match a key in the file
    /etc/fwknop/access.conf on the remote system.

Encryption Key:

[+] Building encrypted single-packet authorization (SPA) message...
[+] Packet fields:

        Random data: 4634543881557025
        Username:    eric
        Timestamp:   1168014145
        Version:     1.0
        Action:      1 (access mode)
        Access:      192.168.1.254,none,0
        MD5 sum:     GhC6gtJ1jEdCtzCbZ4V2Ug

[+] Sending 150 byte message to 192.168.1.66 over udp/29905...
```

```
SOURCE: ANY;
PERMIT_CLIENT_PORTS: Y;
DATA_COLLECT_MODE: PCAP;
KEY: $uper$ecre7;
FW_ACCESS_TIMEOUT: 30;
```

Once the configuration file has been modified, the fwknop server must be restarted for the changes to be applied:

```
# service fwknop restart
```

Using the new configuration, the client can specify the target ports through the `-A` switch. It can open a single port, like the previously used TCP port 22, or it can open multiple ports simultaneously (with a single SPA packet).

This functionality, combined with the previous source IP address option (`-a`), allows to almost completely (the source port and destination address cannot be specified) manage the firewall policy, being possible to create any firewall rule by setting the source IP address and the destination port:

```
$ fwknop --Server-port 29905 -
a 192.168.1.254 -A "tcp/22"
 -k 192.168.1.66
$ fwknop --Server-port 29905 -
a 192.168.1.254 -A "tcp/22,udp/
53" -k 192.168.1.66
```

Conclusion

Port knocking, and specifically SPA, provides advance authentication and authorization capabilities to remotely manage a firewall policy, therefore, reducing the exposure of the services protected by the firewall. They allow granting temporary access to these services at the very moment when it is required. The article focuses on how to deploy fwknop in FC6 to increase the security of SSH and other services.

Although port knocking and SPA are presented as techniques for securing access to remote services, and some commercial IPS products use similar stealthy communication channels to send commands to the inline IPS device, they can also be used by attackers to create stealthy backdoors. Some newer rootkits have already implemented similar methods.

Although the article doesn't cover it explicitly, port knocking and SPA can be used not only to protect incoming traffic (ingress filtering) but outgoing traffic too (egress filtering). For a complete defense in depth solution, you should deploy them in both directions. ●

Fwknop Version and OPEN_PORTS bug

The present article was written using fwknop version 1.0, published on November 05, 2006. All file references (size, timestamps, MD5 values…) reflect this version.

While writing the article, a bug associated with the *OPEN_PORTS* variable was found. When the *OPEN_PORTS* variable is replaced by the *PERMIT_CLIENT_PORTS*, as in Step12, the fwknopd 1.0 server dies due to an undefined value used as a HASH reference. Due to the smart design decision of providing a monitoring process, knopwatchd, all fwknop server processes are restarted after the error condition.

After contacting Michael Rash, the fwknop author, he was impressively responsive and fixed it in less than 2 hours on a Sunday evening. To try the examples throughout this article, it is strongly recommended to download the latest tool version, 1.0.1, published on January 09, 2007.

Kudos to Michael for his great security tools and his support and contributions to the open source community!

About the Author

Raul Siles is a senior Independent Security Consultant specializing in advanced security solutions and prevention, detection and response services in various industries (government, defense, telecom, manufacturing, financial...). Raul's expertise and service offerings includes security architectures design and review, penetration tests, incident handling, forensic and malware analysis, network, system, database and application security assessments and hardening, code security reviews, wireless security, honeynets solutions, intrusion detection/prevention, expert witness, information security management and security awareness and training (through The SANS Institute).

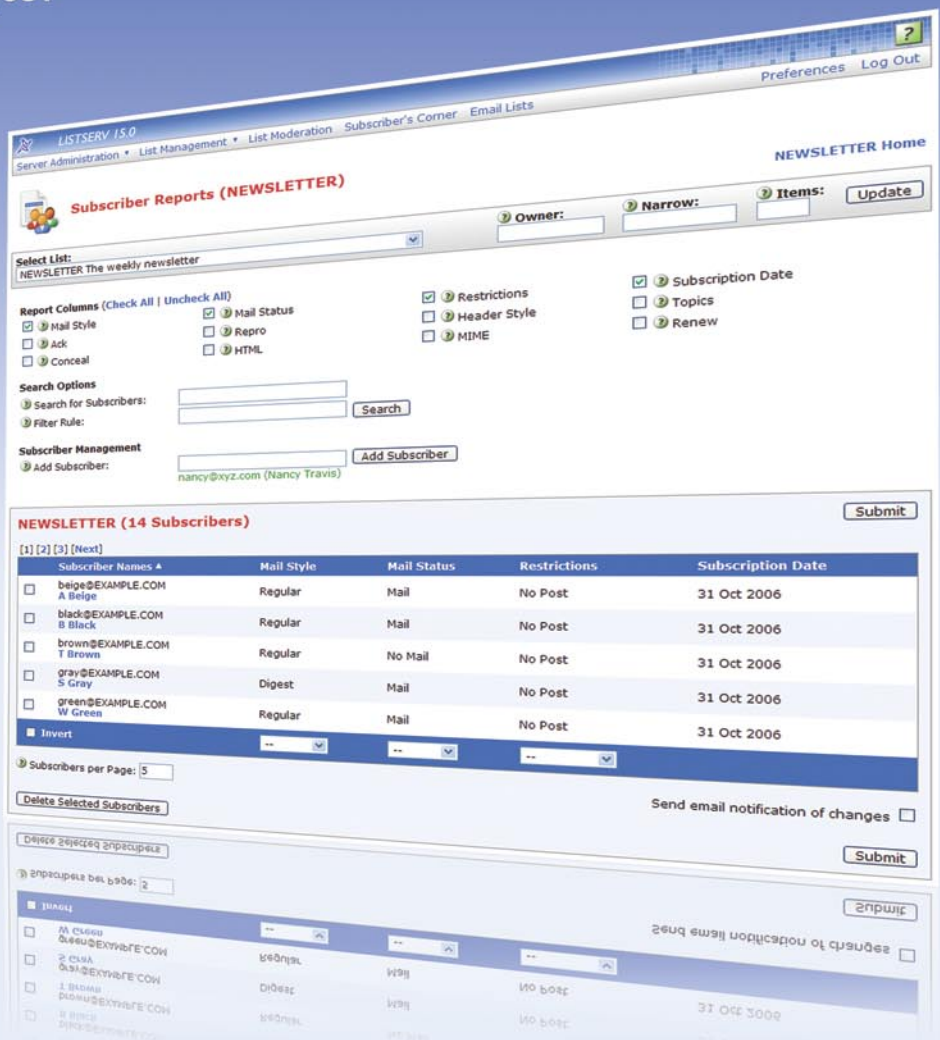
Contact with the author: *raul@raulsiles.com*

LISTSERV® for Linux ...
... an idea to warm up to

The power and performance critical email lists need

The original builder of email communities with its invention in 1986, L-Soft's LISTSERV is today the leading industrial-strength email list manager. LISTSERV offers state-of-the-art deliverability features and is the only email list software with the security of integrated virus protection. LISTSERV can be controlled from anywhere on the Internet through its fully customizable Web interface. LISTSERV is renowned for its flexibility, scalability and performance.

More than 3,700 organizations and businesses worldwide trust LISTSERV for their critical email list needs. ***Isn't it time you turned up the heat on your email challenges?***



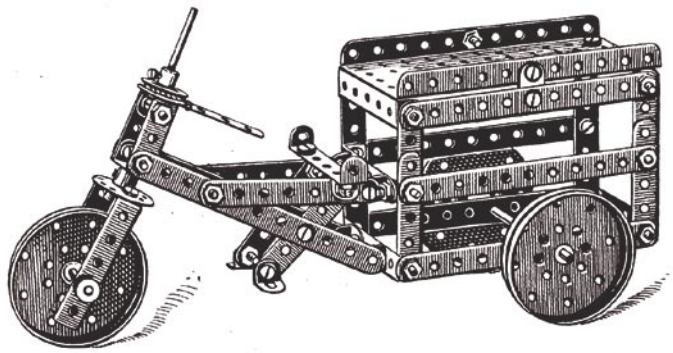
LISTSERV for Linux is available for i386, 64-bit and S/390 architectures.
Virus protection is provided by the integrated F-Secure® Anti-Virus.
LISTSERV Free Edition is available for non-profit hobby users.
LISTSERV is available only from L-Soft.

LISTSERV 20th Anniversary

L-Soft
www.lsoft.com

Highly-Redundant Network Firewall: pf + CARP

Carlos Fragoso Mariscal



A network firewall is a critical piece of our network security infrastructure puzzle. From a security perspective it is important that it offers enough capabilities to fit our protection needs but also from operations point of view it is crucial to provide high-availability and fail-over mechanisms.

There are a lot of aspects and requirements to consider when we need to get and deploy a new network firewall such as: robust *TCP/IP* stack, packet filtering intelligence, network address translation, quality of service, authentication mechanisms, performance, logging capabilities, price, etc. On some environments it is still common to find commodity hardware such as workstations or small servers in combination with a free *open-source* Operating System (OS) and firewall software as a corporate network firewalling approach.

Berkeley Software Distribution (BSD) is a UNIX-derivative OS distributed by the University of California Berkeley born in the 70's. *Linux* and *BSD* could be considered the most popular forks of *UNIX* OS, so both of them have a lot of modern descendant distributions (*distros*), usually also known as *flavours* with some major or minor differences. For example, *OpenBSD* is a *BSD* distro specially focused on proactive security and integrated cryptography.

Since the early days, *BSD* distros provided packet filtering capabilities as a part of the OS kernel natively or as an optional runtime loadable module. *IPFW* and *IPFilter* where historically the most popular ones.

Theo de Raadt, main *OpenBSD* Project community leader, was not so happy about licensing issues related with code modification of Darren Reed's *IPFilter* that pushed him to make a hard and risky move. He removed *IPFilter* from *OpenBSD* source tree and encouraged developers to create a new powerful packet filter for their kernel: *Some people think kernel packet filter is rocket science and it is not!* – Theo de Raadt (OpenBSD Project).

After some initial approaches based on porting IPFW and developing new packet filters, Daniel Hartmeier started to build what he simply named *packet filter* (*pf*). The commu-

What you will learn...

- How to deploy a new network firewall,
- How to implement a high-available network firewall.

What you should know...

- The basic firewall knowledge.

nity was pretty impressed with the first versions so that it was released on *OpenBSD 3.0* in December 2001. It really evolved rapidly on a few months thanks to a group of smart developers and the feedback of the community that improved its stability, performance and features. It even covered some gaps at that time on both open-source and commercial products such as IPv6 protocol firewalling!

After 6 successful years, *pf* has been ported to most *BSD* distros so it could be considered the universal solution for packet filtering on that branch. As you will see later, *pf* features and intuitive configuration is extremely advanced even compared with commercial products and of course it is completely for FREE! A remarkable issue are the built-in quality of service (QoS) features that where merged on 2002 from the *Alternate Queuing Project* (ALTQ).

Availability is always an operational and security critical issue. Most networks are supposed to work even on adverse situations such as link failures, network device problems, etc. Redundancy mechanisms must be put in place to prevent any kind of network downtime.

Data-link layer (L2) redundancy is mostly based on link and device duplication and a protocol named *Spanning Tree Protocol* (STP) that takes care of providing alternative links avoiding loops. Regarding the network layer (L3), where IP addresses live, hosts and network devices need gateways in order to route and forward their packets. What happens then? One solution could be to use a dynamic routing protocol to provide topology change information but it doesn't always solve the problem of a next-hop failure. So what we need is some kind of shared virtual IP under the control of redundant network devices.

Cisco Systems, worldwide network equipment company, was one of the first vendors to focus the next-hop redundancy problem with their *Hot-Standby Routing Protocol* (HSRP) which was obviously patented (US 5.473.599). The *Internet Engineering Task Force* (IETF) worked on an open implementation for vendors, known as the *Virtual Redundancy Routing Protocol* (VRRP) on RFC2338, that didn't include any significant difference from *HSRP*.

The *OpenBSD* Project wanted to provide network redundancy to their OS distribution, and they obviously neither *HSRP* nor *VRRP* protocols could fit their goals. So if they were not happy with that, what's the next move? Déjà vu!!! They designed a cool new protocol named *Common Address Redundancy Protocol* (CARP) released on *OpenBSD 3.4* that had great features such as address-family independence (IPv4/IPv6 transport) and strong cryptographic protection signature based on *SHA1-HMAC* hash function. I love those guys because they even wrote down a song on that: *Redundancy must be free! You see?* – OpenBSD Project

CARP was as successful as it was pf so that the rest of BSD distros ported the protocol to their kernel. Why is it necessary to use a patented protocol in our BSD boxes, if there is a well-designed, robust and completely free protocol? Yes, it doesn't make, sense so *CARP* comes to the rescue !!!

So why pf plus *CARP* is the perfect couple to implement a high-available network firewall? I don't really need to answer that. You will definitively fall in love with them by the end of this article, if you haven't already!

Pick a Distribution and Hardware out

OpenBSD is not the only distro that includes the described pf and CARP combo. You will find a lot of interesting *BSD* flavours out there such as *FreeBSD*, *NetBSD* and *DragonFlyBSD*. On one hand, the really good news is that all of them worked into pf porting during 2003-2004 so it is widely available. On the other hand, although it is currently planned, *DragonFlyBSD* is the only one that has not ported *CARP* yet.

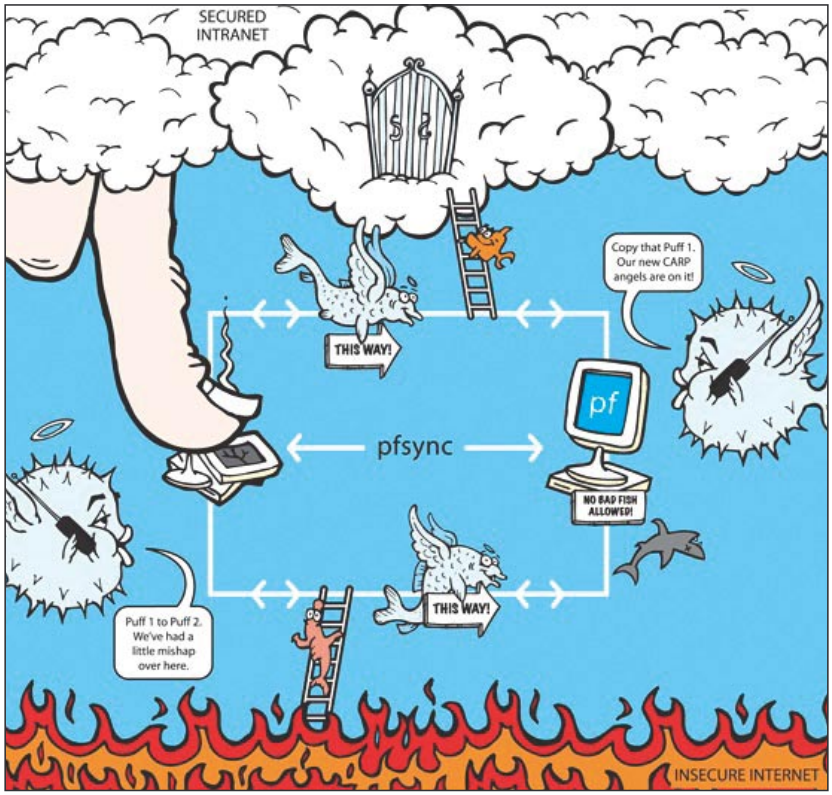


Figure 1. *How pf + carp works*

OpenBSD and *FreeBSD* seems to be the most popular choices between security professionals to deploy their open-source based networks firewalls. Both of them are a good choice but if we carefully consider their base system and goals, *FreeBSD* is a more flexible platform and *OpenBSD* is less functional but more secure by default so it is the best solution for the truly paranoid people like you. Aren't you? Welcome to our exclusive club!

Anyway, do not forget that pf and *CARP* are mainly coded by the *OpenBSD* development team which means that not only patches but also new features will be released first on that platform. Moreover a firewall is not supposed to be a flexible platform so the less functional and secure at system layer it is, the better.

On the last two or three years there has been a new approach which consists on a software package that provides a group of tools and a management interface over pf. Their objective is to make the life of the firewall administrator easier through simplified configuration over a graphical user interface and additional protection, monitoring and reporting tools. `m0n0wall` and *pfSense* are two remarkable examples which really improves the administration experience, specially for non-geek level skilled admins.

The operating system is completely free but hardware it is not, so at this point you are probably wondering or considering which would be the necessary hardware platform? Unless you have a strong logging policy, storage won't be something to worry about so you just need to keep an eye on the *processing power* (CPU), *the main memory and the network interfaces*. Let me tell you that it is quite difficult to exhaust the CPU resources unless the packet rate is so high that the network interface card is constantly generating interruptions.

Figure 2. *The OpenBSD logo*

On BSD-based firewalls, usually the only limit is the *amount of memory* and consider that it grows linearly based on the traffic state information. A 64 MB RAM device can handle about 65000 states without any problem. Some performance tests published out there state that pf can handle about 16000 packets per second (pps) without loss if the ruleset is not longer than 20-30 rules. On a 100 Mbps interface with 84-byte Ethernet frames, it can handle about 148809 pps.

The low profile required to run a network firewall opens the door to a huge amount of embedded devices that could fit our needs such as: PC-*Engines WRAP*, *Soekris NET*, *Mini-VIA ITX*, etc.

The main advantage of those platforms is their small 64-128 MB Compact Flash memory where the firewall software is stored. This is really convenient because they are more faulty tolerant than magnetic drives such as an standard IDE/SATA hard drive.

Some other approaches are based on using a read-only media to boot the OS and a removable writable media such as a USB pendrive or a floppy disk to store the configuration.

`m0n0wall` and *pfSense* have especial embedded versions that could run on light devices. For instance, m0n0wall is a reduced version of *FreeBSD 4.11* that can be managed through a web interface and incorporates noticeable things such as a complete configuration of the system on an *eXtensible Markup Language* (XML) file. pfSense is another top choice that is similar to m0n0wall but it is based on FreeBSD 6.1.

Case Study

Big Banana Company (BBC) is a medium enterprise that had severe availability problems on their network infrastructure during the last year. *Big Banana*, main BBC product's wasn't very successful so the business results were very poor. That means that although they need a strong change, there isn't any money left for investment on IT.

BBC Chief Technical Monkey, known as Armando, decided to hire a young but enthusiastic firewall administrator called Gustavo. That lovely monkey was quite well-known inside the BSD community because of his amazing skills and level of paranoia.

The existing network infrastructure was based on a single two-legged firewall between two private IP address ranges: 192.168.0.0/24 (external side) and 10.10.0.0/24 (internal side).

Current BBC's ISP, called CocoNet, was providing them a generous IP public range: 1.0.0.0/25. CocoNet decided to offer a redundant Internet connection through a second CisCoco router using the infamous HSRP protocol.

Gustavo made a proposal about using a couple of old desktop computers, upgraded with a couple of *FastEthernet* NIC cards, to deploy an open-source approach with pf and CARP. Armando was really happy with that, specially because that solutions seemed to solve their problem for free.

Let's help Armando and Gustavo on their firewalling quest!

Operating System set-up

Installation and hardening of an Operating System (OS) is completely out-of-scope of this article so we won't definitively go through the details. Anyway there are some aspects that we must certainly take into consideration:

OS installation must be performed on a preproduction controlled environment, a minimal installation and latest patching level of the most

recent OS version is required and secure remote access with strong authentication mechanisms must be provided. The most important thing is not to provide a secure baseline at the beginning but to guarantee that it is the same during the complete system life-cycle.

Install a minimal OpenBSD 4.0

Local user accounts for each firewall administrator and operator will be added to the system using tools like *useradd*. Administrator and operator group creation with *groupadd* is highly recommended even if only one person belongs to any of them. Users are mapped with each group using the usermod tool. Creation of 'fwadmin' and 'fwoper' groups:

```
# groupadd fwadmin
# groupadd fwoper
# grep fw /etc/group
fwadmin:*:1009:
fwoper:*:1010:
```

Creation and group-mapping of armando and gustavo users:

```
# useradd -m armando
# useradd -m gustavo
# usermod -G fwadmin armando
# usermod -G fwoper gustavo
# passwd armando
# passwd gustavo
```

Do not forget to encourage users to change their passwords not only the first time but on a frequently basis.

Personal accounts is the best way to keep track of the users and groups is the escalable way to give them privileges. Authorization of which privileged commands are allowed for each group is possible using the *sudo* tool. Firewall administrators usually have higher control over the box than operators so we must identify and configure necessary commands to allow them to perform usual operation tasks.

```
Configure group privileges on /etc/
sudoers using visudo
Cmnd_Alias PF = /sbin/pfctl
Cmnd_Alias SYSTEM = /usr/sbin/reboot,
 /usr/sbin/shutdown
Cmnd_Alias NET =
tcpdump ifconfig netstat
%fwadmin ALL = PF, SYSTEM, NET
%fwoper ALL = PF
```

If you want the firewall admin to be able to switch to root, you must include him into the wheel group.

Secure SHell (SSH) is the preferable in-band remote management mechanism to access the firewall remotely. If you didn't enable it during installation time you could enable and configure it appropiately.

```
Enable the SSH daemon on /etc/rc.conf
sshd_flags=""
Configure SSH on /etc/ssh/sshd_config
Port 1234
Protocol 2
ListenAddress 10.10.0.2
PermitRootLogin no
PubKeyAuthentication yes
AllowTCPForwarding no
X11Forwarding no
PrintMotd yes
PrintLastLog yes
Banner /etc/issue.net
AllowUsers armando gustavo
```

An alternative port will obfuscate SSH presence and further hardening will be based on limiting the users, disable root login and map the daemon to the internal IP address. Warning banners before (*/etc/issue.net*) and after (*/etc/motd*) logging-in are highly recommended.

Out-of-band management through the serial communications port is desirable but requires a special device such as a terminal console server.

Last but not least, system logging is local by default but could be forwarded to an external centralized server. It is your choice to decide which levels of logging do you want to send out of the box.

```
Configure syslog at /etc/syslog.conf
*.*       @loghost
```

For an advanced network logging approach you could deploy *syslog-ng* (next-generation) software package that provides extra features such as reliable TCP transport. Please be paranoic !!! *SSLTunnel* tool either built-in *IPSec* will definitively help you to protect logging information over the network.

High-available Network Configuration

It is time for setting up the firewall network configuration, that means not only IP addresses but also any

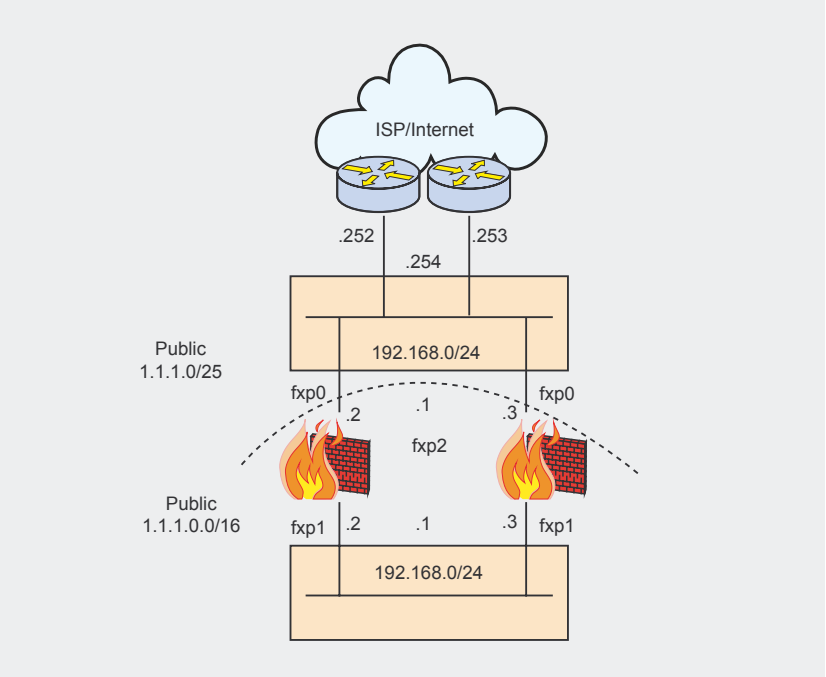


Figure 3. *The case study*

related name resolution issue. Firewalls are not supposed to perform DNS name resolution mainly because that slows them down, so any name resolution they are supposed to know must be defined locally at hosts file.

- Set the name of the firewall at */etc/myname*,
- Configure */etc/hosts* file appropiately,
- Delete any information on */etc/resolv.conf*.

An IP address has to be configured on each firewall interface to make them reachable and a default gateway must be set up so that the firewalls knows where to send traffic to non-defined networks. In our case it is the virtual IP address of the CisCoco routers provided by the ISP:

- Configure default gateway at */etc/mygate*

```
192.168.0.254
```

- Configure IP address for each interfaces at */etc/hostname.fxp0* inet 10.0.0.2 255.255.255.0 NONE

One file *hostname.interface* has to be configured for each interface (fxp0, fxp1 and fxp2) and after that configuration must be applied to the system.

- Run *sh /etc/netstart* to apply the network configuration

IP Forwarding is the process of making IP packets flow through the device from an interface to another using its routing table to make path decision, that is what makes the difference between an end-system and a network device. This feature it is not enabled by default on most OS.

- Enable IPv4 forwarding for boottime on */etc/sysctl.conf*

```
net.inet.ip.forwarding=1
```

- Enable IPv4 forwarding instantly using the *sysctl* command:

```
sysctl -a | grep ip.forwarding
sysctl -w net.inet.ip.forwarding=1
```

In case it was necessary we can configure any static network routing information that could be necessary at */etc/rc.conf.local*.

At this point we are ready to set up the Common Address Redundancy Protocol (CARP) to enhance our infrastructure. CARP is an IP protocol (112) that provides a virtual MAC and a virtual IP to a group of hosts (cluster), one of them will act as MASTER (active) and the other one as BACKUP (passive). Announcements are constantly sent using multicast packets so that the state of the MASTER is known.

- Set a virtual carp interface for external network */etc/hostname.carp0*

```
inet 192.168.0.1 255.255.255.0
192.168.0.255 vhid1 pass <password>
carpdev fxp0
```

- Set a virtual carp interface for internal network */etc/hostname.carp1*

```
inet 10.0.0.1 255.255.255.0 10.0.0.255
vhid2 pass <password> carpdev fxp1
```

- Run *sh /etc/netstart* to apply the network configuration

A password (*shared-secret*) is setup to protect CARP cryptographically against spoofing attacks.

Some other cool additional features of CARP include *preemption* that allows a firewall to recover MASTER state after a failure based on priorities, and *arpbalance* as a load-balancing feature.

Before going on into the packet filtering stuff it is extremely important to test network connectivity and configured CARP redundancy so that we are pretty sure that it works fine.

Use the *ping* tool from the firewall and router sides and do not bother about disconnecting some cables, *ifconfig <if> down* will have the same effect, or even why not rebooting one of the firewalls? Using *ifconfig* tool over the carp interfaces will show you their redundancy state.

Basic packet filtering configuration

Packet filtering process consists on inspecting packets based on layer 3 (IPv4, IPv6) and layer 4 (TCP, UDP, ICMPv4, ICMPv6) headers. The most used criteria is source and destination IP addresses and TCP/UDP ports.

Tabela 1. *Glossary*

ALTQ	ALTernate Queuing
BSD	Berkley Software Distribution
CARP	Common Address Redundancy Protocol
HMAC	Keyed Hash Message Authentication
HSRP	Hot Standby Routing Protocol
IETF	Internet Engineering Task Force
ipfw	IP Firewall
NAT	Network Address Translation
OS	Operating System
pf	Packet Filter
pps	Packets Per Second
SHA1	Secure Hash Algorithm 1
VRRP	Virtual Router Redundancy Protocol
XML	Extensible Markup Language

GlobalTrust

Enigma Lite Desktop Edition

Company documents and personal data are an important patrimony that you have to protect from thefts, misuses and from the access of unauthorized users, both inside and outside the organization. Moreover, most personal and company information are stored in the form of files and electronic documents and are transferred through e-mail messages or geographic networks.

Enigma Lite Desktop Edition suite offers a modular and scalable solution to secure document storage and transfer; it meets not only the requirements of single users, by protecting their desktop or mobile computer, but also those of great organization local and geographic networks. Enigma allows you to implement the strongest security policies and to ensure the protection and the privacy of files, e-mails, databases and of the whole document flow.

WHY ENIGMA? (LEGAL CHANGES):

- In Italy new laws and rules have been introduced, such as D.Lgs 196/2003, also known as The Privacy Code, imposing to protect personal data from such risks as destruction, loss or unauthorized access: this is the reason why Institutions, great companies, SMEs and professionals have to comply with the minimum security measures concerning personal data processing;
- According to the Privacy Code, digital sensitive data (e.g. judicial data, health related information) must be encrypted or made inaccessible through the use of the best technology solutions currently available on the market;
- The Code also provides for several strict penalties in case of breach of the law.

WHY ENIGMA? (ITALIAN MARKET)

- In Italy there is a great demand for easy-to-use solutions able to protect the electronic format documents and personal data that are stored in personal computers (especially notebooks) and in organization archives of such domains as:
  - Military (e.g. Encrypting Classified Documents);
  - Financial (e.g. No repudiation Digital Messaging);
  - ISP (e.g. Secure Web Transactions and Archiving);
  - System Integrator (e.g. Secure Document Management Solution);
  - University and others institutes of instruction;
  - Company;
  - Retailers (SW/HW);
  - and others;
- But the offer of competing products and solutions is inadequate, since the existing ones (BestCrypt, Signo and mainly PGP) are neither well known nor spread yet.

ENIGMA FUNCTIONALITY

A Stronger Internal Security and Privacy Enigma Lite Desktop Edition is the best solution to manage personal data in accordance with Privacy laws and provisions (D.Lgs. n. 196/2003). Enigma Lite Desktop Edition allows you to centralize all operations involving personal and corporate document security: digital signatures, cryptography, secure storage, time stamping, secure destruction, authenticity verification and digital certificate validation.

IT IS INTEGRATED WITH WINDOWS SYSTEMS AND MICROSOFT OFFICE

Enigma Lite Desktop Edition is a solution designed for Microsoft operating systems: the Desktop Document Security components are completely integrated with Windows client operating systems and the server software components are installed on MS Windows Server systems.

IT IS INTEGRATED WITH PKI APPLICATIONS

Enigma Lite Desktop Edition uses the most advances cryptography mechanism and is compatible with X.509 and SSL v.3.0 standards, allowing to use and to manage digital certificates and identities. Moreover, Enigma is compatible with Common Access Card (CAC) and PKCS#11 (Cryptographic Token Interface Standard) standards for the use of SmartCards and USB Token.

ENIGMA LITE DESKTOP EDITION

- Product page: http://www.globaltrust.it/products/enigmalitedesktopedition.aspx; http://www.enigmatrust.org;
- Product Buy: https://www.globaltrust.it/software_buy/enigmalitedesktopedition_12.aspx;
- Web Site: http://www.globaltrust.it.

ABOUT GLOBALTRUST

GlobalTrust is a Certification and Registration Authority, recognized at worldwide level, able to issue all classes of certificates and all types of digital security technology; using the more advanced security techniques, authentication, verifies and insurance covering. GlobalTrust is an Indipendent Authority in the protection from the attacks coming from the Net. Thanks to its technology, GlobalTrust allows organizations of all sizes to secure e-business transactions, always keeping present the importance of a good report quality/price. ●

Packet filter (pf) is not enabled by default on an *OpenBSD* system so we must configure the system to enable it at boot time.

- Enable pf at */etc/rc.conf* file:

```
pf=YES
pf_flags=""
pf_rules="/etc/pf.conf"
```

A minimal configuration is recommended to begin with the deployment that, as a recommended practice, must `deny all traffic by default` except those we have defined as necessary until now (CARP).

- Edit *pf.conf* file to configure ext_if="fxp0"

```
int_if="fxp1"
sync_if="fxp2"
block in all
block out all
pass quick on $ext_if
 proto carp keep state
pass quick on $int_if
 proto carp keep state
```

- Enable pf inmediatly through the pfctl command

```
pfctl -e -f /etc/pf.conf
```

As you can see, we have defined three variables (ext_if, int_if and sync_if), known as macros, to name the physical interfaces through the configuration. Next to that you will see the first rules that deny all inbound and outbound traffic and finally rules to allow CARP. Pf evaluates the rules from top to bottom but it is different from other firewalls because *the last matching rule wins*! The only way to overcome that rule is to use the *quick* option that cancels further rule processing. Pf configuration file consists on different ordered sections described below:

- Macros,
- Options,
- Scrub,
- Network Address Translation,
- Filter rules.

pf doesn't provide any way of synchronizing the configuration file between both firewalls, so the firewall administrator is responsible for setting up that mechanism. The most common way is deploying a file synchronization tool such as *rsync* over SSH.

Failover capability in our firewall will be provided by a protocol named *pfsync*, introduced on OpenBSD 3.4, which takes care of state table synchronization among firewalls of the same redundancy group. A pseudo-device interface *pfsync0* is provided to pass state table changes and needs to be mapped with a physical one in order to make it work. That information is sent over the interface using IP multicast packets (group 224.0.0.240) unless a synchronization peer (*syncpeer*) is set.

A dedicated trusted network between both firewalls with a crossover cable is extremely recommended because the *pfsync* protocol provides neither any kind of authentication nor cryptographic protection.

Synchronization peer set up and IPSec communication protection could provide further security to overcome mentioned pfsync main weakness.

- Create /etc/hostname.pfsync0:

```
up syncpeer 172.16.0.2 syncdev fxp2
```

- Permit pfsync traffic on synchronization interfaces at pf.conf file:

```
pass quick on $sync_if proto pfsync keep state no-sync
```

- Reload pf configuration through pfctl tool:

```
pfctl -f /etc/pf.conf
```

Network Address Translation

Network address translation, known as NAT, is the mechanism to map IP addresses between networks (RFC1631) based on different issues such as private address space (RFC1918) visibility through public addresses, overlapping networks, etc. There are different ways of mapping IP addresses: one to one (static), many to one (hide) or many to several (pool). Even we could use layer-4 information like TCP/UDP ports to achieve traffic redirection.

Let's do as an example a simple static NAT mapping of BBC's main DNS server and allow navigation of Armando and Gustavo workstations through a hide-NAT rule.

- Add necessary variables and *one to one* rule at pf.conf file:

```
dns_server_int="10.10.0.10"
dns_server_ext="1.1.1.10"
binat on $ext_if from $dns_server
 to any -> $dns_server_ext
```

- Add necessary variables and 'one to one' rule at *pf.conf* file:

```
armando_int="10.10.0.40"
```

On the Net

- [1] *Documentation and Frequently Asked Questions*, OpenBSD Project; URL; *http://www.openbsd.org/faq/*,
- [2] *The OpenBSD Packet Filter (pf)*, OpenBSD Project; URL; *http://www.openbsd.org/faq/pf/*,
- [3] *Firewall Failover with pfsync and CARP*, Ryan McBride (Countersiege); URL: *http://www.countersiege.com/doc/pfsync-carp/*,
- [4] *Firewalling with OpenBSD's PF packet filter*, Peter N.M. Hansteen; URL; *http://www.bgnett.no/ peter/pf/en/long-firewall.html*,
- [5] *Redundant firewalls with OpenBSD, CARP and pfsync*, Daniel Mazzochio; URL; *http://www.kernel-panic.it/openbsd/carp/*,
- [6] *Basic pfctl control*, RDRS; URL: *http://www.rdrs.net/document/pfctl_overview.txt*.

```
gustavo_int="10.10.0.41"
workstations_pub="1.1.1.20"
nat on $ext_if from
{$armando_int, $gustavo_int }
 to any -> $workstations_pub
```

- Reload pf configuration through *pfctl* tool:

```
pfctl -f /etc/pf.conf
```

As you could see the syntax is quite simple, you just need to indicate which is the interface where the translation is performed and based on described source and/or destinations which is the source IP address used. For additional information take a look at the NAT section at OpenBSD PF's FAQ.

Ruleset definition

Packet filtering ruleset is a written implementation of the network security policy, because by just using a group of rules we define which traffic is allowed or denied. It is very important that we really have enough knowledge not only about the protocols themselves but also about which is the traffic in our network.

Pf has a very simple language to describe rules that enables a firewall administrator to implement the policy on a very intuitive way:

```
action [direction] [log] [quick]
[on interface] [af] [proto protocol]
[from src_addr [port src_port]]
[to dst_addr [port dst_port]]
[flags tcp_flags] [state]
```

Let's see an example of allowing BBC's main SMTP server to send and receive mails from the outside.

- Add necessary macros and rules at *pf.conf* file:

```
mail_server_int="10.10.0.10"
mail_server_ext="1.1.1.5"

# inbound mail
pass in on $ext_if from any to
$mail_server_ext port smtp keep state
pass out on $int_if from any to
$mail_server_int port smtp keep state

# outbound mail
pass in on $int_if from
 $mail_server_int
to any port smtp keep state
pass out on $ext_if from
$mail_server_int
to any port smtp keep state
```

One of the main pf abilities is *keeping state* that means it is stateful-inspection capable. That embedded-intelligence makes it easier to evaluate the packets and dramatically improves the performance because previously established flows are not evaluated against the ruleset.

Please check OpenBSD PF's FAQ to see further information about how flexible a rule can be such as the use of IP networks, port ranges, options, etc.

- Use the *pfctl* tool to see current rules:

```
pfctl -s rules
```

Operations

Once the firewall is deployed and is considered in operation, we must keep care of it not only as a part of the system's lifecycle but also to troubleshoot problems and to keep the rules updated.

Logging is one of the coolest features on pf. Why? Because it doesn't log only basic information about a matching packet, it logs the packet itself! Packets are written to the file */var/log/pflog*, on PCAP format, if the last pf rule they match includes the log keyword. So add log keyword to those rules that catch packets you want to log (pass, block), log keyword will just log packets that make state while log-all will log all packets.

- Read logged packets using tcpdump tool:

```
tcpdump -n -e -ttt -r /var/log/pflog
```

- Read realtime logged packets using tcpdump tool:

```
tcpdump -n -e -ttt -i pflog0
```

pflog can be rotated with the system tool newsyslog configured as a automated task (*crontab*). It mainly creates a new file, rotates the files between them and compressing the old ones on ZIP format. Rotating policy could be adjusted at */etc/newsyslog.conf* file. *pflog* files could be automatically downloaded by another host for later analysis just using SCP (SSH Secure Copy).

Logged packets are extremely useful not only for troubleshooting but also for network forensic purposes. But be careful, if you log too much information you should carefully consider your storage approach.

pf is fully controlled using the previously introduced *pfctl* tool, it has several parameters to show current state of configuration, tables, performance, etc.

- Show pf realtime information using pfctl:

```
pfctl -s info
```

I extremely recommend configuring some system aliases to ease your daily pf administration work of usual tasks:

- Configure aliases on *.profile* of the user:

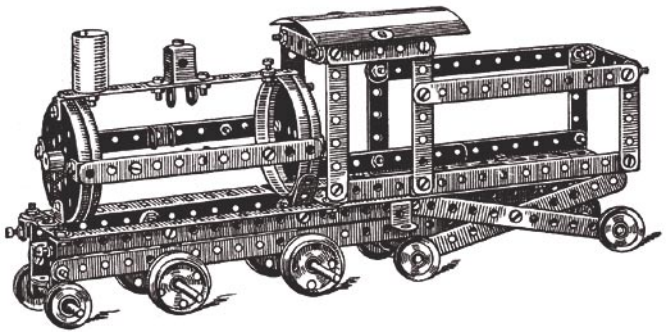
```
alias pflog='sudo tcpdump

 -n -e -ttt -i pflog0'
alias pfconf='sudo vi /etc/pf.conf'
alias pfchk='sudo pfctl
 -n -f /etc/pf.conf'
alias pfload='sudo pfctl -f /etc/
pf.conf'
alias netext='sudo tcpdump -nnpi fxp0'
alias netint='sudo tcpdump -nnpi fxp1'
alias netsync=
'sudo tcpdump -nnpi fxp2'
```

For instance *pfctl -n -f /etc/pf.conf* checks the configuration without loading it which is quite useful for identifying syntax errors. Finally do not forget to install any software package that could help your pf's managing and monitoring experience better such as pfstat, hatchet, *pftop*. ●

Linux Netfilter – Packet Mangling and Applications

Lucian Gheorghe



What is packet mangling? As the term mangling might mislead people to conceive it as malicious, packet mangling is not like that at all. Packet mangling refers to the process of intentionally altering data in IP packet headers before or after the routing process.

If you are familiar with *Network Address Translation* (NAT) and/or *Port Address Translation* (PAT or NAPT), then you know that using iptables we can alter the source and destination IP addresses in an IP packet header or the source and destination port in the TCP or UDP headers. This is a type of mangling that is done in the nat table of netfilter/iptables and it's referred to as NAPT (*Network Address and Port Translation*) which is a totally different subject.

The netfilter mangle table

The mangle table has 5 chains named PREROUTING, INPUT, FORWARD, OUTPUT and POSTROUTING.

By looking at the following simplified flowchart, we can see how the Linux kernel looks up filtering rules in those five chains (Figure 1).

The rules in PREROUTING chain of the mangle table are analyzed by the kernel when a packet comes in, before the routing process takes place.

If the packet is not for the router (is destined to a host behind it), the kernel looks up the rules in the mangle table FORWARD chain and afterwards the mangle table POSTROUTING chain.

If the packet is destined to the router then the mangle table INPUT chain is analyzed. When the response is generated, the kernel looks up the rules in the OUTPUT chain of the mangle table, and last the POSTROUTING chain of the mangle table. On those chains, all iptables operations are possible. It is not recommended to do packet filtering in the mangle table. Even if you are not restricted to drop packets in the mangle table, it should be only for rules that mangle IP packets.

A particularity of the mangle table is that, unlike the filter or nat tables, all rules are analyzed, even if the packet is matched against

What you will learn...

- Why and how to mangle IP packets,
- What is packet mangling.
- How to do source routing ussing packet mangling

What you should know...

- Basic iptables syntax
- Basic IP concept

one rule. This functionality of the mangle table was needed, because you might want to alter the same IP packet in more ways, so, unlike the other tables, if a match is found against one rule in a chain the kernel doesn't stop there and tries to match the packet against all the other rules in that chain.

This is one of the major reasons for which packet filtering in the mangle table is strongly discouraged.

What kind of data can I alter in the mangle table?

I'm not going to go too deep in the theory of what an IP packet contains, but the header of all IP packets look like this Figure 2.

For many of the fields in the packet header we won't have any applications if we modify them, but the most interesting fields are:

- ToS – the 8 bit Type Of Service field,
- TTL – the 8 bit Time to live field.

And those are the only fields of the IP packet that can be modified in the mangle table.

Besides altering the TOS and TTL fields, there is another operation that can be done in the mangle table, that doesn't actually alter any data in the IP packet. This operation is called *marking* packets using an internal nfmark (netfilter mark) value that is usable only on the machine that marks the packets this way. The nfmark is lost when the packet leaves the machine (other routers will not be aware of this mark). Nfmark is also referred to as fwmark (firewall mark).

How and why do I mangle IP packets?

How to mangle IP packets is a simple question with a simple answer – *using iptables*. The difficult question is *why*.

I would like to start with the nfmark option, but, before that, let's build a simple test scenario that we'll use for all our cases.

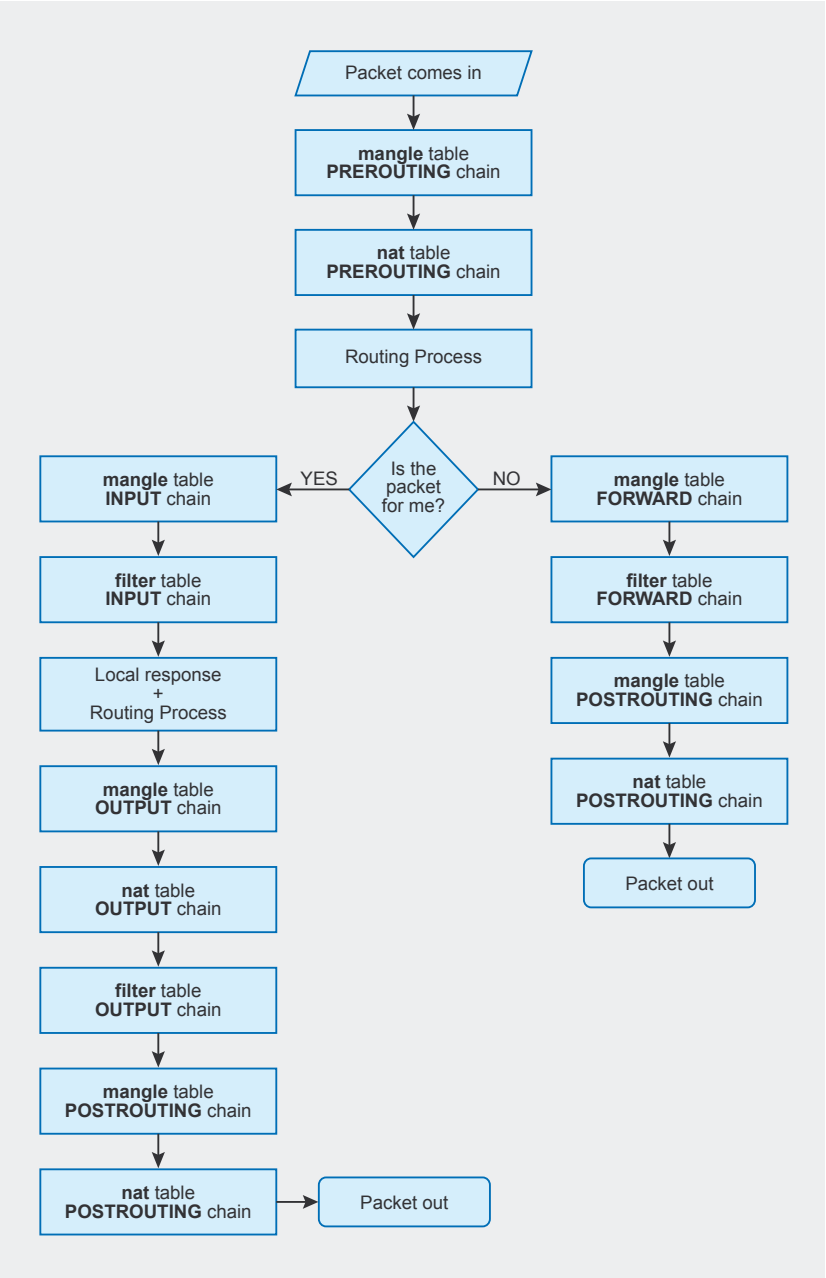


Figure 1. *Filtering rules in five chains*

Let's say we have a Linux router with 2 Internet connections and an internal network with users for which we provide Internet connectivity standing behind our Linux box, like in the following diagram Figure 3.

Now, if we don't run BGP with the 2 providers then we'll only have a default route (let's say through ISP1). We will do NAT for our internal network `192.168.1.0/24`, but what we want to do is to use ISP2 for all the web and ftp traffic, and ISP1 for everything else. To do that, we must find a way to send the web and ftp packets on eth2 through `2.2.2.1` and leave the rest of the packets going through our default route.

Many of you will think that using different SNAT rules will help (like SNAT www packets to `2.2.2.2`), but it won't because you will rewrite the source IP address to `2.2.2.2` and send the packet to ISP1 which will most probably drop it. That's why we will use MASQUERADE so the packets will be rewritten with the source IP of the interface on which we send the packet, but we must

find a way to send different types of traffic on different interfaces.

For that, we'll use `iproute2` in conjunction with netfilter/iptables. First, we will set an internal nfmark for web and ftp packets using the following iptables commands:

```
iptables -t mangle -A PREROUTING -p
 tcp --dport 80 -j MARK --set-mark 1
iptables -t mangle -A PREROUTING -p
 tcp --dport 20:21 -j MARK --set-mark 1
```

Both rules will be added in the PREROUTING chain of the mangle table so they can be matched against the IP packets before the routing decision is made. The first rule matches TCP packets with destination port 80, the second rule matches TCP packets with destination ports 20 and 21. The target in this case is MARK which will set an internal nfmark of 1 for those packets. Now, we will use `iproute2` to send the packets on `eth2`:

```
# echo 200 isp2 >>
/etc/iproute2/rt_tables
# ip rule add fwmark 1 table isp2
# ip route add default via 2.2.2.1
 dev eth2 table isp2
# ip route flush cache
```

If you will do IP rule list you will see our rule:

```
# ip rule ls
0:      from all lookup 255
32765:  from all fwmark 0x1 lookup isp2
32766:  from all lookup main
32767:  from all lookup default
```

And everything else will go on the default gateway. More information on iproute2 and nfmark applications can be found on *http://lartc.org* and in my book *Designing and Implementing Linux Firewalls and QoS using netfilter, iproute2, NAT and l7-filter* at *http://www.packtpub.com/linux-firewalls/book*. Time to Live field alteration is done using the TTL target in iptables

```
root@router:~# iptables -j TTL --help
… some lines missing…
TTL target v1.3.1 options
--ttl-set value   Set TTL to
<value 0-255>
--ttl-dec value
Decrement
TTL by <value 1-255>
--ttl-inc value
Increment
TTL by <value 1-255>
```

There are many applications for modifying TTL in IP headers, but an interesting application in our example scenario would be to limit the distribution of Internet services by the clients in the `192.168.1.0/24` network.

Let's say that we are only allowing the clients in the private network behind our Linux router only to have 1 computer that is allowed to be connected to the Internet. If they use a SOHO router each of them can have multiple computers connected to the Internet, even if we don't allow them to. To make sure that they don't use proxy servers or SOHO routers, we can alter the Time to Live field in the IP packet and set all packets going to them with a TTL value of 1:

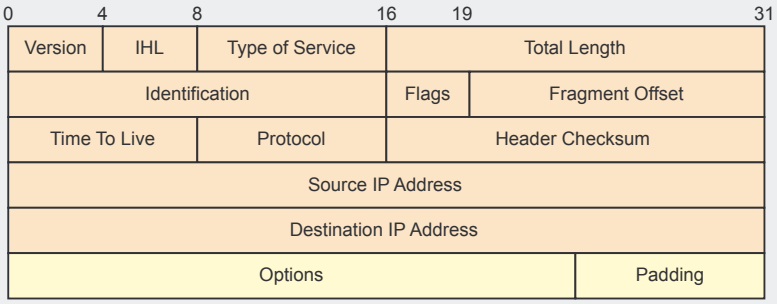


Figure 2. *The header of all IP packets*

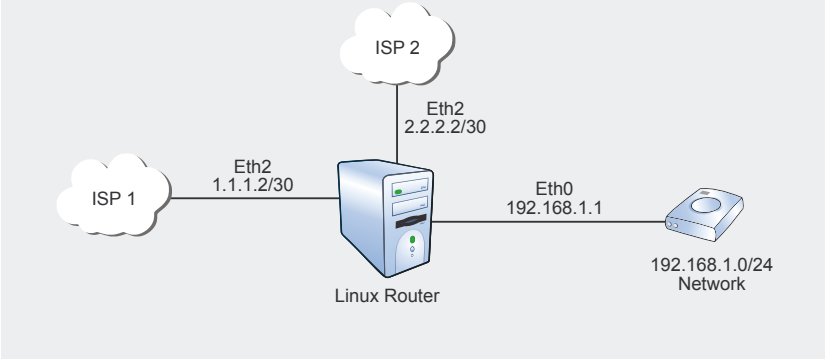


Figure 3. *The Internet connectivity*

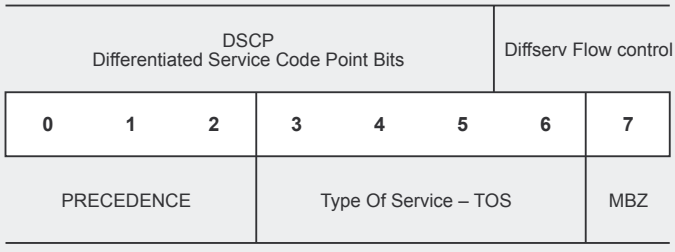


Figure 4. *The TOS byte*

About the Author

Lucian Gheorghe has just joined the Global NOC of Interoute, Europe's largest voice and data network provider. Before Interoute, he was working as a senior network engineer for Globtel Internet, a significant Internet and Telephony Services Provider to the Romanian market He has been working with Linux for more than 8 years putting a strong accent on security for protecting vital data from hackers and ensuring good quality services for Internet customers. Moving to VoIP services he had to focus even more on security as sensitive billing data is most often stored on servers with public IP addresses. He has been studying QoS implementations on Linux to build different types of services for IP customers and also to deliver good quality for them and for VoIP over the public Internet. Lucian has also been programming with Perl, PHP and Smarty for over 5 years mostly developing in-house management interfaces for IP and VoIP services.


```
iptables -t mangle -A POSTROUTING
 -o eth0 -j TTL --ttl-set 1
```

Of course this won't give you 100% protection to redistribution of IP services, but at least the users won't be able to use SOHO routers or socks proxies, because the packets will arrive in their SOHO router with a TTL of 1. Their SOHO router will decrement the TTL value before passing the packets further, but TTL will become 0, so, instead of passing the packets to the destination machine, the SOHO router will drop them.

The *Type of Service* field is 8 bits long and its primary usage is for prioritization of data (QoS). Altering the TOS byte has many applications in real life, but we have to be careful about using the TOS target of iptables because the TOS target modifies the TOS bits of the TOS byte and not the whole TOS byte. The TOS byte looks like this Figure 4.

Using the TOS target of iptables, we can only alter the TOS bits (bits 3,4,5,6 of the TOS byte).

```
root@router:~# iptables -j TOS --help
… some lines missing…
TOS target v1.3.5 options:
  --set-tos value    Set Type
of Service field to one of the
following numeric or descriptive
values:
Minimize-Delay 16 (0x10)
Maximize-Throughput 8 (0x08)
Maximize-Reliability 4 (0x04)
Minimize-Cost 2 (0x02)
Normal-Service 0 (0x00)
```

Example – set TOS to Maximize-Throughput for outgoing ftp-data:

```
iptables -t mangle -A POSTROUTING -p
 tcp --sport 20 -j TOS --set-tos 8
```

Many applications set the TOS value of the IP packets so if we want to send data that needs high throughput and minimum cost to ISP2, we can do the following:

```
iptables -t mangle -A PREROUTING -m
tos --tos 8 -j MARK --set-mark 1
iptables -t mangle -A PREROUTING -m
tos --tos 2 -j MARK --set-mark 1
```

The 2 rules above mark the packets having the TOS value 8 and 2 with a nfmark of 1. We already defined iproute rules to send the packets with a nfmark of 1 through ISP2, so we achieved what we wanted. Another way to modify the TOS byte is using the DSCP target:

```
root@router:~# iptables -j DSCP --help
… some lines missing…
DSCP target options
--set-dscp value   Set DSCP field
in packet header to value
This value can be in decimal (ex: 32)
or in hex (ex: 0x20)
--set-dscp-class class   Set the DSCP
field in packet header to the
value represented by the
DiffServ class value.
This class may be EF,BE
or any of the CSxx
or AFxx classes.
These two options are
mutually exclusive !
```

Example – set DSCP 32 to packets arriving on interface `eth0`:

```
iptables -t mangle -A PREROUTING -i
 eth0 -j DSCP --set-dscp 32
```

DSCP is explained in more detail in RFC2474 which can be found at *http://www.rfc-editor.org/rfc/rfc2474.txt*. The TOS byte is used mainly for QoS and bandwidth shaping. The TOS byte is not only a way of marking IP packets, but the Linux kernel uses it to make decisions on how to prioritize data, even if you don't configure anything in this matter. Linux has a default queuing algorithm called `pfifo_fast` that uses 3 bands that have different priorities and packets are inserted in those bands depending on their TOS byte.

My book *Designing and Implementing Linux Firewalls and QoS using netfilter, iproute2, NAT and l7-filter* (*http://www.packtpub.com/linux-firewalls/book*) contains a more detailed explanation on the process of how Linux queues data and many examples using TOS, DSCP and nfmark to do bandwidth shaping and to prioritize data.

How does packet mangling relate to firewalls?

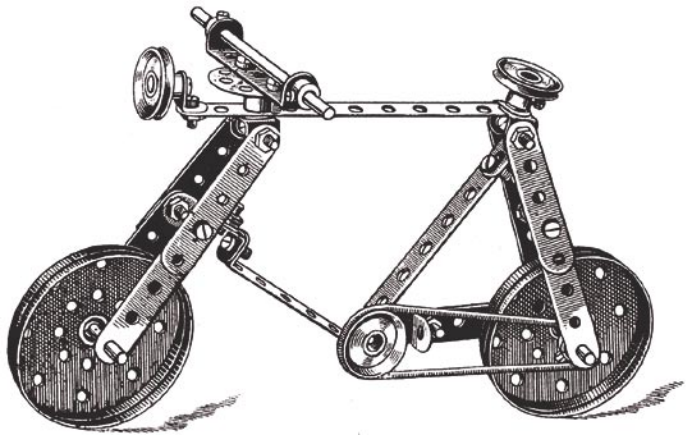
Firewalls on Linux are built using netfilter/iptables. Packet mangling is a feature of netfilter/iptables that I personally find very useful and can be used for many security issues. If you use packet mangling in conjunction with the l7-filter project (*http://l7-filter.sourceforge.net*) you can really optimize your network by restricting different types of traffic (e.g. P2P which can be very disruptive for your network).

Conclusion

Packet mangling is a very useful feature of netfilter/iptables. It can be used in many real life application, enabling users to make their network intelligent by perform data prioritization and bandwidth shaping for different types of traffic. ●

Basic of Firewalling and iptables

Antonio Merola & Arrigo Triulzi



Internet is a tremendous bin of information and this is a great advantage, but likewise is quite easy to get lost or waste lots of time to collect info about topic we are interested in. This is where magazines like this come to play; with this Starter Kit firewall issue we want give you everything concerning, packet filtering and firewalling starting from scratch, basically you have in your hands the fastest way to learn about basic of firewalling and iptables!

A firewall is a software box used to filter packets according to rules to match; these rules are intended to protect hosts behind this box, with this box we are defining a perimeter. For instance, at your home DSL connection, quite surely you have a box acts as a router/firewall between you and your service provider network, this because you want to be safe while connecting to an untrusted network as internet.

Of course a firewall is not a heal-all for security, because a single security technology will give a single point of failure, anyway a firewall is a must to secure networks. So let's start understanding key concepts that we will meet while firewalling, first of all with a packets review.

Basic packets review

Dealing with TCP/IP networks we deal quite often with packets. To better understand what a packet is made of, we start with the concept that depending on the type of network, packets may be referred by another name: frame, block, cell, segment, datagram etc., anyway regardless this, they are quite often composed of three part: *header*, *payload* and *trailer*.

The header contains instructions about the data carried by the packet, these instructions may include for instance the Protocol as in the case of an IP packet carrying on another packet (transport protocol) as TCP, UDP or ICMP packet (ICMP is embedded as a transport protocol as TCP/UDP, although it is a layer three protocol as the IP protocol itself).

The *payload*, also called the body or data of the packet is the actual data that the packet is delivering to the destination; the *trailer* typically contains some bits that tell the receiving device that it has reached the end of the packet. It may also have some type of error checking such as the Cyclic Redundancy Check (CRC). Packet filtering/firewalling is

What you will learn...

- you will learn about stateless and stateful filtering, along with iptables basics.

What you should know...

- the reader should have at lea basic knowledge about the TCP-IP protocol.

surely linked to some kind of connection with values defined inside packets header's fields, some kind of comparison that let decide what allow and what not. Let's start understanding the so-called *stateless filtering*.

Stateless filtering

First firewall generations were called stateless filter. Stateless filtering is the ability to filter packets taking decisions based on values discovered inside header's fields. In Figure 1 an IP packet header is displayed.

Now let's suppose we want our packet filter, allow only outgoing web traffic; we already know that this means only TCP packets directed to the TCP destination port number 80. Well, our stateless filter has to *look* inside IP packets and according to the field in byte position number 9 in the IP header it has only to allow TCP packet, in this case only IP packets with the binary value of *00000110* (decimal number 6, tcp protocol value identifier) which means only IP packets that carry on TCP packets, no UDP or ICMP. Now in the same way it has to *look* the second e third byte in the TCP header looking for binary value of *00000000 01010000* (decimal number 80, http port identifier) has showed in Figure 2.

In the same way we can filter only certain networks, only packets with flag SYN or ACK set, etc. or packets directed on certain ports and so on, basically all the fields of the packets header might be compared.

A limit of these kind of filtering is that it isn't able to keep the state of a connection and isn't able to *look* inside the payload in order to perform a kind of advanced filtering.

This limit plays a problem handling complex protocols that deals with dynamic port allocation as in the case of FTP or RPC protocols.

As a matter of fact to let an active ftp session go throughout the packet filter, would be necessary allow all traffic from source port 20 to all destination ports greater than 1023, while in the case of passive ftp mode would be necessary allow all traffic to ports in the 1-1023 range. As a matter of fact we would have lots of open ports to handle these kind of dynamic port allocation and we would be unable to recognize packets not belonging to legitimate sessions. Now a question: *is this kind of filtering useful?*, of course the answer is yes. We can perform a so called absolute filtering, to be more precise the filtering we want apply regardless of our security rules, for instance is good choice to drop inbound traffic with source ip belonging to internal networks or belonging to private addressing space or to IANA unallocated address space such as `0.x.x.x - 1.x.x.x - 2.x.x.x` etc. or more, drop critical ports as `TCP/UDP 135-139` and so on.

Almost every router implements the so called ACL (*Access Control List*) that let you specify this type of filtering; of course with a firewall is possible to perform this filtering and later on you are going to see how you can do it with iptables. It's now time to understand the other side of filtering called Stateful.

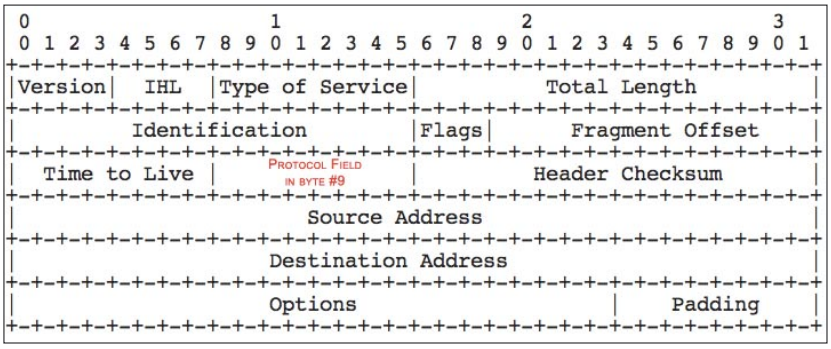


Figure 1. *IP protocol header, note that one tick mark represents one bit position*

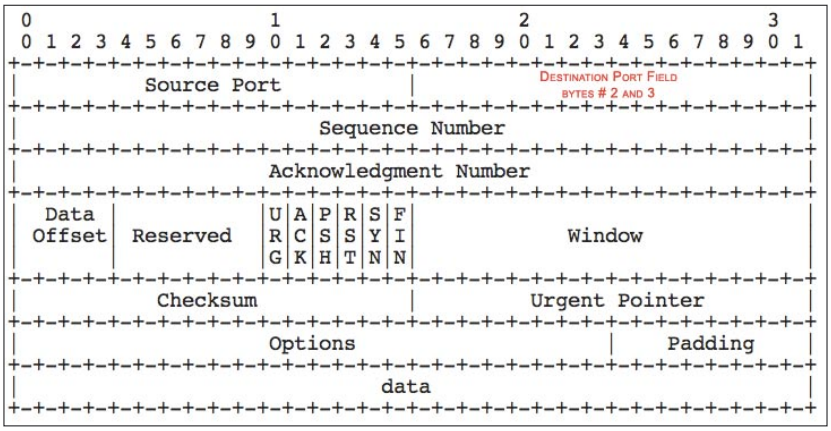


Figure 2. *TCP protocol header, note that one tick mark represents one bit position*

Stateful filtering

As an improved mechanism this kind of filtering is able to keep a session table in order to apply decision based on communications already logged in this table. As a rule of thumb, as soon as a connection is initiated it is compared with the filter rules; if there is an *allow/permit* for that connection, an entry is created in this table; this table is called quite often *connection table*. This because further packets related are matched with the entries into the connection table in order to verify if they are intended as a part of the connection or are uncorrelated packets to discard. Once the session has ended, its entry in the table is removed.

Basically the packet filter store the four values representing a TCP/IP session: source ip and port, destination ip and port; advanced or let's say modern stateful packets filters are able to save also others value as the sequence number of

the connection etc., appropriate behavior is to better know how stateful filter handle the connection table. As a note, only lately packets filter are able to handle stateful connection with also some kind of ICMP messages, where a reply message packet is allowed only as a response to a request message packet. Another restriction for this kind of filtering is linked to the inability of protocol or payload checking.

Complex protocol behavior as in the case of FTP or RPC etc. a bit unsuitable with stateless filtering, let's foresee we only want to allow http port 80 traffic, it's impossible with this filtering to check that actually http is going through the port 80. So in the first case the packet filter has to be able to look inside the payload to correctly allocate the port to open during the session, while in the second case it has to be able to recognize application-level data, the latter filtering is also well known as proxying or application firewall such as the relatively new XML firewall concept, a specialized box able to perform security with XML (Extensible Markup Language) messaging exchange. We are arriving with that kind of filtering called Deep Packet Inspection (DPI) capable to check also for non-protocol compliance along with predefined rule to decide if allow or deny packets, basically a layer 2-7 filtering inspection; nowadays included in network appliance this is a technology also used to perform a sort of basic intrusion prevention, as result of the predicted convergence of Firewall and Intrusion Prevention Systems.

Now that we now know what we are talking about, we can look up on internet the word *firewall* and check others opinions on the topic; I advise to use the following method every time you want to get a general overview on a topic. I think the best way to do this is invoking google and wikipedia; just type www.google.com and google in this way: *define:firewall*, (see frame Definition of Firewall), while from *www.wikipedia.org* you should get something like showed in frame *Firewall (Networking) – from Wikipedia, the Free Encyclopedia*

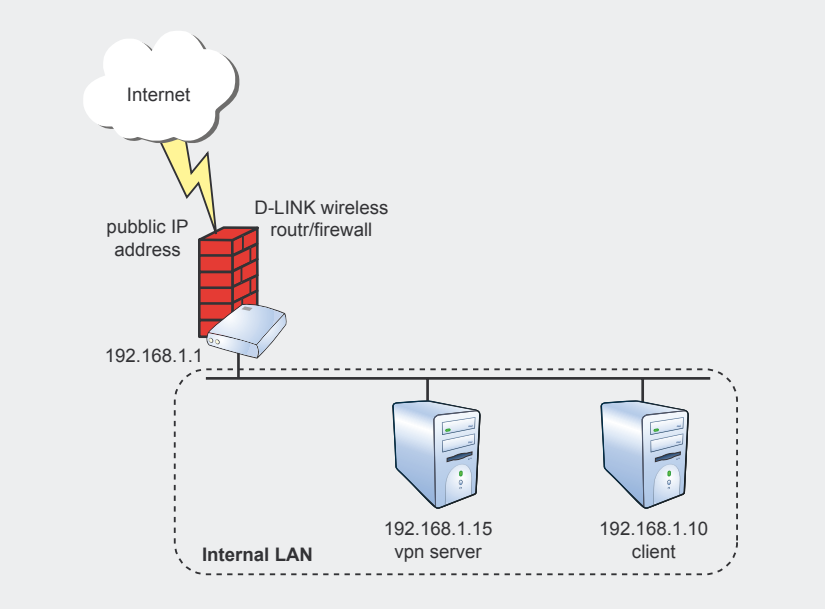


Figure 3. *Typical home DSL network*

```
# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere            state RELATED,ESTABLISHED
DROP       all  --  anywhere             anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
TCPMSS     tcp  --  anywhere             anywhere            tcp flags:SYN,RST/SYN TCPMSS set 1460
ACCEPT     all  --  anywhere             anywhere            state RELATED,ESTABLISHED
ACCEPT     tcp  --  anywhere             192.168.1.15        tcp dpt:443
DROP       all  --  anywhere             anywhere

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DROP       icmp --  anywhere             anywhere            icmp destination-unreachable
DROP       icmp --  anywhere             anywhere            state INVALID
```

Figure 4. *D-Link iptables output*

iptables

On most Linux systems, iptables is a software installed under */sbin/iptables* it's widely used and that is why is also present in many commercial firewall; lots of DSL box are custom Linux with iptables. For an easy approach to iptables and just to avoid writing the umpteenth iptables tutorial we preferred to analyse a typical home DSL configuration, after a proper introduction. On the web inset there is the link where, you can reach a valid tutorial about iptables for an in depth look written by Oskar Andreasson. Basically if the iptables host has one interface it acts as a personal firewall and has to scrutinize all the incoming packets directed to the firewall itself, that is called INPUT chain in iptables; if it has two or more interface, then incoming packets to get somewhere are matched against the FORWARD chain; finally when the firewall itself generates packets sent somewhere else the OUTPUT chain is matched. The detailed syntax of the iptables command is documented in its man page, which can be displayed by typing the command `man iptables`.

iptables packet filtering rules

The configuration we are going to understand in Figure 4 is related to a D-Link packet filtering default configuration, taken from command line but set-up via web interface flagging *Enable Firewall* box and adding some filtering rule. The network tested is drew in Figure 3, while in Figure 5 the NAT table is showed.

The D-Link box is a wireless router with firewall capabilities, a customized Linux host with iptables version

Firewall (Networking) from Wikipedia, the Free Encyclopedia

A firewall is an information technology (IT) security device which is configured to permit, deny or proxy data connections set and configured by the organization's security policy. Firewalls can either be hardware and/or software based. A firewall's basic task is to control traffic between computer networks with different zones of trust. Typical examples are the Internet which is a zone with no trust and an internal network which is (and should be) a zone with high trust. The ultimate goal is to provide controlled interfaces between zones of differing trust levels through the enforcement of a security policy and connectivity model based on the least privilege principle and separation of duties. A firewall is also called a Border Protection Device (BPD) in certain military contexts where a firewall separates networks by creating perimeter networks in a DMZ. In a BSD context they are also known as a packet filter. A firewall's function is analogous to firewalls in building construction. Proper configuration of firewalls demands skill from the firewall administrator. It requires considerable understanding of network protocols and of computer security. Small mistakes can render a firewall worthless as a security tool.

v1.2.6a. Here, we are not focused on the quality of the filtering realized while in the basics of a simple layout and iptables.

Let's analyse row-by-row the outputted configuration and it's meaning (Figure 3). The command `#iptables –L` is used to list the rules in a chain (--list is valid as well), if no chain is specified the operation is performed on all chains. For example, the command to list the rules in the OUTPUT chain is `#iptables -L OUTPUT`

If you have this type of router, in order to get a shell you have to telnet your D-LINK enter root as login and the administration password you use to log as admin in the web http gui.

As you can see there is a default policy ACCEPT which means if no rule is matched in the chain apply ACCEPT on that packet, in this case there are just two rows in the INPUT chain and the second one is quite simple: *drop* – discard everything regardless the protocol, the packet source or packet destination IP, while the first rule is an *accept* policy which means allow any packets from any source to any destination where the *state* of the connection is already in place (ESTABLISHED) and not a new one regardless of the protocol. Within iptables, packets can be related to tracked connections in four different states: NEW, ESTABLISHED, RELATED and INVALID.

The parameter RELATED is the way iptables handle complex protocols as FTP; the port allocation problem discussed before with stateful filtering where port allocation is negotiated within the actual payload of the protocol data. Surely this rule had been inserted with the following command `iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT`. We can read the INPUT chain in this way: every packet not related to connection initiated from the internal network has to be discarded. The FORWARD chain is just a little bit more populated than the others; as the previous chain there is a default policy ACCEPT, which means if no rule is matched in the chain apply ACCEPT on that packet. The second and the last rules are the same as in the INPUT chain with the same meaning.

The first rule is related to the tcp protocol and is to set the Maximum Segment Size on a packet, this only for SYN and RST/SYN packets the firewall sees. For detail about the MTU read RFC 1191 – Path MTU discovery. The command for this rule is `iptables -A FORWARD -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --set-mss 1460`, this allows setting the MSS value and usually it is the outgoing interface's MTU minus 40.

The advantage of this rule is linked to an icmp message type (fragmentation required but DF bit is set, DF stands for Don't Fragment) used by a router/firewall to inform a sender about the necessity of fragmentation in the original packets because of an MTU too big. The error message sent back as icmp message, contains the MTU of the network that requires fragmentation; the problems is that some Internet Services Providers and some servers as a filtering policy block this type of icmp mes-

```
# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
DNAT       tcp  --  anywhere             anywhere            tcp dpt:443 to:192.168.1.15:443

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE all  --  anywhere             anywhere

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Figure 5. *D-Link iptables nat output*

```
# cat firewall_start
/sbin/insmod ip_tables
/sbin/insmod ip_conntrack
/sbin/insmod iptable_filter
/sbin/insmod iptable_nat
/sbin/insmod ipt_MASQUERADE
/sbin/insmod ipt_state
/sbin/insmod ipt_TCPMSS
/sbin/insmod ipt_limit
/sbin/insmod ipt_REDIRECT
/sbin/insmod ipt_multiport
/sbin/insmod ip_conntrack_ftp
/sbin/insmod ip_nat_ftp
/sbin/insmod ip_conntrack_h323
/sbin/insmod ip_nat_h323
/sbin/insmod ip_conntrack_irc
/sbin/insmod ip_nat_irc
/sbin/insmod ip_conntrack_tftp
/sbin/insmod ip_nat_tftp
/sbin/insmod ip_conntrack_talk
/sbin/insmod ip_nat_talk

# cat firewall_stop
/sbin/iptables -F
/sbin/iptables -X
/sbin/iptables -t nat -F
/sbin/iptables -t nat -X
/sbin/rmmod ip_nat_talk
/sbin/rmmod ip_nat_tftp
/sbin/rmmod ip_nat_irc
/sbin/rmmod ip_nat_h323
/sbin/rmmod ip_nat_ftp
/sbin/rmmod ip_conntrack_talk
/sbin/rmmod ip_conntrack_tftp
/sbin/rmmod ip_conntrack_ftp
/sbin/rmmod ip_conntrack_h323
/sbin/rmmod ip_conntrack_irc
/sbin/rmmod ipt_multiport
/sbin/rmmod ipt_limit
/sbin/rmmod ipt_TCPMSS
/sbin/rmmod ipt_state
/sbin/rmmod ipt_MASQUERADE
/sbin/rmmod ipt_REDIRECT
/sbin/rmmod iptable_nat
/sbin/rmmod iptable_filter
/sbin/rmmod ip_conntrack
/sbin/rmmod ip_tables

# cat flush_firewall
/sbin/iptables -F
/sbin/iptables -X
/sbin/iptables -t nat -F
/sbin/iptables -t nat -X
```

Figure 6. *D-Link iptables firewall scripts*

sage, which can give you problems when large packets are exchanged, because you are not able to receive this message in order to set a correct MTU. We can read this rule in this way: set the Maximum Segment Size to 1460 to any TCP SYN and TCP RST/SYN packets across the firewall.

The third rule is to let any external host establish a TCP connection directed to the 443 port of the internal server with `IP 192.168.1.15`, an openvpn server listening for incoming connection on the ssl port; the command is:

```
iptables -A FORWARD -p tcp -d
192.168.1.15/24 --dport 443 -j ACCEPT.
```

The last chain is the OUTPUT where the default policy ACCEPT is applied, which means if no rule is matched in the chain apply ACCEPT on that packet; only two icmp drop are applied, the first one in order to avoid sending icmp destination unreachable messages

```
iptables -A OUTPUT -p icmp --icmp-type
destination-unreachable -j DROP
```

the second and last one to drop icmp packets with an invalid state or better if the ICMP error messages does not correspond to any known connections

```
iptables -A OUTPUT -p icmp -m state --
state INVALID -j DROP;
```

it is a good idea to discard everything in this state.

As you can see there isn't a drop as last rule and according to the default policy any other outgoing packet is permitted.

iptables NAT rules

We turn our attention now on what nat is and how iptables handle nat tables. NAT stands for Network Address Translation and its main goal is to be a solution to the problem of limited IPv4 (IPv4 stands for Internet Protocol version 4) address space, albeit IPv6 protocol also resolves

Definitions of Firewall

- Gateway that limits access between networks in accordance with local security policy. [NS4009] – *www.pki.vt.edu/help/glossary.html*;
- A firewall is a hardware or software solution to enforce security policies. In the physical security analogy, a firewall is equivalent to a door lock on a perimeter door or on a door to a room inside of the building – it permits only authorized users such as those with a key or access card to enter. A firewall has built-in filters that can disallow unauthorized or potentially dangerous material from entering the system. It also logs attempted intrusions, – *www.tecrime.com/0gloss.htm*;
- A firewall is either the program or the computer it runs on, usually an Internet gateway server, that protects the resources of one network from users from other networks. Typically, an enterprise with an intranet that allows its workers access to the wider Internet will want a firewall to prevent outsiders from accessing its own private data resources, – *www.course.com/careers/glossary/internet.cfm*;
- A dedicated gateway machine with special security precautions on it typically used to protect a network when it is connected to an outside network, especially the Internet – *www.assistireland.ie/glossary.asp*;
- is a type of proxy server with additional features. Firewalls are usually placed between the users of a LAN and the Internet (some ISPs also use firewalls). The firewall can be set to screen for incoming viruses and only allow access to certain resources on the Internet as a security measure. It can also cache previously visited sites to avoid excessive use of bandwidth, – *www.smallbizonline.co.uk/glossary_of_internet_terms.php*;
- and so on.

About the Authors

Antonio Merola works as senior security expert. He started his career 10 years ago; he worked as consultant serving several company as Microsoft Certified Systems Engineer. Since 2001 he has been involved in many aspects of security such as perimeter protection, vpn, intrusion detection etc. as employee for Telecom Italia.

Additional, as a freelancer, he serves several companies as consultant and instructor on a wide variety of security topics. Antonio, holds several certifications and is a frequent conference speaker; he published numerous papers on a broad range of security subjects. In his spare time he maintains his security portal *http://www.securityindepth.org*.

Arrigo Triulzi is a SANS certified instructor, trained in Pure Mathematics, holds an MSc in Mathematical Computation from Queen Mary, University of London, and is working towards a PhD in Algebraic Computation. He is co-founder and Chief Security Officer of K2 Defender Limited, a bespoke high-end IDS solutions provider. Arrigo is also a free-lance consultant in IT Security with particular expertise in secure network design, network security analysis, and incident handling.

He is also the administrator of the IDS Europe mailing list. Having worked with both popular and less common flavours of Unix he is comfortable working in any heterogeneous networking environment and his knowledge also includes esoteric operating systems such as Guardian/NSK.

Arrigo is co-inventor in an EU patent for a high-performance distributed IDS design, and has written on a variety of security topics. Recent work includes web research into IDS deployment on IPv6, firewall verification using IDS, and distributed concept virii.

On the Net

- *http://www.iptables.org/* – iptables/netfilter's home,
- *http://iptables-tutorial.frozentux.net* – iptables tutorial by Oskar Andreasson.

this problem. Basically with IPv4 we have been consuming all the IPs available in the address space and with nat is possible to use less ip address than necessary, masquerading many hosts to one ip address as general rule.

NAT allows hosts to share the same IP address, for example all hosts in our Internal LAN (Figure 4), belonging to the `192.168.1.0/24`, while connecting to the internet through their default gateway get translated by the firewall itself thanks to it's NAT capability. In this kind of connection the firewall replace the source ip addresses of packets; all `192.168.1.0/24` hosts will get translated their source IP with the firewall public IP address set on external interface as in Figure 4.

Without this, packets would be forwarded by the firewall with their internal ip and that would be useless because private ip are not routable on Internet.

As you can see there is the POSTROUTING chain used to alter all packets just as they are leaving the firewall, in this chain a MASQUERADING rule is applied: `iptables -t nat -A POSTROUTING -j MASQUERADE`. Masquerade is pretty much the same as SNAT (Source NAT) with the exception that masquerading will automatically set the new source IP to the default IP address of the outgoing network interface. There is also the so-called DNAT (Destination NAT) extremely useful when it comes to setting up servers as in the case of our openvpn server.

We may then this service on the same firewall public IP address from the outside via DNAT as in the case of the PREROUTING chain where it is applied. In this way from any location on Internet we can reach our openvpn server just doing a TCP connection to the firewall public IP address on port 443:

```
iptables -t nat -A PREROUTING -p tcp
--dport 443 -j DNAT --to-destination
192.168.1.15:443.
```

In the end we would like you give a look to the scripts in Figure 6 to better understand when you flag or unflag *Enable Firewall* on this kind of Linux based router/firewall, what actually is invoked.

You can see that enabling the firewall means (*firewall_start*) load `ip_tables` and load other modules (`/sbin/insmod module_name`) as in the case of the `ip_conntrack_ftp` involved in the stateful filtering of the ftp protocol and so on; flushing the firewall (*flush_firewall*) means flush all rules in all the iptables chains (`iptables -F`) and erase all chains (`iptables -X`), along with the nat table in the same way (`iptables -t nat -F` and `-X`). Disabling the firewall (*firewall_stop*) a flushing before is performed and then all the related loaded modules are removed (`/sbin/rmmod module_name`).

Conclusion

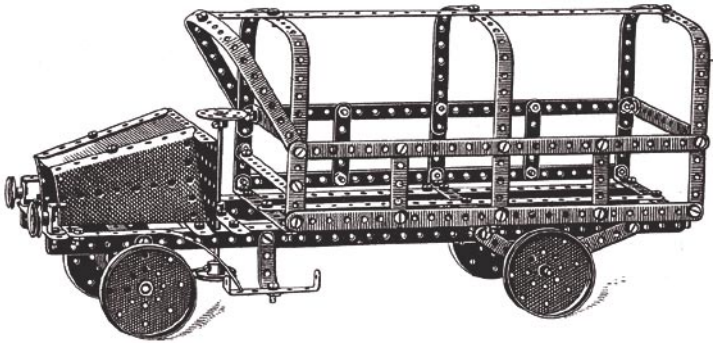
As we already written at beginning, firewalling is not only related to packet filtering; in fact, while reading other articles in this issue, you will learn with Massimo Fubini's article how to secure the Apache web server with mod-security, a web application firewall.

You will learn how iptables is powerful along with others software as in the case of Raul Siles's article; he wrote about a mechanism that allows creating new rules opening specific ports, let's say on-demand firewall rules.

With Carlos Fragoso's article you will understand that is better to opt for OpenBSD pf firewall in order to build a robust perimeter with high availability. Last but surely not least, Jess Garcia will explain what additional security technologies can be deployed at your firewall. ●

Much More Than Just a Firewall

Jess Garcia



Firewalls are continually evolving. Due to their role as network bottlenecks and enforcing points, they are the right place to implement controls and sometimes other types of functions that are not strictly related to traditional traffic filtering.

Additionally, firewalls have had to adapt themselves to the new technologies and threats that the 21st century has brought along.

If you take a look at the most popular commercial firewalls, you will see an extensive set of features which may or may not be related to the firewall's traditional role as a traffic filtering device: Virtual Private Network (VPN), Certificate Authorities, Content Inspection, Intrusion Prevention, Intrusion Detection, VLAN support, etc. These features have been included along the years to provide more flexibility to companies who need to deploy firewalls in effective, scalable and easy-to-manage ways.

It is not that common to see many of these features bundled in open source firewalls such as netfilter/iptables or pf, as these typically follow a more modular design; if you want extra functionality you will have to add it yourself. It is possible without too much effort to extend your firewall's capabilities to provide extra funcionality such as Intrusion Detection, Intrusion Prevention, Honeypotting or Virtualization.

In this article we will be covering some of those extra functionalities, and some special ways in which you can use your firewalls.

Intrusion Detection

Intrusion Detection is the art of detecting inappropriate, incorrect, or anomalous activity. In simple words and in the most common scenario this means that if someone hacks into one of your systems you will want to find out as soon as possible.

An evolution of this concept is Intrusion Prevention, which applies the same methodology to stopping attacks rather than (or in addition to) detecting them. Obviously, prevention is better than detection, but only if you can guarantee that a particular activity is malicious. That's a difficult task, and that's why Intrusion Preven-

What you will learn...

- which additional security technologies can be deployed at your firewall.

What you should know...

- you should have read, at least, Antonio Merola & Arrigo Triulzi's article on the basics of packet filtering in this issue.

tion Systems (IPS) typically fall to Intrusion Detection mode when they are not certain about a particular type of apparently malicious activity.

There are specialized devices, Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS), that are specifically designed to carry out Intrusion Detection and Prevention functions but, wouldn't it be nice if we could do that at our firewalls? This is possible, indeed, and actually the most popular commercial firewalls are already including basic to advanced IDS and IPS capabilities. There also are programs in the open source world that allow you to do it but, as you probably know, you typically have to work a bit to make them work. Let me help you in that process.

Let's start with IDS. There are many ways of doing Intrusion Detection. One is by using specific IDS software. You are lucky in this case, one of the best IDS software is open source: it's called *snort*[1] and you may have heard of it before because it's really popular and widely used. The type of intrusion detection that *snort* mainly does is called *signature-based*, because it relies on traffic patterns which are characteristic to a particular attack, and looks for them in the network traffic.

But using a specific IDS is not the only way to do it. Most firewall administrators don't even realize it, but firewalls are a great source of Intrusion Detection data. Why? Well, the firewalls are the devices that take care of enforcing the policy of the organization (the set of rules that establishes what is permitted and what is prohibited) at the network level. If properly configured, the firewall will not allow anything that is unnecessary and it will log all violations to the policy, providing that Intrusion Detection data. Now, when an intrusion occurs there often is a violation of the policy: the attacker starts scanning the internal network, or a compromised server tries to connect to a remote IRC channel, etc. Those activities will typically be detected and logged by appropriately configured firewalls. You will see those log messages and you will be able to detect the intrusion – that is, if you are watching!

What tools can I use to do this log monitoring and detection, you may ask. In the first place, there is a good bunch of tools[2] out there that can do firewall log monitoring, from the simplest ones (such as IPTables Log Analyzer or fwlogwatch for netfilter/iptables or hatchet for OpenBSD's pf), to the most sophisticated Security Information Management engines (such as OSSIM), which can correlate your firewall activity with other security, network and system logs.

PSAD

There still is another tool for Linux which is even more Intrusion Detection-oriented than the ones mentioned before: PSAD[3]. PSAD uses Linux netfilter/iptables log files for its operation, can do scan detection, signature detection (based on snort signatures), passive fingerprinting, alerting and a lot more things. Actually, if you combine PSAD with it's companion tool, fwsnort [4], you can even use the snort application level signatures thanks to netfilter/iptables string matching capabilities.

PSAD installation is quite simple; you will find ready-to-install PSAD packages for most Linux distributions. PSAD works by monitoring the log messages that iptables sends to syslog so, to start with, you'll have to configure syslog to log kernel messages to a named pipe, */var/lib/psad/psadfifo*, so the PSAD daemon can receive them:

```
echo -e 'kern.info\t|/var/lib/psad/
psadfifo' >> /etc/syslog.conf
```

```
And then restart syslog:
/etc/init.d/sysklogd restart
```

Now you are logging your netfilter/iptables messages to syslog but... what are you logging? You can see your current iptables configuration with: iptables -L -vn. Can you see any LOG lines? By default Linux does not log dropped or rejected packets if you set a default DENY or REJECT Policy (default policies are established with the iptables -P command). You will have to set a kind of convoluted configuration to be able to do it. See Listing 1.

Once we have configured firewall logging, it's time to configure PSAD. PSAD configuration is not complicated either: even when there are lots of options you can tailor, you'll typically just have to edit the */etc/psad/psad.conf* file to specify your email address, the hostname, the HOME network, and little more. Then, start the daemon:

```
/etc/init.d/psad start
```

and just wait for alerts, either via email or through PSAD log files, stored in */var/log/psad*. You can use log watching software such as swatch[5] to monitor those log files if you want to perform sophisticated actions after an alert is issued. You have a lightweight IDS running now!

There is something important to take into account in this scenario, which you will see very soon if you deploy this kind of tool in your external firewalls: in today's hostile Internet environment, you'll receive thousands of attacks from the outside, which will very soon lead you to disable or ignore your PSAD alerts. The recommended procedure at your external firewalls is to look at traffic going out of your network, a strategy known as *Extrusion Detection*[6]. In today's environment where client-side attacks and *botnets* (armies of compromised hosts that hackers use in order to perform Denial of Service attacks, send SPAM, etc.) are the rule, it will be more effective to look at what is going out (that is,

Note

Most of the capabilities discussed in this article will be covered using iptables as an example, just because Linux use is more widespread than OpenBSD or FreeBSD in which pf works. However, pf is usually easier to configure and more appropriate for the corporate environment.

what is being blocked to the outside, if your firewall is properly configured) rather than what is coming in. At your internal firewalls – a much quieter universe – you will probably be able to supervise both directions.

Intrusion Prevention

Let's talk about Intrusion Prevention now. The first thing we must mention is that Intrusion Prevention is not a replacement for Intrusion Detection, but rather a complement. No security technology is perfect, and Intrusion Prevention is no exception; there is a number of ways of fooling it. However, the combination of different host-based and network-based Intrusion Detection and Prevention techniques will provide the best results (this multi-layered defense approach, called *Defense in Depth*, has proven to be the best way to protect today's system and network infrastructures). Additionally, do not forget the old saying: *Prevention is Ideal, but Detection is a Must*. Sooner or later you will get hacked (that's a fact), but what you definitely cannot afford is not knowing it.

The second thing we can mention is that it makes a lot of sense that IPS and Firewalls are combined in the same box, because they share a lot of common characteristics: they sit in-line with the traffic, dropping it or letting it through according to a set of predefined rules. In the firewall case, these rules have to do with ports, protocols and connection states. In the IPS case, these rules have to do with packet characteristics and contents and traffic profiles.

snort_inline

Now, what tool can we use for Intrusion Prevention in the open source world? The answer is *snort_inline*, an extension to snort that interacts with *netfilter/iptables* to drop packets that are identified as malicious by the snort engine.

If you are interested in playing with a pre-configured version of netfilter/iptables, snort and *snort_inline* all working together, the best thing you can do is download the Hoyewall CD[7], which is a bootable CD that is used in Honeynet deployments. (For more information about Honeynets visit the Honeynet Project website[8].) If you want to use *snort_inline* for production purposes, just download it from the *Snort* website, and install it with:

```
./configure --enable-inline ;
make ; make install
```

snort_inline configuration options are stored in the *snort_inline.conf* file, a file that defines its overall behaviour. Under the rules directory you will find a good number of rules that identify malicious traffic patterns. Let's examine one of them:

```
drop tcp $HOME_NET any ->
$EXTERNAL_NET 6666:7000
(msg:"CHAT IRC channel join";
rev:2; sid:1729;
classtype:misc-activity; content:
"|4A 4F 49 4E 20 3A 20 23|"; nocase;
flow:established,to_server;)
```

This rule will drop any packet that belongs to an attempt to join a chat channel from our organization to the outside. It makes a lot of sense to apply such a rule these days in our organizations. Why? Simply because botnets make use of this mechanism to control compromised hosts. It may not be possible to apply this rule universally to your whole organization if users are allowed to use chat, but it will definitely make a lot of sense to apply it to your servers networks, for instance, because your servers should never be using chat.

At this point you may be wondering how is *snort_inline* capable of seeing the payload of the packets. Well, the mechanism is slightly different than the PSAD case we introduced before, although in this case we will also use the capabilities of the powerful netfilter/iptables engine.

Netfilter/iptables has the possibility to define several targets for the traffic, such as ACCEPT, DROP, QUEUE, or RETURN. The QUEUE target (which relies on a kernel module named *ip_queue*) is a mechanism for passing packets out of the stack for queueing to userspace where, in this case, *snort_inline* will receive them and process them. It will be possible then to receive these packets back into the kernel along with a verdict by *snort_inline* specifying what to do with the packets (such as ACCEPT or DROP). This verdict will be reached by inspecting the packets handled by netfilter/iptables and applying the ruleset to each one of them.

A cool feature of *snort_inline* is that it actually allows you to modify the packets prior to reinjection back into the kernel. A rule that would perform such an action would be:

```
alert tcp any any <> any 80
(msg: "tcp replace"; content:
"GET"; replace:"BET";)
```

This would replace the GET keyword with BET, effectively disabling a supposedly malicious HTTP download. This is useful for disabling an attack without breaking a connection, because if you drop a packet out of a tcp session, the session will not be allowed to go on, which may alert

Listing 1. *A set of configurations*

```
   iptables -N LOG_DROP   # Creates a new chain called LOG_DROP
   iptables -A LOG_DROP -j LOG
# Packet will be logged first in the LOG_DROP chain
   iptables -A LOG_DROP -j DROP
# Packet will then be dropped in the LOG_DROP chain
   [...]         # Your INPUT / FORWARD ruleset goes here
   iptables -A INPUT   -j LOG_DROP
# At the end of the INPUT and FORWARD rulesets the
   iptables -A FORWARD -j LOG_DROP
# packet is LOG_DROPped (if not previously allowed)
```

the attacker of the presence of the IPS. However, as you can imagine, this method is not foolproof and will only work for the most inexperienced attackers.

One important thing to mention is something called False Positives. A False Positive is a false alert. In the Intrusion Detection world False Positives are a problem because they confuse (and often overload) the analyst. In the Intrusion Prevention world False Positives are much more dangerous because a False Positive means that a legitimate connection is being interrupted. Imagine if that particular connection is a critical one! That's why IPS rulesets must be carefully configured, and that's why the main value of a particular IPS is the accuracy of its signatures. The truth is that the lead in this case belongs to IPS vendors, which provide a rule update and maintenance service to their customers. Unfortunately, the open source community is not currently maintaining an efficient IPS signature (except maybe the rulesets provided by the Honeynet project, but they are not updated periodically, just with the release of a new Honeywall CD version). The snort rules are freely available, but they are IDS-oriented; if turned straight into IPS signatures a great risk of Denial of Service arises.

Honeypots

At this point you have seen how to use your firewall to detect intrusions and even to prevent them. There still is a million things you can do with your firewall to increase the security of your organization. One of them is Honeypotting.

According to the consensus definition reached by the members of the Honeypots mailing list at Securityfocus[9], *a Honeypot is a resource whose value is in being attacked or compromised*. This broad definition makes room for a big number of Honeypot types[10], strategies and approaches, from very simple to very complex, which do not necessarily have to be technological. Some of them can be applied at your firewall.

One of the many uses for Honeypots is to confuse attackers. If they are trying to attack your organization and, all of a sudden, they see a lot of possible victims, open ports, protocols, services, systems, networks, etc., it will be much more difficult for them to identify what the valuable resources are (if they don't have insider information, that is), and this will typically force them to be more 'noisy,' increasing the probability of being discovered.

So, how can you do this at your firewall? This is kind of simple: just use the netfilter/iptables REDIRECT capabilities. There actually was a program developed by George Bakos called The Tiny Honeypot[11] (THP) which made use of this strategy, among others. THP was a set of scripts devoted to Malware collection. The idea was quite simple: all the ports in a particular Linux system would be redirected to a single port where a listener (netcat-type in the simplest case, service-aware in the most sophisticated) would respond to connection attempts and would log the data exchanged (often containing malware). The netfilter/iptables entries that made this possible were:

```
iptables -t nat -N thp-redir
iptables -t nat -A PREROUTING
   -i $EXTIF -p tcp --tcp-flags
FIN,SYN,RST,ACK SYN
   -m limit --limit 60/minute
   -j thp-redir
iptables -t nat -A thp-redir
   -p tcp -j REDIRECT
   --to-ports 31337
```

We will not further elaborate on the workings of THP, because it was superseded by a much powerful software called *nepenthes*[12].

There is another very interesting approach in which Honeypots can be used at your firewall, and that is Tarpit mode. A Tarpit, sometimes called *Sticky Honeypot*, is a type of defensive Honeypot whose mission is to slow down network connections, like the ones typically observed when worm infections plague the organization. As you should know, TCP connections are established in 3 phases: SYN, SYN-ACK, ACK (this is known as the TCP 3-way handshake). The trick the Tarpit uses when a host tries to connect to an unserved port is to answer to the malicious connection requests and then become silent. The malicious host will try to verify for a given amount of time (typically 10-20 minutes) if the connection is active or not before it times out. The benefits are that during that particular amount of time the *energy* spent by the malicious or infected host trying to infect the tarpit will allow him to compromise other possibly vulnerable victims in the network (the malicious host only has a certain amount of *stamina*, that is, a limited amount of connections he can keep open).

There are several programs that can do Tarpitting, the first and most popular of them is LaBrea[11], which was born to fight the Code Red worm, and which in its version 2 implemented an even more sophisticated mechanism than the one described above.

Netfilter/iptables introduced a new target called TARPIT which mimics LaBrea v1 behavior. Imagine you are facing an infection by a worm which hits port number 31337/tcp. What you can do is simply create the following entry in your firewall:

```
iptables -A INPUT -p tcp -m
tcp --dport 31337 -j TARPIT
```

From that moment on, whenever an infected host hits port 31337 at the firewall, it will be tarpitted. Isn't it simple? If you want to magnify the impact, you can forward at the router level the unused IP addresses of your IP address space to a Linux system not acting as a router. Then just enable IP forwarding, and add:

```
iptables -A FORWARD -p tcp -j TARPIT
iptables -A FORWARD -j DROP
```

Please note that the tarpit module may not be enabled or even configured by default in your kernel.

Transparent Firewalls

There is a very interesting and useful way in which you can use your firewalls, and it is in *Transparent Mode*, that is, at the layer-2 level, without routing.

Moreover, not all firewalls (especially the commercial ones) support transparent firewalling, but fortunately the two most popular open source ones, netfilter/iptables and OpenBSD pf do.

To be fair, the capability of a firewall to be transparent does not depend only of the firewall code; the operating system must be able to operate in *bridged mode*. As stated above, both Linux and OpenBSD, among others, can do it.

How do you do that, for instance, in Linux? It's easy. To create a bridge, you obviously need at least two interfaces, e.g. eth0 and eth1 (those interfaces will not typically have an IP address, although they may have it). So the only thing you have to do is bring up those interfaces, create a bridge (e.g. br0), bind the two interfaces to the bridge, and bring up the bridge:

```
ifconfig eth0 0.0.0.0 up -arp
# Bring up eth0 (with no arp response)
ifconfig eth1 0.0.0.0 up -arp
# Bring up eth1 (with no arp response)
brctl addbr br0
# Create the br0 bridge
brctl addif br0 eth0   #
Bind eth0 to the br0 bridge
brctl addif br0 eth1    #
Bind eth1 to the br0 bridge
ifconfig br0 0.0.0.0 up -arp
# Bring up the br0 bridge
```

Now you can bind standard firewall rules to the appropriate interface. For instance, to allow DNS traffic to your DNS server:

```
iptables -A FORWARD -p tcp
-m physdev --physdev-in eth0
-d dnsserver \
   --dport 53 -j ACCEPT
iptables -A FORWARD -p udp
 -m physdev --physdev-in eth0
-d dnsserver \
   --dport 53 -j ACCEPT
```

Virtual Firewalls

As you probably have noticed, our computer world is going *virtual*. It's already a long time since our systems were so powerful that they were most of the time idle, and that our networks have so much bandwidth that they are almost unused most of the time (if Peer-to-Peer is forbidden in your organization, that is).

The answer to that situation came by the hand of *virtualization*. In the systems world we've seen technologies such as Vmware, Virtual PC, Xen or Parallels allow users to utilize several operating systems at the same time. In the network world, it's a long time since VLANs emerged as a useful technology to create isolated network segments (even when VLANs are not considered by the security community as a secure segregation mechanism). VLANs (Virtual Local Area Networks) are a way of dividing a physical switch in isolated environments, similar to having several isolated switches. To make it easy to understand, it would be like grouping all the existing sockets in the switch in groups at your will, so a host inside a group will not be able to communicate with another group unless it traverses an external router (well, sometimes the switch can also act as a router itself, but let's not get into that complication at this point). There is a concept called 'trunking' which allows you to merge different VLANs into a single port. This is very useful if you want to expand a VLAN across different switches, or if you want to pass all the traffic to a device that can do VLAN-based routing or filtering, such as a virtual firewall, as we explain below. But, how do you identify which packet belongs to which VLAN if all the VLANs are traveling in the same *wire* you may ask? The 802.1q specification (the one that defines how Virtual LANs must behave) tells us that in order to identify to which VLAN a particular frame belongs (we actually call packets *frames* when we are talking at the network layer) an identifier (tag) will be added to it: that's the VLAN number.

Big commercial companies, especially those positioned in the network/security field, are putting a lot of effort and interest in integrating VLANs with the network infrastructure (in an effort to create what they call the *Secure Network Fabric*). All this means is that obviously VLAN support is now a requisite for present and future firewalls. Fortunately, open source firewalls meet once again the expectations as they have been supporting VLAN-based filtering for a long time already.

On the Net

- [1] *http://www.snort.org* – Snort,
- [2] *http://www.jessland.net/JISK/IDS_IPS/Log_Analysis/Tools.php* – Log Analysis, SIM & SEM Tools,
- [3] *http://www.cipherdyne.org/psad/* – PSAD,
- [4] *http://www.cipherdyne.org/fwsnort/* – fwsnort,
- [5] *http://swatch.sourceforge.net/* – swatch,
- [6] *http://www.awprofessional.com/title/0321349962* – Extrusion Detection: Security Monitoring for Internal Intrusions – Richard Bejtlich – Ed. Addison-Wesley; ISBN 0321349962,
- [7] *http://www.honeynet.org/tools/cdrom/*: Honeywall CD-ROM,
- [8] *http://project.honeynet.org* – The Honeynet Project,
- [9] *http://www.securityfocus.com/archive/119* – Honeypots mailing List at Securityfocus,
- [10] *http://www.jessland.net/Honeypots/Types.php* – Honeypot Types,
- [11] *http://www.alpinista.org/thp/* – The Tiny Honeypot,
- [12] *http://nepenthes.mwcollect.org/* – Nephentes,
- [13] *http://www.hack-busters.net/LaBrea/* – LaBrea,
- [14] *http://www.jessland.net/JISK/Architecture.php* – Security Architecture.

As you can imagine, in order to operate on VLANs, VLAN interfaces (interfaces that are capable of understanding 802.1q traffic) must exist. In Linux this is easy, and is created with the following command:

```
vconfig add eth0 2
```

This would create a virtual interface eth0.2 which would be connected to VLAN 2, provided that the eth0 network card is connected to a *trunk* port, that the Linux host has the 8021q kernel module loaded (*modprobe 8021q*), and that the vlan utilities are installed in the system. Now, creating a rule that would operate on the new interface is very intuitive.

```
iptables -i eth0.2 -A INPUT -j DROP
```

This would drop all traffic on the VLAN 2 interface eth0.2 (default deny rule).

Still much more

Up until this point we have covered some of the most common *extra* firewall features, but there still are many more.

If you take a look at a powerful firewall such as pf you will see that there still is much more:

- Honeypotting features with daemons such as spamd that allows you to fight spammers,
- Pf's Passive Fingerprinting capabilities, a technology that permits the identification of the remote operating system by just inspecting the packet headers, allow you to apply rules according to the remote operating system trying to connect to your firewall,
- It is possible to use QoS (Quality of Service) properties to *reserve* some bandwidth for your critical applications,
- High availability, which in pf's case is based on an open protocol called CARP, allow you to ensure that your infrastructure will be resilient to single points of failure at the firewall level,
- Packet normalization (scrubbing) provides pf the capability of protecting old systems with weak TCP stacks by modifying the network characteristics of the packets on their way out (things like the TCP sequence number),
- And last but not least authpf, a powerful mechanism to perform filtering in a user basis.

Different commercial tools can implement many of the different features we've been covering during this article, and some more.

A field that commercial firewalls are specially targeting for some years now is content inspection. As client-based attacks become more prevalent, the need for application layer inspection (e.g. web traffic, email traffic, etc.) becomes a bigger need. On the reverse side, sometimes organizations worry about where their employees are surfing (e.g. pornography sites) and want to stop them from visiting different types of websites. In both cases the firewall will need to understand the application it is filtering in order to stop or allow traffic based on some more sophisticated criteria. This is doable mainly for malicious traffic, sometimes even when relatively sophisticated malware is in place (remember the IPS capabilities we described before?), but if the inspection becomes too intense it will slow down the traffic, and that's something you don't want to happen. That's why firewall vendors have developed protocols that allow their products to integrate with external devices specifically designed to do that kind of job (a well-known example is CheckPoint's OPSEC).

In summary, as you can see there are tons of things that you can do with your firewall (and I'm sure I still have forgotten to mention some!).

Conclusion

I hope that at this point you have a clear idea of some of the extra things you can do with your firewall. Firewalls today are much more than filtering devices, and firewall software such as netfilter/iptables or pf provides a strong base in which a number of additional security technologies rely.

It's quite certain that within a few years from now we will see highly sophisticated firewalls with advanced IPS capabilities and many more things, and that will definitely improve the security of our organization and the manageability of the security infrastructure. However, please do not forget that you should never rely on a single device or technology to provide all the security to your organization. Defense-in-Depth is the right way to go, turning the firewall into a critical part, but just a part, of a rock-solid Security Architecture[14]. ●

About the Author

Jess Garcia, founder and CEO of JSS – Jessland Security Services, is a senior security consultant for government, commercial, financial and telecommunication organizations in Europe, Latin-America, Canada and the USA, and a certified instructor for the SANS Institute in several security technologies.

Jess' specializations include Computer Forensics, Intrusion Detection and Prevention, Security Architecture, and Honeypots, among others. Jess served for 10 years as a system, network and security engineer for the Spanish National Aerospace Institute (INTA), where he collaborated as a security expert with other space agencies (ESA, NASA) and international government, research, educational and commercial organizations.

Jess holds a M.Sc. in Telecommunications Engineering from the Univ. Politecnica de Madrid, and he is a frequent speaker at international events. Jess has authored the book Securing Solaris 8 & 9 Using the Center for Internet Security Benchmark, has contributed to some others, including diverse SANS Courseware, and is the author of a number of security standards for the Spanish administration.

CLUB PRO

hakin9

hakin9 is the only IT security magazine in the world which manage to merge the highest quality level with huge popularity. We want to make your life easier and invite you to join our special, new project:

One year hakin9 PRO subscription conjoined with a membership in our PRO SUBSCRIBERS CLUB! With the PRO SUBSCRIBERS CLUB you are entitled to:

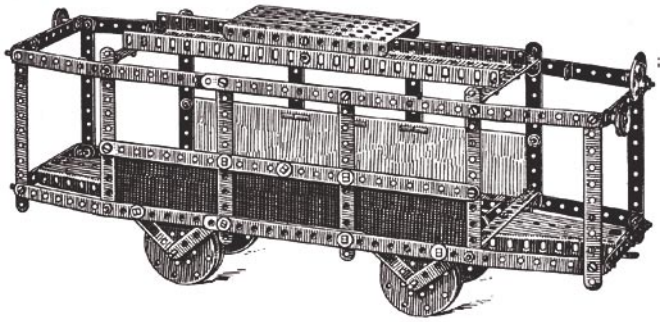
- **publish text announcement (max. 300 characters with spaces) in English version of hakin9 magazine**
- **having 20% discount on advertisement in the magazine**
- **space in the companies catalogue focused on PRO SUBSCRIBERS CLUB on hakin9 website**

DON'T MISS THE CHANCE! JOIN US TODAY!
PRO is the most profitable option for your company and costs ONLY $99!
Want to know more?

Just e-mail us at: en@hakin9.org

Web Application Firewall – ModSecurity for Apache

Massimo Fubini



Layer 3-4 firewalls are largely used as perimeter protection in corporate networks, but they can't address application level security. Luckily there are ways to detect and in case block, known or unknown web security attacks. With this article we are going to understand a so-called Web Application Firewall.

Recently we had a tremendous growth of the HTTP/HTTPS protocol usage, for every kind of data communications. In past years any application used to be managed by a custom protocol, while nowadays is quite necessary to develop and deploy applications with HTTP capabilities. Some examples are SSL VPN (instead of IPSec), Skype (able to work with HTTP protocol instead of H323 Protocol Stack), HTTP Chat (instead of *unix talk*, IRC), SOAP (instead of Corba, RPC), RPC over HTTP for Mail (instead of POP/IMAP & SMTP) and so on. The reason to let every single application work with HTTP protocol is not linked to its speed, reliability or efficiency (it is normally less optimal than other specifically developed protocols). One of the main reason of this tendency resides in its simplicity along with it being a *firewall friendly* protocol. HTTP applications are *firewall friendly* because often perimeter protection works just at OSI Layer 3 (IP) or 4 (TCP/UDP) and can't handle deep analysis in order to block at OSI Layer 7 (Application).

The fact that the HTTP protocol can often cross perimeter protection without security filtering is a great feature for users and application developers. They doesn't need to worry about specific characteristic of the protocol if it is open or blocked on the perimeter. Web Services are one of the greatest security holes in a network because of:

- The presence of a large number of networks with the HTTP protocol allowed,
- the convergence of every computer communications with HTTP protocol,
- the growing number of possibilities to develop and deploy web services; the continuous growth,
- growth of vulnerability founded in web application,
- the diffused ignorance on how to configure a web server and how to develop an application with security in mind.

In this article we discuss about *mod_security*, an Apache modules that can permit applying

What you will learn...

- Most common vulnerability in web application,
- When to use an HTTP filtering reverse proxy,
- Why and how to configure Apache with mod_security.

What you should know...

- The reader should know the http protocol and get knowledge of the apache web server configuration along with Unix/FreeBSD daemon configuration.

http filtering for some common security issues or to permit only traffic that is permitted by the security policy.

mod_security is often used for HTTP Server protection and not to manage content filtering on client activity. Content Filering on client activity can be better managed with other applications (for example Squid with the use of filtering modules).

Typical architecture

mod_security can be deployed on any apache web server. For example, it can be used on a stand-alone apache web server, in order to filter requests for that server. A typical use of *mod_security* is on an HTTP/HTTPS reverse proxy (See Frame Reverse Proxy). In *Figure 1* a typical architecture of a filtering http reverse proxy is shown. In a reverse proxy configuration, every time a user requests a page for i.e. *www.mysite.com,* it has to be processed and analyzed by the reverse proxy. The reverse proxy can accept requests and pass them to the right web server, block them or modify them. This kind of architecture can be used to filter request before the user reach the real web server, but can be used for many other purposes such as centralizing authentication, load balancing and high availability.

If the content published has to be SSL encrypted, it is usually done on the reverse proxy. The reason of this choice is because that is the only way for the reverse proxy to analys https data. Traffic going from the reverse proxy to the web server is usually in clear text both for performance reasons and the possibility to eventually insert other network security equipment (such as a network IDS) to analyse the traffic.

Installation

Installation in this article is based on FreeBSD 6.2 release and it is done using the FreeBSD ports package. It could be done directly from source code on any UN*X, however with different commands. It is possible to download a precompiled version of *mod_security* for lots of Operating Systems, also for MS Windows albeit I never tested it. I used FreeBSD as operating system both because I personally find it a very neat OS and because FreeBSD has a nice package for apache & *mod_security.* Basically during the installation, the following step are to be executed:

- download and compile apache, with modules for proxying,
- download and compile *mod_security,*
- download and compile *mod_proxy_html* (not mandatory).

For the purpose of this article, I decided to use Apache 2.2.3 with *mod_security* 1.9.4. This version of mod security can also be used with Apache versions 1.x and 2.0. The newer version of *mod_security* (2.0.x) can only be used with Apache 2.0.x or better.

To make a basic installation of apache and *mod_security*, using the FreeBSD ports collection, you can start by synchronizing the ports collection with the following command (you have to install cvsup on the system using the standard FreeBSD mechanism):

```
#cvsup -g -L2 -h cvsup.ch.freebsd.org
/usr/share/example/cvsup/ports-supfile
```

after the ports are fully synchronized, you can install Apache 2.2 (latest version) by typing:

```
#cd /usr/ports/www/apache22
#make WITH_SSL_MODULES=
yes WITH_PROXY_MODULES=
yes WITH_AUTH_MODULES=yes
#make install; make clean
```

This command compiles and installs apache, and all the needed libraries, with a set of modules for SSL encryption, proxying funcionality and authentication. After compiling apache22, you should compile and install *mod_security.* You can install latest version of *mod_security* 1.x by typing:

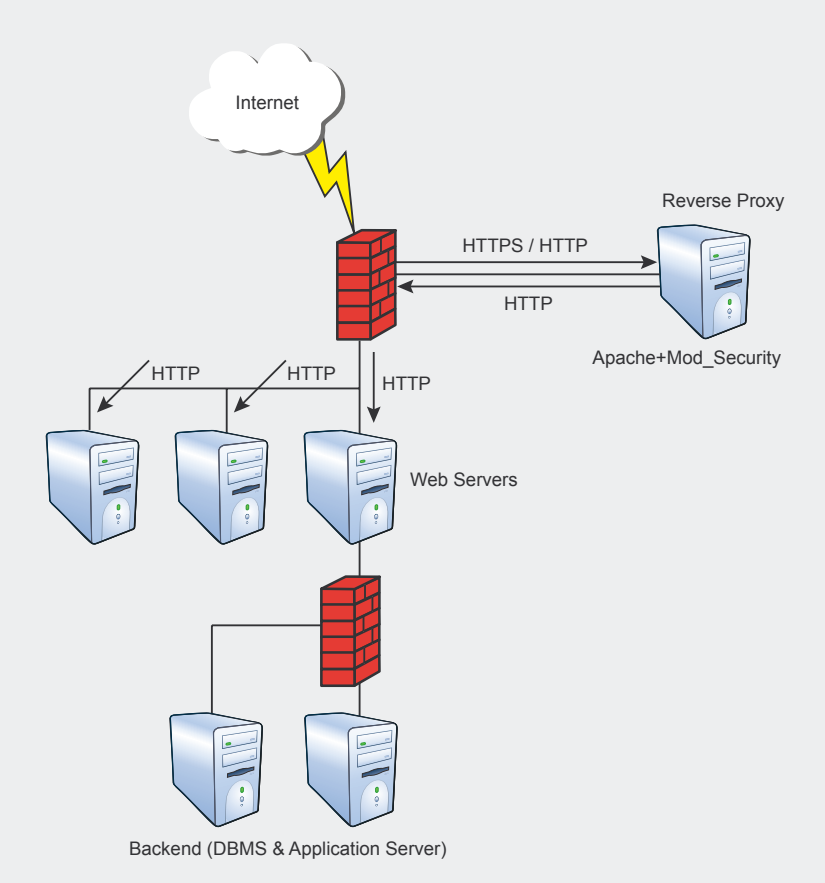


Figure 1. *Typical apache – mod_security deployment*

Listing 1. *Suggested modules lines are:*

```
LoadModule unique_id_module  libexec/apache22/mod_unique_id.so
LoadModule rewrite_module    libexec/apache22/mod_rewrite.so
LoadModule security_module    libexec/apache22/mod_security.so
LoadModule proxy_module    libexec/apache22/mod_proxy.so
LoadModule proxy_http_module    libexec/apache22/mod_proxy_http.so
    an optional module is required if you need url fixing:
LoadModule proxy_html_module  libexec/apache22/mod_proxy_html.so
```

Listing 2. *A typical virtual reverse proxy host in apache looks like this (hakin9.conf file):*

```
Listen 192.168.1.1:80
<VirtualHost 192.168.1.1:80>
    ServerAdmin m.fubini@ieee.org
    DocumentRoot /usr/local/www/data
    ProxyPass / http://realserver.local/
    ProxyPassReverse / http://realserver.local/
# Yes, we want to use mod_security
SecFilterEngine On
#Mod Security Configuration:
[…]
</VirtualHost>
```

```
#cd /usr/ports/www/mod_security
#make install ; make clean
```

You can optionally install *mod_proxy_html* for a site that requires html rewriting in a reverse proxy configuration. A typical case where *mod_proxy_html* is needed is when a site uses absolute urls, and rewriting of the web content is required. To install *mod_proxy_html* use:

```
#cd /usr/ports/www/mod_proxy_html
#make install; make clean
```

to enable Apache 2.2.x enter the following command (it will enable apache to start automatically at boot-time):

```
#echo 'apache22_enable=yes' >> /etc/
rc.conf
```

To manually start apache use the command:

```
#/usr/local/etc/rc.d/apache22 start
```

To stop apache use the command:

```
#/usr/local/etc/rc.d/apache22 stop
```

You can find some suggestion in (See Installing from Source frame) on how to compile *mod_security* in case your OS does not have a precompiled package or you prefer installing from the source code.

Configuration

After the binary files installation, you will deal with the most important part: configuring apache for reverse proxying and for security checking of http requests and responses. The entire configuration is done editing the file *httpd.conf*, for a FreeBSD installation located at */usr/local/etc/apache22/*. First of all you need to check if the needed modules are present in *httpd.conf* file.

Presence of other modules loaded, strongly depends on what kind of configuration you need to do on the reverse proxy.

It is important to highlight that the best thing to do is to disable all unused modules to prevent potential bug exploitation, this in order to have a bit more secure environment. It is also advised to harden the installation, by making apache run with as a non-privileged user (usually www is used by the default with FreeBSD). It could also be useful to further isolate apache with technologies like chroot (available on any *nix platform), FreeBSD Jail, Solaris Zone or Security Modules like SELinux or LIDS to prevent possible exploitations of the web server, compromising the overall host. You can look at (see frame Hardening the Installation with FreeBSD Jail) for some suggestion on using FreeBSD Jail to isolate the apache installation from the rest of the Operating System.

After configuring modules loaded by the web server, you need to start configuring Mod_Proxy on how serve pages. At the end of the httpd.conf in the default apache installation you will find the following directive:

```
Include etc/apache22/Includes/*.conf
```

This directive means there will be an inclusion in the httpd.conf file of all the file in `/usr/local/etc/apache22/Includes/*.conf`. I use this kind of configuration to create a file for every VirtualHost. If you have many virtual hosts it is easier to maintain. For this configuration I created the file:

```
/usr/local/etc/apache22/Includes/
hakin9.conf
```

The ProxyPass and ProxyPassReverse statements let the web server know what to do when it receives a connection on ip address `192.168.1.1` port 80. It will redirect all the http requests from the root web site / to the web server *http://realserver.local*. After I configured the web server I activated *mod_se-*

Web Application Firewall Definition

Source: *Web Application Security Consortium* web site [6].

Web application firewalls (WAF) are a new breed of information security technology designed to protect web sites from attack. WAF solutions are capable of preventing attacks that network firewalls and intrusion detection systems can't, and they do not require modification of application source code. As today's web application attacks expand and their relative level of sophistication increases.

Listing 3. *Some illegitimate activity examples of urls with cmd.exe command execution are:*

```
/certsrv/..%255cwinnt/system32/cmd.exe?/c+dir
/cgi-bin/..%255c..%255c..%255cwinnt/system32/cmd.exe?/c+dir
/iisadmpwd/..%255c..%255cwinnt/system32/cmd.exe?/c+dir
/msadc/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+dir+c:%5c
/pbserver/..%c0%af../..%c0%af../winnt/system32/cmd.exe?/c+dir
/rpc/..%c0%af../..%c0%af../winnt/system32/cmd.exe?/c+dir
/scripts/..%c1%1c../winnt/system32/cmd.exe?/c+dir
/scripts/tradecli.dll?template=nonexistfile?template=..\..\..\..\..\winnt\
                      system32\cmd.exe?/c+dir
```

Listing 4. *Some example of urls used to exploit a vulnerability and grab the password file:*

```
/shop/member_html.cgi?file=|cat%20/etc/passwd|
/shop/normal_html.cgi?file=;cat%20/etc/passwd|
/viewimg.php?path=../../../../../../../../../../../etc/passwd&form=1&var=1
/viewpage.php?file=/etc/passwd
/shoutbox/expanded.php?conf=../../../../../../../../etc/passwd%20
/tmp_view.php?file=/etc/passwd
/userreg.cgi?cmd=insert&amp;lang=eng&amp;tnum=3&amp;fld1=test999%0acat</var/
                      spool/mail/login>>/etc/passwd
```

curity checks on the VirtualHost. I published `192.168.1.1` port 80 with a PAT on the internet firewall.

We can start using *mod_security* to mask the web server header with a custom one. Changing the Header is not a strong protection, but can permit you to give away less information than necessary to a potential attacker, making his work a bit more difficult. Be warned that a skilled attacker will be able to obtain the web server version even with this in place (although probably in a less precise way). You can use the following statement to let *mod_security* rewrite the server header:

```
#Change Web Server Header to fool
lamersHeader set "Server" "MWS 0.1b"
SecServerSignature " MWS 0.1b"
```

Now we are going to activate some configuration of *mod_security* to configure what it will check and how. To enable *mod_security* to check information inside an HTTP POST:

```
SecFilterScanPOST On
```

To set the default action to deny, log and return HTTP 404 error (Not Found) on *mod_security* rules:

```
SecFilterDefaultAction
"deny,log,status:404"
```

to enable URL Validation:

```
SecFilterCheckURLEncoding On
```

To permit in request page just ascii character 32-126 (printable ascii characters) and eliminate *dirty* characters you can use:

```
SecFilterForceByteRange 32 126
```

There are lots of other configurations and tuning that can be done. I suggest you to have a look in the well-done documentation included in the package; you can download it from *mod_security* Home Page [5].

I would like to remind you that some of the filters that you can put on *mod_security* can break application functionality stopping legitimate requests that *looks strange*.

It is not easy to eliminate this behaviour, so be sure to better know how the application works and test it quite enough before going to a production environment.

Let me give you an example of application that can be broken with a filter. Suppose you want to stop all the urls that use `../` in the url. Stopping this kind of url make sense because they are used in directory traversal attacks, to access portions of file system that should not be visible. The signature to stop this request is:

```
SecFilter "\.\./"
```

The escape character `\` is there because character . Is a special regular expression (see Regular Expression frame) char (it means any character).

Reverse Proxy

Source: *Wikipedia.org*

A *reverse proxy* is a proxy server that is installed in the neighborhood of one or more web servers. All traffic coming from the Internet and with a destination of one of the web servers goes through the proxy server. There are several reasons for installing reverse proxy servers:

- Security: the proxy server is an additional layer of defense and therefore protects the web servers further up the chain,
- Encryption/SSL acceleration: when secure websites are created, the SSL encryption is often not done by the web server itself, but by a reverse proxy that is equipped with SSL acceleration hardware. See Secure Sockets Layer,
- Load balancing: the reverse proxy can distribute the load to several web servers, each web server serving its own application area. In such a case, the reverse proxy may need to rewrite the URLs in each web page (translation from externally known URLs to the internal locations)
- Serve/cache static content: A reverse proxy can offload the web servers by caching static content like pictures and other static graphical content,

Compression: the proxy server can optimize and compress the content to speed up the load time.

This kind of filter stops a large number of attacks. For example it stop the following attack (example taken from the log of a web server on internet):

```
/..%c0%af../winnt/system32/cmd.exe?/
                        c+dir+c:\
```

This filter can be used to block a family of attacks, but unfortunately sometimes blocks applications developed using .. as a reference to the parent directory.

To write filtering rules in *mod_security* (like with any firewall) there are basically two approaches:

- permit any and block what you know is bad, you block what you are sure willl not block legimitimate activity,
- block any, and permit just traffic you know is legitimate.

The first one is the most widely used approach with *mod_security*, because it is possible with a generic application without a detailed knowledge on how the application works. The latter is used when you need very high security, you know your application very well and you have enough time to conduct detailed tests (to be done every time the application is modified).

A possible criticism to this second approach is that if you have time for testing, and need very high security, you should put all the checks in the application and stay secure. This is partially true because:

- it is not always possible to modify an application for additional security checks (for example there are lots of undocumented but critical legacy applications that need to work),
- can be beneficial to security, checking on the perimeter that application works as supposed,
- can be beneficial to security, having separate equipment that logs every transaction. It probably logs when requests are blocked so it could be used as a miniature IDS.

It is better to underline that filtering some known attacks and some illegitimate transaction can stop a large number of attacks but can't stop them at all. Now we have setup the system and got idea on what it is possible to do, we will start with some filter to block illegitimate activity (or activity that looks illegitimate).

Invalid request filtering

The HTTP 1.1 standard allows a web server to respond to many request methods. Some of the methods are for testing or particular need and are uncommon. Standard methods are:

- *OPTIONS,*
- *GET,*
- *HEAD,*
- *POST,*
- *PUT,*
- *DELETE,*
- *TRACE,*
- *CONNECT.*

The widely most used methods in a standard web server are GET and POST. With the following configuration you will permit just those ones:

```
SecFilterSelective REQUEST_METHOD
"!^(GET|POST)$"
```

This simple rules can create some false positive with some browsers that use the HEAD methods.

When a vulnerability is found for a web server or for a web application, attackers usually try to find a way to make the server do something illegitimate but useful for the attacker. We can try to not permit the attacker to do *standard* nasty stuff,.

For example when a vulnerability is found in a Microsoft application, attackers usually try to execute commands using the *cmd* command in order to do something. Historically we had many viruses propagating with IIS vulnerabilities using the *cmd.exe* command. For this reason Microsoft suggests to configure the OS in a way that doesn't permit the user that runs IIS (IUSR users) to access *cmd.exe* and other operating system commands.

The Listing 3 url are for different vulnerability in IIS or Windows HTTP Server and applications, but they share the fact that the vulner-

Installing from Source

If you need to install *mod_security* from source here you have the steps (from the well done *mod_security* 2.x documentation):

- Make sure you have *mod_unique_id* installed,
- (Optional) Install the latest version of `libxml2`, if it isn't already installed on the server,
- Unpack the ModSecurity archive,
- Edit Makefile to configure the path to Apache (for example: `top_dir = /usr/local/apache2`),
- (Optional) Edit Makefile to enable ModSecurity to use `libxml2` (uncomment line `DEFS = -DWITH_LIBXML2`) and configure the include path (for example: `INCLUDES=-I/usr/include/libxml2`),
- Compile with make,
- Stop Apache,
- Install with make install,
- (Optional) Add one line to your configuration to load libxml2: LoadFile `/usr/lib/libxml2.so`,
- Add one line to your configuration to load ModSecurity: LoadModule security2_module modules/*mod_security2.so,*
- Configure ModSecurity,
- Start Apache,
- You now have ModSecurity 2.x up and running.

You can find instruction on how to install Apache from source code from the Apache homepage documentation [11].

ability has been exploited using the *cmd.exe* command. With the following you will not permit a request containing the text *cmd.exe* to reach the web server:

```
SecFilterSelective
REQUEST_URI "cmd\.exe"
```

Hardening the Installation with FreeBSD Jail

Hardening Operating System and Apache installation can be useful to prevent a compromise of the Proxy Machine or to minimize impact of the compromise. To minimize the impact of an eventual Proxy compromise, I like to use FreeBSD Jail. Basically with Jail you can create a separate environment for a set of process. If a Jail is compromised the attacker will not be able to access the parent operating system data.

You can find full detail for creating an configuring a FreeBSD jail il the nice jail(8) man pages (you can find FreeBSD manuals also on the FreeBSD web site [12]).

Away to exploit a vulnerability of web server on UN*X platform is to download, using a server vulnerability, sensitive operating system files such as the password file (passwd/shadow). This kind of approach has been used widely in a number of vulnerabilities (see Listing 4) and could permit the attacker to obtain the full list of Operating System users and possibly the hash of the password (a good input for a password cracker to obtain the password).

In order to block all these attacks, write the following in the configuration: (Note that this could be circumvented by requesting i.e. `/etc/./passwd`)

```
SecFilterSelective REQUEST_URI "/etc/
passwd"
```

I leave as a reader exercise to try a configuration to block attack to download */etc/passwd* with `./` in the path.

SQL Injection

With SQL Injection vulnerabilities, attackers can send direct command to the DBMS (Data Base Management System) that handles data using the web front-end as a user interface. This kind of vulnerability has a high impact, because if successfully exploited, it can permit attackers to gain illegitimate access to the most critical part of typical web architecture. The way attackers inject a Web Application to send direct command to the DB depends on:

- DBMS technology (SQL Language is standard but every DB has its *standard extension* and dialect),
- Web Application Bug (with full input checking and a secure web application it is not possible to inject the DB),
- Programming language used for Web Development.

we have a large number of variables in SQL Injection attacks and building a regular expression to block any SQL Injection attack is not easy. We are going to try a basic example of SQL Injection to get the idea:

```
http://www.realserver.com/
login.php?username=massimo'
;DROP%20TABLE%20users--
```

The table *users* is dropped after this is performed, if the page is vulnerable. If you have *mod_proxy*, you can insert this rules that block this kind of attack:

```
SecFilterSelective ARGS "drop\busers"
```

This kind of rule avoids the attack, blocking every request that has as an argument the string *drop users*. This rule can help, but probably this attack could succeed:

```
http://www.realserver.com/
login.php?username
=massimo';DROP%20TABLE%20cardnum--
```

As a generic change, you could modify the rule to block the SQL drop statement with:

```
SecFilterSelective ARGS "drop\b"
```

but you could be possibly attacked with:

```
http://www.realserver.com/
login.php?username=massimo';insert%20I
NTO%20users
(usr,pass)%20VALUES('evil','666')—
```

that might be blocked using:

```
SecFilterSelective ARGS "(drop|insert)\
b"
```

You can find signature that block a large number of possible attacks on collections of generic configurations like *mod_security* core rules [3] and gotroot[2], I cannot list all of them here because of the regular expression complexity to block *generic* SQL Injection. Skilled developers and systems administrator wrote lots about this, examples already done can be used to learn and to write custom rules.

Regular expressions are the basis of every rule, so if you want to use the full power of *mod_security*, I suggest you to pay attention to them. The way you write Regex in the rules will impact both the filter semantic and server performance.

Rules to filter SQL attack can be very useful, but like many rules with *mod_security* can have false positive. For example I installed *mod_security* in front of a server farm that have some web mail access. A company hosted in the server farm develop DBMS application and for that reason employee often send email with SQL statements, so with generic filtering you could possibly have false positive (and some legitimate mail blocked). To prevent legitimate traffic blocked you should apply:

- Don't use rules that are not need (for example don't block SQL Inject in an application that does not have a SQL Backend),
- Test your rules before going in production,
- Start configuring your rules to just log traffic that look malicious. After some week and viewing *mod_security* logs you can think start blocking traffic.

Cross Site Scripting

Another *typical* vulnerability for web applications is *Cross Site Scripting*. Summarizing this family of vulnerability is based on:

Regular Expression

You can find some basics of how regular expression work from the Wikipedia definition:

A regular expression, often called a pattern, is an expression that describes a set of strings. They are usually used to give a concise description of a set, without having to list all elements. For example, the set containing the three strings Handel, Händel, and Haendel can be described by the pattern `H(ä|ae?)ndel` (or alternatively, it is said that the pattern matches each of the three strings). In most formalisms, if there is any regex that matches a particular set then there is an infinite number of such expressions. Most formalism provides the following operations to construct regular expressions.

alternation

A vertical bar separates alternatives. For example, `gray|grey`, which is commonly shortened to the equivalent `gr(a|e)y`, can match *gray* or *grey*.

grouping

Parentheses are used to define the scope and precedence of the operators. For example, `gray|grey` and `gr(a|e)y` are different patterns, but they both describe the set containing gray or grey.

quantification

A quantifier after a character or group specifies how often that preceding expression is allowed to occur. The most common quantifiers are `?`, `*`, and `+`:

- `?`: The question mark indicates there is 0 or 1 of the previous expression. For example, *colou?r* matches both color and colour.
- `*`: The asterisk indicates there are 0, 1 or any number of the previous expression. For example, `go*gle` matches ggle, gogle, google, gooogle, etc.
- `+`: The plus sign indicates that there is at least 1 of the previous expression. For example, `go+gle` matches gogle, google, gooogle, etc. (but not ggle).

These constructions can be combined to form arbitrarily complex expressions, very much like one can construct arithmetical expressions from the numbers and the operations `+`, `-`, `*` and `/`.

So `H(ae?|ä)ndel` and `H(a|ae|ä)ndel` are valid patterns, and furthermore, they both match the same strings as the example from the beginning of the article. The pattern `((great )*grand )?((fa|mo)ther)` matches any ancestor: father, mother, grand father, grand mother, great grand father, great grand mother, great great grand father, great great grand mother, great great great grand father, great great great grand mother and so on.

The precise syntax for regular expressions varies among tools and application areas. If interested in better understanding Regular Expression inner workings I highly suggest you read *Mastering Regular Expressions* from O'Reilly [9].

- A user sends malicious code to the server (usually, active content like javascript or vbscript),
- The vulnerable application doesn't recognize the input received from the user as malicious and reproduces it without modification on further request.

Cross Site Scripting can be used in a large number of attacks. Let's see an example to better understand what type of damage can be done:

- Forum. A User posts malicious code on Web Forum or site that permits to publish information with malicious code (for example a javascript code that reads a users' cookies and sends them to a third party site). The site does not check the content and every user that accesses it will expose their cookies, which could possibly be used to log-in to the site,
- Email. Let's say you have a *trusted site* vulnerable to XSS (for example it is very common on site search functionality). You can probably produce URL of the trusted site with malicious code, the user will see in the email a link to a site she/he trusts and will click on it exposing her/his computer to a security risk. Or a XSS hole in a bank site that allows a phisher to create a genuine looking fake login screen on the real bank site.

Let's make an example of what an XSS attack looks like and what approach can be adopted to protect our application with *mod_security*. What is usually done on web applications to test if they are vulnerable to XSS is to put some simple javascript code as input of a dynamic application to see if the application does not check the input and, eventually, reproduce the code. To make a test without potential damage, the *alert* function in javascript (which opens a pop-up windows) is often used. So the input to the dynamic web pages will look like:

```
"<script>alert('XSS');</script>"
```

How the javascript can be injected, strongly depends on how the vulnerable application is developed and what bugs are present. So depending on the language it is developed in and the bug that is present in the input validation the XSS attack will be different. We will find most cross site scripting attacks is the presence of HTML/Javascript tags in the input of Forms.

So we can write a rule that stops the presence of Tags in the input to our dynamic site to stop some XSS attack:

```
SecFilterSelective ARGS "<(.\n)+>"
```

This kind of filter doesn't stop all the XSS attacks as they can be different from the one presented here. You can find more general rules, with complex regular expressions, on [2], [3].

Future functionality of mod_security

I would like to suggest you to try the crypto patch that can be downloaded from the *mod_security* web site[3]. This patch is still not production ready but it is incredibly interesting and useful.

It basically introduces a set of tags for various additional security checks.

The most interesting are the ones for verifying the integrity of links, cookies and hidden fields in forms. That way you can prevent attackers from:

- accessing parts of the site not linked from the homepage,
- tampering with cookies to violate the security of the site,
- tampering with hidden fields.

You can find further information on the crypto patch on *Advanced Web Application Security Defense with ModSecurity* [10]. *mod_security* developers are working to add crypt functionality to production version.

Conclusion

Web application firewall is a very promising technology that is still unused in typical deployments of web services. This kind of technology has demonstrated to be very useful in a large number of installations (*mod_security* is used in server farms with hundreds of web servers protected by its filtering).

I hope this article gave you an idea of what can be done with this kind of technology and how to successfully use it.

Acknowledgment

I would like to thank Antonio Merola for his continuous advice and friendship. ●

About the Author

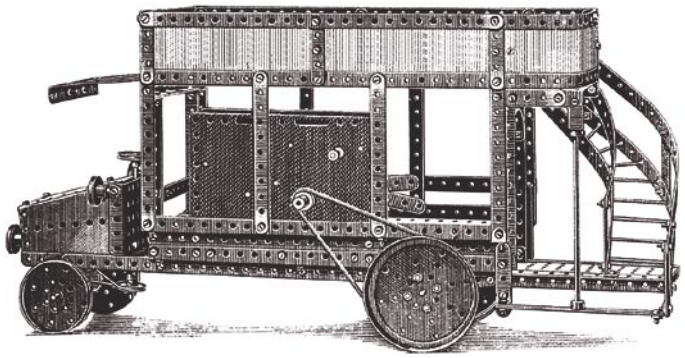
Massimo Fubini works as a Senior IT Security Consultant for a multinational IT company. He has been working from 2000 in the Telco & Finance fields on many aspect of Network Security such as IDS, Firewall, VPN, security assessment, incident detection & management, identity management, privacy & compliance. He has been using free Operating Systems for 10 years and is a big supporter of open source projects and community. Massimo holds a Master Degree in Computer Engineering from the Univ. of Rome – La Sapienza Italy. *M.Fubini@ieee.org*

On the Net

- [1] *www.apachetutor.org/admin/reverseproxies*,
- [2] *www.gotroot.com*,
- [3] *www.modsecurity.org*,
- [4] *www.cgisecurity.org*,
- [5] *www.owasp.org*,
- [6] *www.webappsec.org*, Web Application Firewall Evaluation Criteria,
- [7] Http Standard. RFC2616. *http://www.ietf.org/rfc/rfc2616.pdf*,
- [9] Jeffrey Friedl, *Mastering Regular Expressions*, O'Reilly,
- [10] *http://wiki.whatthehack.org/images/8/8c/Wth-slides-modsecurity.pdf*,
- [11] *httpd.apache.org*,
- [12] *www.freebsd.org*.

Easy Firewalling with IPCop

Robert Larsen



Every company irrespective of its size will have a web site. There is nothing new about it, but with faster Internet connections, cheaper hardware and easier server configuration tools, more and more companies are hosting their own servers. This is also true for many technical home users.

These servers and networks should be protected, and their first line of defense is usually a firewall, but anybody who has read the manual of iptables knows that it can be a hassle to set up a good firewall, and to have it properly configured.

This is where IPCop and similar firewall distributions come in. They make the process of setting up, and configuring the firewall much easier.

With IPCop, a firewall is only part of what you get. Being a Small Office Home Office (SOHO) firewall a whole bunch of other softwares has been packed into the distribution including a DHCP server, web proxy, Intrusion Detection System (IDS), Virtual Private Network (VPN) and much more. Security experts would say that these services have no place on a firewall and should have a machine of their own, and they would be right. But often, small companies and private users do not have the finances, nor time to set up and configure these services separately and hence this could be a compromise worth using.

Planning our work

In this article, we will set up a firewall for a small imaginative company that wishes to host a couple of servers and have a secure LAN for the staffs work stations. We also want to make it possible for users to work from home by setting up a VPN.

But first things first. Let us sketch out the company network as we want it to work as shown in figure 1.

The company has bought eight IP addresses (`62.111.243.80 to 62.111.243.87`), and we assign one of them to the firewall's external interface and the rest will be aliases. The servers are placed in a DMZ that has the network address `10.0.0.0/24`, but the last quad of the servers addresses should correspond to the last quad of

What you will learn...

- how to set up a basic IPCop based firewall
- how to reconfigure IPCop to better fit your needs.

What you should know...

- you should be familiar with the Linux operating system,
- basic networking terms.

the addresses we assigned to the firewall's external connection or one of its aliases. There is no technical reason for this, but it can make it much easier for us to get the big picture. The only services we want to offer to Internet users are HTTP and HTTPS on the web server, and SMTP on the mail server. We haven't used all eight IP addresses but this is a growing company, and hence we will need more servers in the near future.

The workstations are placed on a separate network segment, which is better secured than the DMZ. Everybody in the world should be able to contact the servers in the DMZ but this also opens a hole through which hackers can enter our network, but they should not be able to enter the LAN where customer records and other important private documents are stored. Workstations will get IP addresses from `192.168.10.50` and above. The addresses below 50 are reserved for different Intranet servers. The reason for not choosing the `192.168.1.0/24` network is because, it could clash with our VPN users who probably have routers from their ISP's that by default connects to this network.

We want our VPN users to connect to the virtual network with network address `10.59.190.0/24`. IPCop will indicate the unused IP range, but it is our option to choose any range.

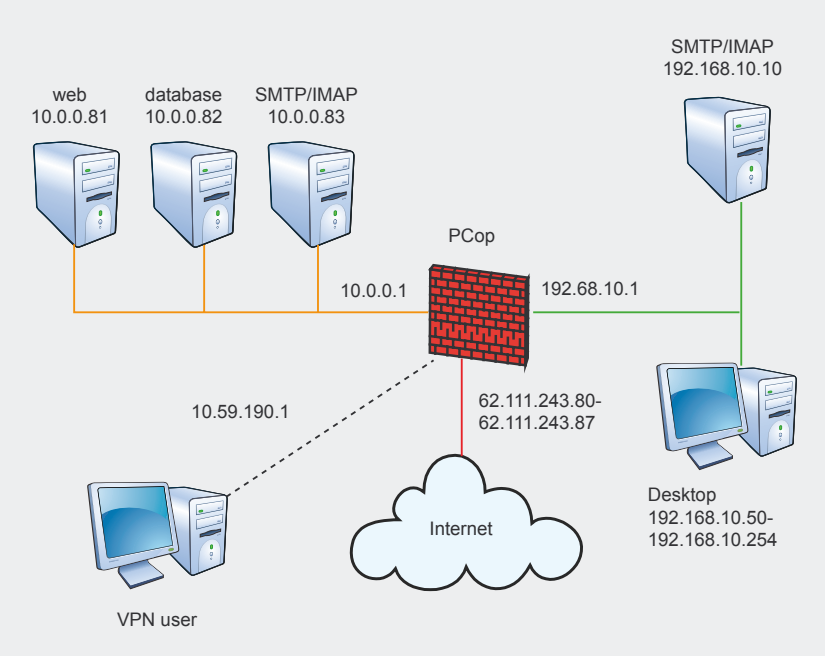


Figure 1. *Company network*

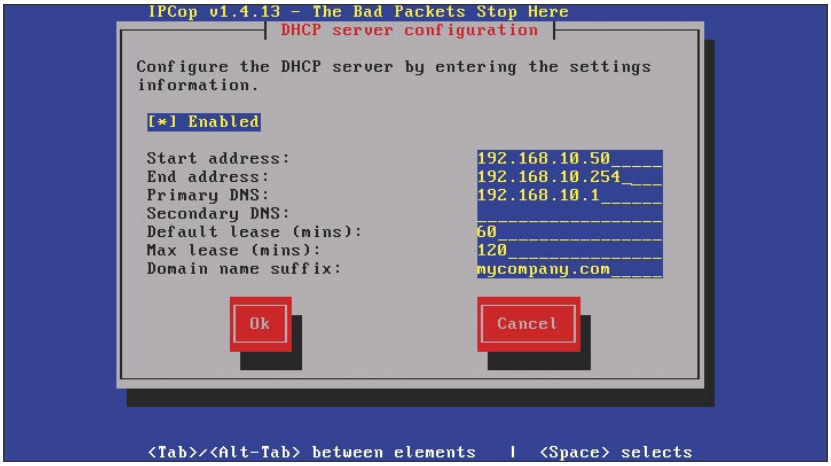


Figure 2. *DHCP configuration*

Install IPCop

Now that we know what we want, let us start with the installalation. First, go to *http://www.ipcop.org* and grab the latest iso, and burn it to a CD. As of this writing the latest version is `1.4.13`.

IPCop does not require a powerful machine to run, but it depends on the number of simultaneous connections that an IPCop can run on a 486 with 32 MB RAM. More realistically, if you want to host servers, you would need more RAM and a faster processor but more or less any old used machine can be used for the same. Of course the more services you enable on the firewall the more CPU power and RAM you will need but IPCop can show you lots of statistics that can tell you how much more the machine can take. For our network, three network interfaces will be needed.

Insert the CD into the firewall machine and boot it up. You will be greeted with a warning that all your data on the firewalls hard disk will be lost if you chose to continue. If you can live with that, just press enter. You also have the option to specify boot parameters and load drivers, if your machine has special hardware. If you continue, you will see a normal Linux boot, and then the installer menu system is started. First you are asked to choose a language for your installation. IPCop has been translated into 22 languages so far. The language can be changed at a later point also, depending on the need.

Then, you can choose the installation media. You can choose between CD-ROM/USB and HTTP/FTP. The network installation could be chosen if the computer lacks a drive, and if we have another machine on the LAN which has the IPCop software, in which case we would have booted the firewall over the network. Since we have a working CD drive lets use that.

When we have chosen the installation medium IPCop probes for a disk to install on, and will warn us another time that the disk will be repartitioned and formatted.

After the disk has been formatted, we have the option of restoring the configuration from a backup, from either a floppy, USB key or HTTP/FTP server, but since this is our first installation, we do not have a working configuration so we can skip that.

Now we must set up the network interfaces and assign roles to each. IPCop offers to probe for a network interface, which will get the green role. We can also select a driver ourselves, but we will just let IPCop do the work, as it usually works. When the green interface has been found, we must give it an IP address, and looking at our sketch we already have decided to give it `192.168.10.1`.

After this is done, we are told that IPCop has been successfully installed and that we can reach the IPCop web server and do further configuration with a web browser. But configuration is not entirely done yet. Next we are asked to specify a keyboard mapping and a timezone.

Then we choose a host name (I usually keep the default name which is *ipcop*) and a domain name.

Our firewall is not worth much without an uplink. The next thing we have to do is to configure the red interface. We are presented with a menu for configuring an ISDN modem but we also have the option of disabling ISDN.

Having done that we can configure the network configuration type. This presents us with different mixtures of red, green, orange and blue interfaces. As we want an Internet interface, a DMZ and a LAN, all ethernet cards, we will choose the *GREEN + ORANGE + RED* configuration. Next, we have to assign drivers and network cards to the different roles. IPCop finds two unused cards and we assign them to the orange and red interface respectively. Then the two unconfigured interfaces needs an IP address and we take another look at our sketch. The orange IP should be `10.0.0.1` but the red has eight. It can only have one for now so we will just choose the first: `62.111.243.80`. We will create aliases for the other addresses later.

If we were setting IPCop for a home network and our IP address was dynamically assigned using DHCP or some other protocol, it is not a problem for IPCop. This is a special feature for the red interface which can have either a static or a dynamic address.

Now, we need to provide IP addresses for the default gateway and primary and secondary DNS server. You can skip this if your ISP provides this information through a DHCP server.

Next thing we need to do, is to configure the DHCP server for our LAN. If we do not want one, we can just skip this part leaving the *Enabled* option unchecked. But we want one and looking at the sketch again we see that the first 49 addresses are reserved for Intranet servers. We fill the menu as shown in figure 2.

The last thing we need to do is to specify a root, admin and backup password. *root* is the normal Unix root user, the *admin* user is allowed to do configuration through IPCops web interface and the *backup password* is used for encrypting the configuration during backup.

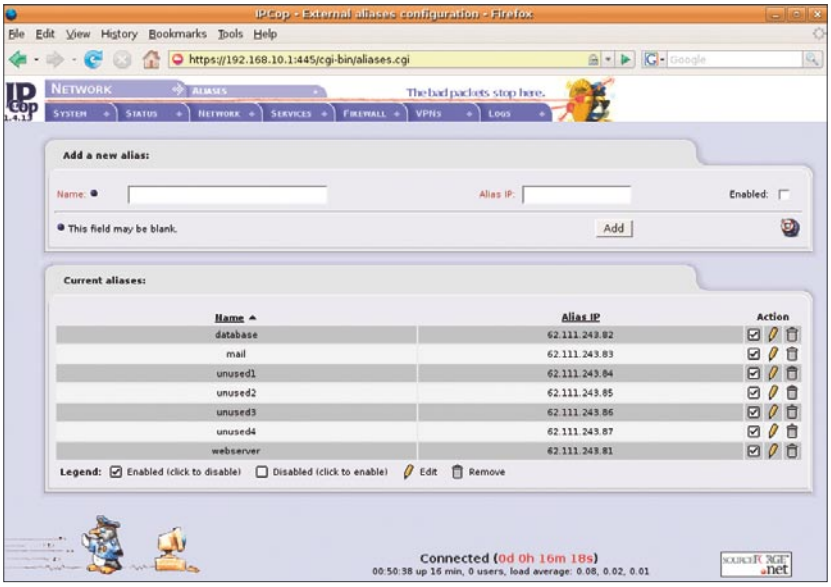


Figure 3. *Aliases*

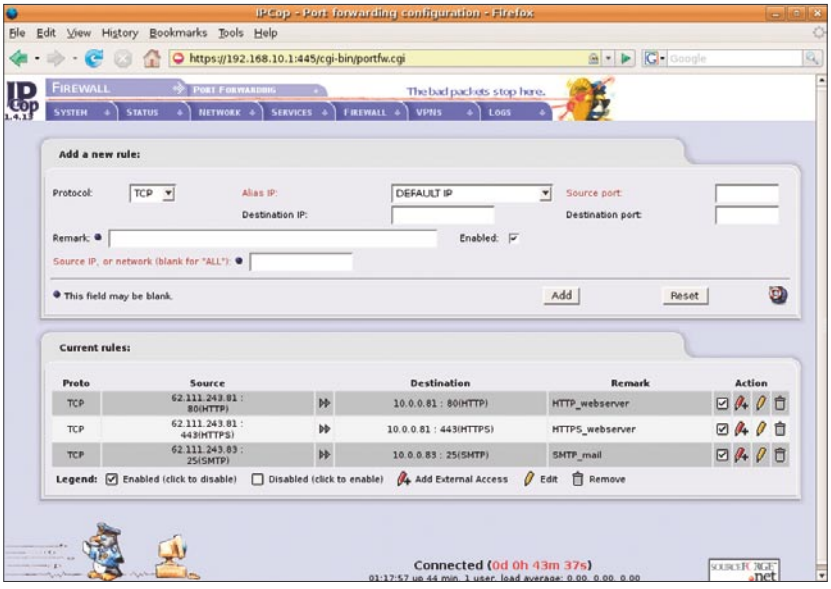


Figure 4. *Port forwarding*

Now it is time to reboot. This may seem like a long process but when I rush it, it takes less than five minutes.

Post install configuration

Now that IPCop has been installed, we can start configuring port forwarding and other stuff. To do this, we need a machine on the LAN. This is a security precaution for not letting everyone to access the configuration. We point a browser to the IPCop web interface which is *http://192.168.10.1:81*. It actually redirects us to *https://192.168.10.1:445/cgi-bin/index.cgi* so that the entire configuration setup happens over an encrypted channel. We are greeted with IPCops friendly front page, which gives us the duration of machine connectivity and the time when IPCop last checked for updates and maybe that it is time to check again.

IPCop is under active development but very easy to update. We can check for updates through the menu system on the top of the page. It is placed under *System->Updates*. The first time we access a submenu we are prompted for a username and a password. This is the *admin* user and password that we set during the install. When we have entered this we can start reconfiguring the system.

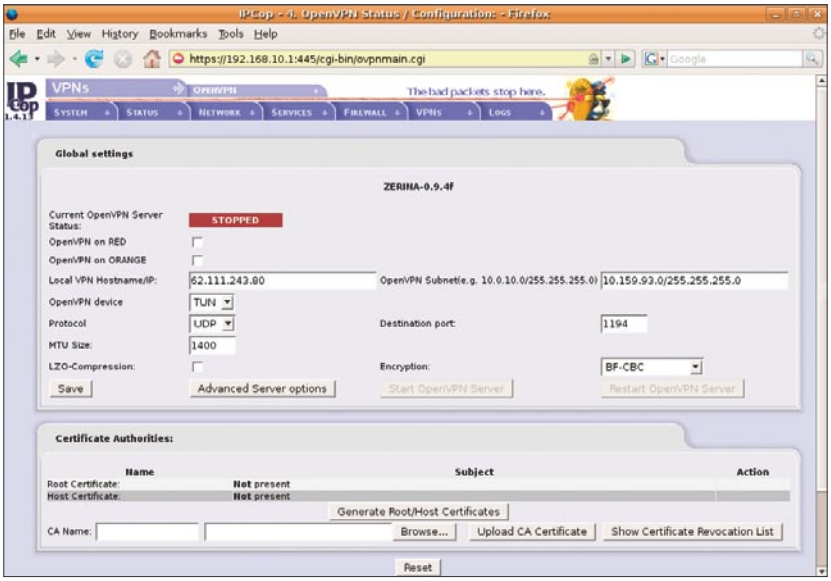


Figure 5. *OpenVPN*

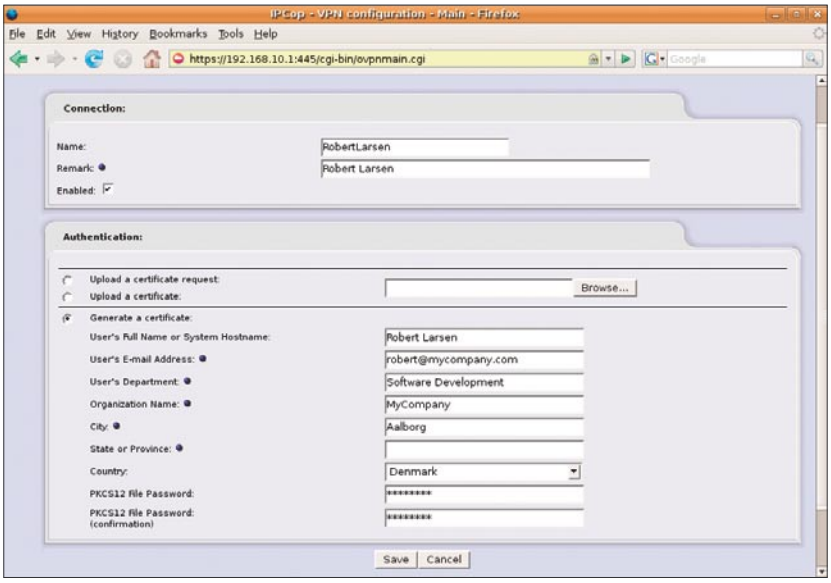


Figure 6. *Adding OpenVPN user*

On the updates page we can press the *Refresh update list* button, after which IPCop checks for updates. If one or more updates exist, they will be shown on the top together with a description of which bugfixes and features it provides and whether or not a reboot is required after the update. If we decide on upgrading the system, we need to download the file and upload it to the firewall and maybe reboot the machine. Rebooting is done through *System->Shutdown* where we can provide a time and day that the reboot should occur. That way we can wait until 4 AM where we probably don't have as many users as other points of the day.

Other interesting menu points in the *System* menu are *SSH access*, *GUI settings* and *Backup*. IPCop by default does not allow SSH access, but the SSH daemon can be enabled. SSH usually listens on port 22, but the IPCop configuration has changed this to port 222. This may be security by obscurity, but most automated attacks only try port 22, so the changed port does provide some extra security. We will need SSH access later on.

GUI settings is where we can change the language used by IPCop. We can also enable and disable JavaScript on the web pages.

IPCops backup system allows us to copy the entire configuration directly to a floppy disk or download an archive. This backup can prove valuable if something should goes wrong or if we want to switch to a more powerfull machine. Remember that during the installation we had the choice of loading the configuration from a backup?

Under the *Status* menu there are also several items worth looking at. These include, different statistics such as memory/CPU/disk and network usage. It is a very good idea to monitor these regularly. They provide clues to, whether or not we should enable different features and when

the network capacity is getting too high. If you have many current connections you should be carefull not to choose the *Connections* submenu. It lists information on all connections and can bring down the machine.

The *Network* menu contains submenus for configuring different modems, and to upload firmware for USB modems. As we have an ethernet card, those are not interesting to us. The last item on the list, *Aliases*, is however (this is not visible if we use DHCP on the red interface). This is where we assign extra IP addresses to our red interface. Remember that we have eight IP addresses, but we only assigned one so far. Each alias will be given a name and I usually make each alias correspond to exactly one server on the DMZ and name that alias after the server. That leaves us with the configuration shown in figure 3.

The *Services* has loads of interesting submenus. Experts would probably say that most of these are out of place on a firewall but again, this is a SOHO firewall and we want the simplicity. We can choose not to enable any of them but that would be boring.

Listing 1. *Nmap port scan:*

```
robert@robert-laptop:~$ nmap -P0 62.111.243.81 62.111.243.83

Starting Nmap 4.10 ( http://www.insecure.org/nmap/ ) at 2007-01-22 21:21 CET
Interesting ports on 62.111.243.81:
Not shown: 1677 filtered ports
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https

Interesting ports on 62.111.243.83:
Not shown: 1678 filtered ports
PORT   STATE SERVICE
25/tcp open  smtp

Nmap finished: 2 IP addresses (2 hosts up) scanned in 105.224 seconds
robert@robert-laptop:~$
```

First, we have the option of enabling a Squid web proxy server on the green network interface. We can even do this transparently. We can also configure different parameters for the proxy, such as specify an upstream proxy and how large the cache should be. This can prove very useful especially if the employees frequently send youtube links to each other.

The DHCP server also lies under the *Services* menu. We have already configured it but we can also add fixed leases and view a list of current leases under this menu.

Figure 7. *OpenVPN client status and control*

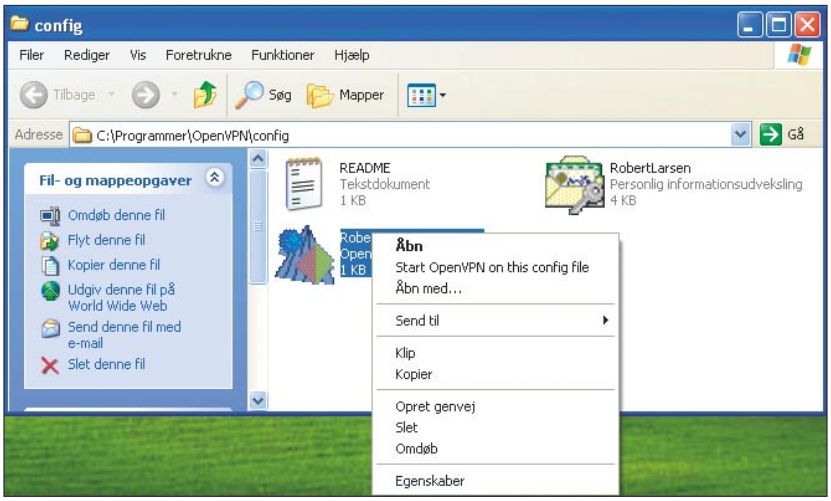


Figure 8. *Starting OpenVPN on Windows*

If we were setting up IPCop for a home office and we had a dynamic IP address the next item on the *Services* list could prove very useful. The *Dynamic DNS* allows us to register our current IP address with *dyndns.org*, whenever IPCop connects to the Internet. But since we have a fixed IP we can ignore this.

IPCops hosts file can be edited under the *Hosts* menu. This provides, sort of a small DNS server on the green net. If we add an IP and hostname to the hosts file, any machine connected to the green network can look that hostname up and get the IP address we specified. That way we can name all our servers on both the green and orange network without having to share those names with the rest of the world and without setting up a real DNS server. Usually you would have intranet servers on the green net which no outsiders should care about so naming them in IPCops hosts file is a simple solution.

Next we have a time server submenu. This can be used for setting IPCops clock, but also to enable an NTP server on IPCop to provide a time service for our networks.

We also have a trafic shaper under *Services*. This can be used to prioritize the different traffic our firewall handles. This way we can assure that our employees MP3 download don't get in the way of our customers access to our servers... or the other way round.

Last submenu in the *Services* list is *Intrusion detection*. This is an installation of Snort which can be enabled for each network interface separately. You will need an oink code which is sort of a license key but it can be gotten for free on *http://www.snort.org*. When the code has been entered, new rulesets can be downloaded and installed with a single click on a button. A warning is in place as Snort does need a lot of CPU power if you have much traffic so keep an eye on the system graphs both before and after you enable Snort.

Enabling the DMZ

The next menu is where the magic happens. The *Firewall* menu contains of course firewall configuration options. The first on the list is *Port forwarding* and this is what brings our DMZ alive. Here we can specify that TCP traffic destined for port 80 on the IP alias we named *webserver* should be redirected to IP `10.0.0.81` port 80 and name that port forward rule *HTTP_webserver*. We want to open up for three services in our DMZ: HTTP and HTTPS on our web server and SMTP on our mail server. That gives us the configuration shown in figure 4.

The *External access* menu point allows us to open up the firewall for outsiders. We don't want to do that so just ignore this.

DMZ pinholes allows us to make holes from the DMZ to the LAN. This can be useful if our web server should be allowed to access customer data or other data that are too sensitive to be on the DMZ. We don't want to do this either.

Finally the *Firewall options* allow us to prevent ping responses. Ping is useful to the administrators on the LAN so we only disable ping for the red interface.

After this I do a portscan to check that everything works as expected as shown in listing 1. Heureka... it works!

Adding VPN access

But we still need to set up a VPN for our hard working road warriors. The next menu is the *VPN* one. The VPN server that comes with IPCop is based on IPSec and is geared towards connecting two networks. Configuring for road warriors is rather complex so instead we turn

Listing 2. *Installing the Zerina OpenVPN addon:*

```
robert@desktop:~$ scp -P 222 ZERINA-0.9.4f-Installer.tar.gz
                     root@192.168.10.1:
root@192.168.10.1's password:
ZERINA-0.9.4f-Installer.tar.gz                100%  349KB 349.4KB/s   00:00
robert@desktop:~$ ssh -p 222 root@192.168.10.1root@192.168.10.1's password:
Last login: Mon Jan 22 02:01:56 2007 from 192.168.10.254
root@ipcop:~ # cd /tmp/
root@ipcop:/tmp # tar xfz /root/ZERINA-0.9.4f-Installer.tar.gz
root@ipcop:/tmp # ./install

Welcome to the OpenVPN for IPCop (ZERINA 0.9.4f) Installer!
The Installer will perform the following steps :

1. Looking for older versions of ZERINA
2. Checking for valid IPCop-version
3. Backing up files for easy uninstall
4. Installing new files
5. Adding entrys to existing files

Starting installation process:

1. Looking for older versions of ZERINA .
2. Checking for valid IPCop-version .
OK!
openvpn.tar.gz
ZERINA-0.9.4e_Libs_Modules.tar.gz
langs/
langs/bz.zerina
langs/da.zerina
langs/de.zerina
langs/en.zerina
langs/es.zerina
langs/fr.zerina
langs/sv.zerina
misc/
misc/header
misc/header.old
misc/rc.firewall.local.start
misc/rc.firewall.local.stop
misc/rc.local
misc/rc.local.zerina
misc/status
OK!
3. Backing up files for easy uninstall ..OK!
4. Installing new files ....OK!
...OK!
5. Adding entrys .......Cannot read ENABLED

ZERINA Installer finished

   * You can now access OpenVPN via the IPCop web gui.
   * WARNING: This package is NOT an official IPCop addon. It hasn't been
     approved or reviewed by the IPCop development team. It comes with NO
     warranty or guarantee, so use it at your own risk.
   * WARNING: You have to create your own certificates for OpenVPN!
   * For Updates/Information/etc try: www.zerina.de
   * howto : http://www.zerina.de/?q=documantation
   * For support try: www.openvpn-forum.de

root@ipcop:/tmp #
```

to one of the many addons for IPCop. On *http://www.zerina.de* you can download an addon that installs an OpenVPN server on IPCop. This makes road warrior configuration so much easier. As of this writing the latest stable version is 0.9.4f. Download this and enable SSH on IPCop. Next, scp the plugin to IPCop (remember sshd runs on port 222 on IPCop). Now, ssh to IPCop and change to the /tmp directory. Most plugins should be unpacked to /tmp including Zerina. Now, just run the install script as shown in listing 2.

Now that the addon has been installed, a new menu entry has popped up under VPN and this is OpenVPN. Open up this and you will see the screen shown in figure 5.

As you can see the *Start OpenVPN Server* button has been grayed out. This is because we must first specify which interfaces to allow VPN connections from and generate a certificate. Just check the *OpenVPN on red* option and enable LZO compression and leave the rest. Click *Save*.

The certificate is a signed statement that this server is who it says it is. For a company I would recommend getting a certificate from a certificate authority such as Netcraft or VeriSign. There are also free certificate authorities such as *Cacert.org*.

You can however also create your own which will then not be signed by an authority and less trustworthy and this is what we'll do as this is easier. Just push the *Generate Root/Host Certificates* button and fill out the form and press *Generate Root/Host Certificates*. Now you can start the OpenVPN server.

Adding users is equally easy. On the bottom of the OpenVPN configuration page under *Client status and control* you push *Add*, choose *Host-to-Net Virtual Private Network (RoadWarrior)* and fill out the new form as shown in figure 6.

Let the user type in his password himself or if this is impossible because he is somewhere else in the world you can let him create his own certificate and just upload that.

Now that we have added a VPN user the *Client status and control* part of the OpenVPN page looks as the one shown in figure 7.

The blue icon shown next to the *CLOSED* status is a link to a zip file containing the users certificate and an OpenVPN configuration file. Download this and give it to the user.

Installing OpenVPN on a Linux system is piece of cake. On most systems the package manager can do it for you but if not the sources are not hard to compile yourself. On Ubuntu a simple `sudo apt-get install openvpn` will do it. Then copy the certificate and configuration file to `/etc/openvpn` (perhaps you need to rename the configuration file extension from `ovpn` to `conf`) and then

IPCops Colored Interfaces

IPCop can assign four different roles to network interfaces and each role has a color. A red interface should face the internet. The firewall rules assume that the packets arriving on the red interface are dangerous and should be considered hostile. When configuring port forwarding for the different servers in the DMZ, IPCop is forwarding traffic from the red interface to the servers in the DMZ. A green interface faces the LAN. All traffic on this interface is trusted and can go unhindered to both the DMZ and the Internet. This interface is also the only one that is allowed to configure IPCop. An orange interface faces the DMZ where to traffic coming to the red interface can be forwarded and finally a blue interface faces a wireless access point. We do not configure this but if we would allow wireless access to the LAN, this would be needed. Wireless traffic can only get to a VPN server, but if a user can authenticate with the VPN server, he is granted access to the LAN.

VPN

A Virtual Private Network (VPN) is like an imaginary ethernet card which is plugged into a switch somewhere far away. It works by starting a VPN daemon which authenticates with a VPN server in the target network. If this succeeds a virtual ethernet card is created with an IP address which the VPN server grants you. Also your routing information is updated so IP traffic destined for the remote network are routed through the virtual ethernet card. When a packet is sent out the virtual ethernet the VPN encrypts it, signs it and sends it to the VPN server through the physical interface. The VPN server verifies the signature and if it checks out right the packet is decrypted and routed to its destination.

Basically two setups are possible:

- *Net to Net*: In this setup two remote offices want their users to be able to see each other as if they were connected to the same network. The default gateway at the first office then works as a VPN client for all clients connected to the network at the first office. All network traffic that cannot be handled locally is then sent to a VPN server at the second office which then tries to handle it. This is transparent to the users on the network.
- *Host to Net*: In this setup we want a single machine to connect to a network. This is typical when a company allows its users to work from home or when visiting customers. It works like multiple *net to net* setups where each VPN user configures and connects his own VPN gateway.
- Before getting too excited by the wonders of VPNs there are also problems. Workstations at the office typically can be administered by people who know what they are doing and can be forced updated. The users machines at home typically can not so if a users computer at home is hacked it would be the same as if a machine on the company LAN had been hacked. Companies often handle the risks by providing laptops or desktops to employees that wish to work from home and install software so the administrators can reconfigure and update the machines at all times.

start the daemon with `sudo /etc/init.d/openvpn start`. You will be prompted for your certificate password. Now the machine should be connected.

For Windows users things are only slightly different. Go to *http://openvpn.net* and download the Windows installer. Install that and then unpack the content of the zip file in `c:\program files\OpenVPN\config` or where you installed OpenVPN. You can then connect by right clicking the config file and choose *Start OpenVPN on this config file* as shown in figure 8.

Enter your password and that should be that. There has been problems reported with Windows Vista but there appear to be a solution. See link [3] for more information.

The default installation of OpenVPN only allows VPN users access to the LAN, not the DMZ. This however can be changed on the server. SSH to IPCop and add the following to `/var/ipcop/ovpn/server.conf`:

```
#Route to 10.0.0.0/24 network
push "route 10.0.0.0 255.255.255.0"
```

On the Net

- [1] *http://www.ipcop.org/* Main IPCop site,
- [2] *http://www.zerina.de/* OpenVPN addon for IPCop,
- [3] *http://openvpn.net/* OpenVPN clients for Unix and Windows,
- [4] *http://openvpn.se/bb/viewtopic.php?t=911&postdays=0&postorder=asc&start=0* Discussion and solution to OpenVPN problems on Windows Vista.

About the Author

Robert Larsen works as a game framework developer and server administrator at one of Denmarks largest gaming sites (hosting *www.komogvind.dk*, *www.playandwin.co.uk*, *www.spielmit.com*). His interests include programming and network security.

Then restart the OpenVPN server. That should do it.

Now what?

We now have a fully functional and fully configured firewall which should keep the bad guys out of our LAN and only provide access to few services in our DMZ. Also our staff can work from home via our VPN so where do we go from here ?

I suggest as we have fulfilled our initial goals, we will wait for a while. Check out the system graphs and logs and if they say that the machine can handle more load, we can try adding more services such as web proxy and intrusion detection.

We could also head to IPCops web site and take a look at the addons such as virus scanning of all web, ftp and mail traffic.

If we need IPCop to do something that the menu system does not help us with, we can just ssh to the machine and put something in the init files. `/etc/rc.d/rc.local` is a good place to put these things. For example, when machines on the LAN or DMZ connect out they appear to come from IP `62.111.243.80` as this is the IP we gave the red interface. This can cause problems for sending mail, as some of the mail servers do a reverse DNS lookup before accepting mail. To make our mail server appear to come from IP `62.111.243.83`, add this line to `/etc/rc.d/rc.local`:

```
iptables -t nat -A CUSTOMPOSTROUTING
-o eth2 -s 10.0.0.83 -j SNAT
 -to-source 62.111.243.83
```

What the future brings

Our installation of IPCop runs kernel 2.4.34, but we speculate that the upcoming version 1.5 will run 2.6. There has been a suggestion on the IPCop developers mailing list to add the OpenVPN addon to the standard IPCop install, as this greatly simplifies the configuration of road warriors and because the Zerina project is looking for someone to take over further development. ●

Introduction to Anti-spam Practices

Alina Popescu

Competitive Antispam products, proper legislation, efforts towards a better user education, it has all been tried in order to stop spam. However, unsolicited emails keep consuming the space and time of all email users. Moreover, spam messages can be the cause of serious virus and spyware outbreaks, while others *phish* for sensitive information like bank accounts and passwords.

According to a research conducted by Microsoft and published by the Radicati Group, the percentage held by spam in the total number of emails sent daily has been constantly growing since 2005. As a result, spam is expected to represent 77% of emails sent worldwide by 2009, amounting to almost 250 billion unsolicited emails delivered every day.

In a world where spam is bound to hold such an important position, methods of preventing it should also be given an increasing importance. Some of the easiest and most widely used prevention methods are host control solutions, Antispam applications and user education.

Host control is an easy way to ensure only valid emails reach end-users' inboxes. Some well known methods are SPF (*Sender Policy Framework*), IP/email address-based lists (blacklisting, whitelisting and graylisting) and DKIM (*Domain Keys Identified Mail Signature*).

SPF

The *Sender Policy Framework* (SPF) is an open standard specifying a technical method to prevent sender address forgery. It protects the envelope sender address, which is used for message delivery. The envelope sender address is used during the transport of the message from mail server to mail server, usually not displayed to the user by mail programs.

Using this method, domains can publish details of their mail sending policy (called SPF records) on *Domain Name System* (DNS) servers. By using SPF checks to validate sender addresses, you can successfully prevent spam and back-scatter emails. Although an effective method of authentication and spam prevention, not all MTAs and ISP providers support SPF checks at this time.

What you will learn...

- what SPF is,
- what DKIM is,
- overview of the ways of sending spam.

What you should know...

- what spam is in general.

DKIM

Domain Keys Identified Mail Signature is an authentication method implemented by Yahoo and supported by Google, Cisco and Sendmail and has considerable chances of becoming the standard authentication method. It offers almost end-to-end integrity from a signing to a verifying Mail transfer agent (MTA). In most cases the signing MTA acts on behalf of the sender, and the verifying MTA on behalf of the receiver.

DKIM implies using a key pair consisting of a public key and a private one as follows: the signing MTA generates a public key, which is published in DNS, and a private key, used to digitally sign all the sent email messages. The verifying MTA retrieves the public key and compares it to the digital signature of the received email. If the key pair is a match, then the email is legitimate and is delivered to the receiver's mailbox.

The wide use of DKIM can force spammers to show a correct source address. Thus other filtering techniques (such as collaborative databases) can be used to detect spam more reliably. Therefore, DomainKeys can make it easier to identify emails known to be legitimate and need not be filtered. The main benefit in such a case would be saving time and system resources.

The main disadvantage of DKIM is that email messages can be significantly modified in certain situations (e.g. when being forwarded by list severs), causing the signature to be invalidated and the message to be rejected. A solution to this issue would be combining *DomainKeys* with SPF, because SPF is immune to modifications of the email data.

Blacklisting

DNS blacklisting is the practice of comparing the routing addresses of incoming e-mails to a list of servers that spammers are suspected to use.

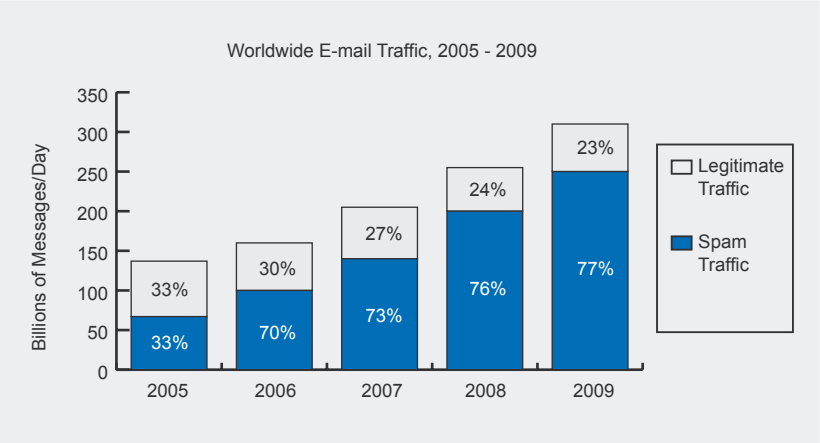


Figure 1. *Legitimate Email Traffic vs. Total Spam Traffic, 2005-2009*

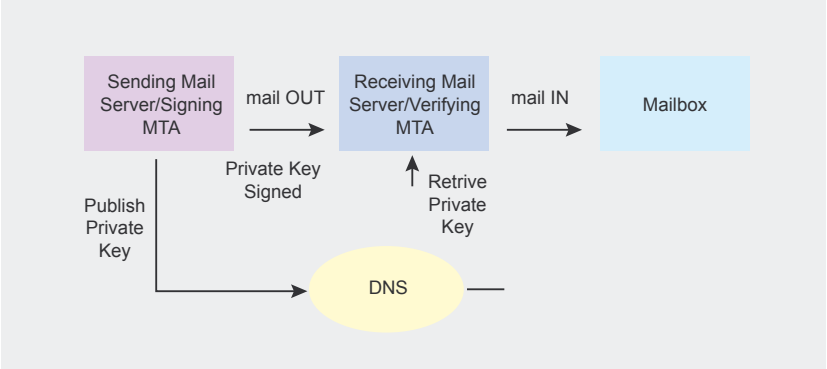


Figure 2. *DKIM Compliant Email Flow*

Blacklists are either public (free of charge) or private and usually contain IP addresses of open-relay servers, open proxies and ISPs with no spam filtering. If an e-mail appears to be from a blacklisted server, it is blocked, usually with an error message for the recipient.

The main advantage of using blacklist is their convenience. Instead of installing and/or training a product to block spam, one can use blacklists to perform the needed email filtering.

On the down side, blacklists never provide an easy way of tracking what and how much is actually being blocked. Moreover, they are known to be highly inaccurate. A given block of addresses may appear on a blacklist, simply because one computer in that block may be a spammer. This causes everyone in the block to be unable to send or receive emails.

Whitelisting

Whitelists are the exact opposite of blacklists. Instead of filtering possible spammers, they keep track of secure IP addresses or email addresses. They are lists of contacts that the user deems are acceptable to receive email from. An email whose sender is on such a list will be accepted with no further filtering.

Internet service providers are known to use whitelists to filter emails to be delivered to their customers. ISPs receive requests from legitimate companies to add them to the ISP whitelists. Companies either pay for a time period to be allowed to email their customers or the companies pay per complaint received by the ISP from their customers.

The main disadvantage of such lists is that, when set incorrectly, they may reject legitimate emails. However, such incidents tend to appear at end-user level, not at server or ISP level.

Graylisting

Graylisting is basically a request to have the email resent, after being temporarily rejected by a *Mail Transfer Agent* (MTA). The sender IP and

the recipient of all unknown senders are saved by the MTA which then returns a temporary error. Based on the fact that spammers and spamming scripts do not resend messages, only valid servers will react to this temporary error.

Greylisting requires no additional configuration made by end-users. If the server utilizing greylisting is configured accordingly, the only effect on end users will be a delay on the first message from a given sender. Moreover, far less system resources are used on sending temporary errors than on using Antispam software.

Many MTAs cannot differentiate at this time between a temporary and a permanent error, thus important emails may be lost. This can be prevented however by using whitelists. Greylisting delays much of the mail from non-whitelisted mail servers – not just spam – until typical patterns of communication are recorded by the greylisting system. Therefore, it cannot always prove to be an optimal solution.

Anti-spam applications

Software products specialized in preventing spam can be used by end users and by MTA and ISP administrators as well. Open source or commercial, there is a wide range of such solutions to choose from, depending on available resources.

Besides the available host control methods, the filtering system provided by Antispam solutions also embeds content filtering. Thus, they can implement lists of keywords or images most likely to be used by spammers.

An important feature of certain content filtering chains is their use of *Bayesian filters*. Such filters divide email into spam or legitimate based on the probabilities of certain words or phrases to occur in both types of messages. For example, most email users will frequently encounter the word *Viagra* in spam email, but will seldom see it in a legitimate email. However, the filters do not know these probabilities in advance. To train the filter, users must manually indicate whether a new email is spam or not. Based on all words in each training email, the filter will assign probabilities to each word in its database, both for spam and legitimate messages.

The main advantage of Bayesian spam filtering is that it can be trained on a per-user basis. The training, however, is time consuming. Moreover, incorrectly identifying emails as spam or legitimate may lead to having to restart the whole training process.

Educating End-Users

Although sometimes overlooked, this aspect is one of the most important aspects of fighting spam. Incorrectly handling existing spam messages often leads to receiving more unsolicited emails.

The most common example is represented by unsubscribe links included in spam messages. A large number of those receiving spam are tempted to click on such links contained by unsolicited emails. Such an action will let the spammer know there is a live person with a valid email address. The email address will be tagged as a good one and can later be sold to other spammers. In a worse case, the link could lead to a website that installs a virus or spyware, leading to severe security issues.

According to a study conducted by Radicati Group and Mirapoint in March and April, 2005, 55% of email users follow the unsubscribe links in spam messages. The correct procedure in such cases is to permanently delete such messages.

Making sure all users know how to treat spam is therefore the first step to be taken in order to reduce this phenomenon. It should also be doubled by research and development in technology and better laws for this field being made available worldwide. However, without proper user knowledge, technology is almost useless and laws cannot be applied; therefore the troubling predictions on future spam expansion may easily come true. Permanent efforts to educate users are not a full proof defense against unsolicited mail, but represent the only optimal available protection. ●

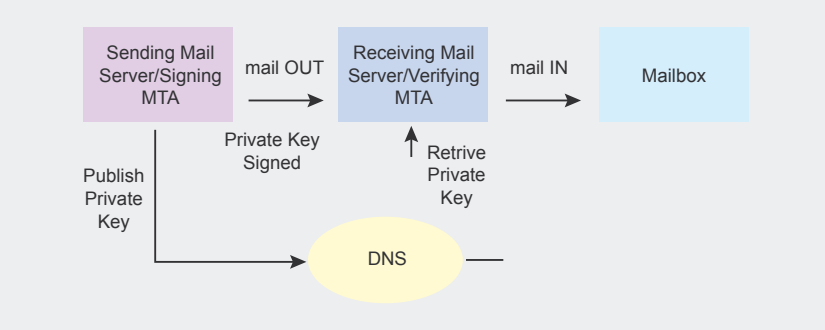


Figure 3. *Users that Follow the Message's Directions to Unsubscribe*

On the Net

- *http://www.radicati.com/reports/whitepapers.asp#* – A list of public whitepapers,
- *http://antispam.yahoo.com/domainkeys/* – DomainKeys: Proving and Protecting Email Sender Identity,
- *http://www.openspf.org/* – The Sender Policy Framework project.

About the Author

Alina Popescu, 24, is a Journalism graduate and has been working as a Technical Writer for AXIGEN (*http://www.axigen.com/*) since August 2006. She has previously published several articles on sites such as *HowToForge.com*, *LXer.org*, *EzineArticles.com* and *Librenix.com*. More pieces written by Alina can be found at: *http://www.axigen.com/mail-server/articles.php/.*

great subscriber offer

SAVE $99.99!

Get your copy of hakin9 and save 60% off shop prizes

Free easy ways to order

- *visit: www.buyitpress.com/en*
- *call: 0048 22 887 10 32*
- *e-mail: subscription@software.com.pl*
- *fill in the form and post it*

hakin9 ORDER FORM

☐ **Yes**, I'd like to subscribe to ☐ *hakin9 or* ☐ *hakin9 starter kit* magazine (6 issues a year)
☐ USA $49 ☐ Europe 39€

☐ **Yes**, I'd like to subscribe to *hakin9 and hakin9 starter kit* magazine (12 issues a year)
☐ USA $79 ☐ Europe 69€

Order information
(☐ individual user/ ☐ company)

Title
Name and surname
address

postcode
tel no.
email
Date

Company name
Tax Identification Number
Office position
Client's ID*
Signature**

Payment details

I understand that I will receive selected number of issues over the next 12 months
☐ Master Card ☐ Visa ☐ JCB ☐ POLCARD
☐ DINERS CLUB

Card no. ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐
Expiry date ☐☐☐☐ Issue number ☐☐

☐ I pay by transfer: Nordea Bank
IBAN: PL 49144012990000000005233698
SWIFT: NDEAPLP2

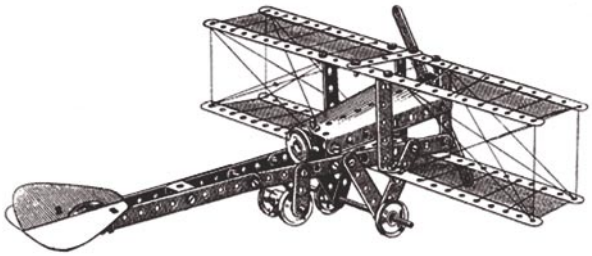
Signature

Terms and conditions:
Your subscription will start with the next available issue.
You will receive 6 or 12 issues a year.

* if you already are Software LLC client, write your client's ID number, if not, fill in the chart above
** I enable Software LLC to make an invoice

Popular Free Software Firewalls for Home/Personal Use

Josh Sawyer



There are many free software firewalls for home use, but there are varying opinions on which is bestin terms of security, user-friendly interfaces, overhead, etc. This simple review is meant to help the home user select the best firewall for their application.

Although many experts consider Windows to be inherently insecure by design, as has been evidenced by the numerous exploits found against the Microsoft OS, the average Windows system can be made relatively secure against most threats. The use of a firewall is a necessity for computer users today, and if used together with other security solutions and user education, Windows can be significantly hardened against the most common threats.

While many free firewall solutions exist for home users, this review selects a two of the most widely used and compares them to provide the user with a concept of which is best for them. It cannot cover every possible area/function however.

The programs reviewed were: *ZoneAlarm*, and *Sygate Personal Firewall*. The test system was Windows XP, SP2. Each was installed on the same system at different times, removing the previous firewall before testing the next. *ZoneAlarm* was installed first and the install asked a few questions from the user which may confuse some, but overall it was pretty straightforward. *Sygate*'s install was the fastest of all, and only took a few seconds to complete. The user was only shown about 3 screens total and it was complete. Once installed, the firewalls were tested to see how easy it was to configure common programs with each. *ZoneAlarm* asked the user upon startup if they wished *ZoneAlarm* to preconfigure access permissions for common programs such as Internet Explorer, Firefox, etc. *Sygate* was mostly silent, until the user opened a program such as Internet Explorer, which then produced a small popup window which asked if the user desired to grant Internet access to the program or not and whether or not to remember the choice.

After the initial configuration was completed and common programs added to the firewalls, additional new programs were tested to see how the firewalls would react. When a new program attempted to access the network, *ZoneAlarm* responded with the following popup.

This asked the user whether or not to allow the program Internet Explorer, and offered a checkbox to remember the choice, and never ask the user again unless the program is changed. *Sygate* produced a similar popup when a new program attempted to access the network. It is very similar to *ZoneAlarm*, in that it asks the user whether or not to allow the program access to the network and provides a 'remember my answer' feature, but it also offers further information via the `Detail>>` button. Both programs were comparable to this

Figure 1. *ZoneAlarm alert*

point, except *Sygate* provides more detail to the user if desired. Next, each program's GUI (*Graphical User Interface*) was compared, to evaluate the ease of use, depth of information, etc. *ZoneAlarm*'s window contained easy to use tabs and buttons as shown below.

Firewall	Ease of Use	Security Level	Log Functionality	Best For Novice Users	Best For Advanced Users	Overhead Efficiency
ZoneAlarm	6	7	7	8	5	5
Sygate	8	8	9	5	8	8

Each program is listed with access rights for each function beside it. The access column lets users choose whether or not a program can access either the trusted or Internet networks, and the server column lets users choose whether or not a program can act as a server on the trusted or Internet networks. A blue question mark shows that *ZoneAlarm* will ask the user to allow or deny the program access the next time it attempts to connect to the network. A green check mark shows that the program will always be allowed, and the red 'X' means that the program will always be denied access. *Sygate*'s GUI was also reviewed, and was similar to the following.

Like *ZoneAlarm*, each program is listed, but *Sygate* offers a more in-depth view. It provides an overview of what programs are active (above svchost, firefox, and others are running), and also shows the detailed permissions of each program in the smaller window. In addition, the traffic history graph shows the history of network traffic for easy troubleshooting.

So far, the two programs are comparable, but the most important feature of a firewall is security. Testing *ZoneAlarm* with an executable which *called* another executable only prompted one alert, that of the original executable. However, *Sygate* is much more picky, and produced a message

warning the user that an *Application hijacking* was detected. This shows that the firewall monitors programs more closely than *ZoneAlarm* in this regard. *Sygate* monitored the first program as well as the program it called. Overall, the firewalls were very similar and both are useful. However, I created the following chart to illustrate which I felt was best in each area, on a scale of 1 to 10:

The above numbers are arbitrary. However, I felt they reflected the performance of each firewall. For example, the security level of any firewall cannot ever be perfect, so a 10 is virtually unattainable. However, *Sygate* was not vulnerable to most types of attacks used, and monitors programs a little more closely than *ZoneAlarm*, so this earned an 8. Log functionality was excellent in *Sygate*, and if the user located an offending IP used in an attack, they simply needed to right-click the IP address, and select *Backtrace* to begin a lookup on that IP address. *ZoneAlarm* offers logging as well, but it is not laid out as smoothly as *Sygate*. For novice users, *ZoneAlarm* received an 8, as it requires little configuration once installed. It preconfigures most common programs, and is not too detailed for most beginning users. *Sygate* was best for advanced users, as it has many detailed screens, and users can easily get confused by all the data presented. Overall, I recommend both for home use, but for novice users, *ZoneAlarm* may be the best, and for advanced users, *Sygate* may be the best choice. Overhead efficiency is basically a comparison of how large the executable was for each firewall. *ZoneAlarm's* main executable, vsmon.exe, took up approximately 17,332K of memory, while sygate's main executable, smc.exe, took only 6,560K of memory. Other firewalls can be a little larger than *ZoneAlarm's*, but most are becoming more efficient due to user complaints. This may provide a noticeable difference to those using the firewalls on gaming machines, etc. Of course, this review is only basic and does not take into account many other factors. It was only conducted on Windows, so there are many other solutions for Linux. While some firewalls perform better than others, it is recommended that all systems have some type of firewall. Not using one is only asking for an attack. I trust this short guide helps someone in choosing the best free firewall for their application. ●

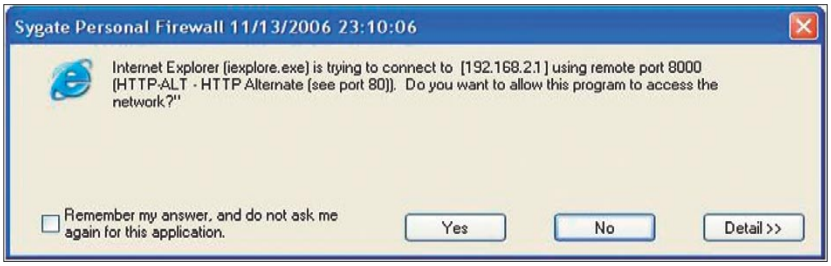


Figure 2. *Sygate Personal Firewall*

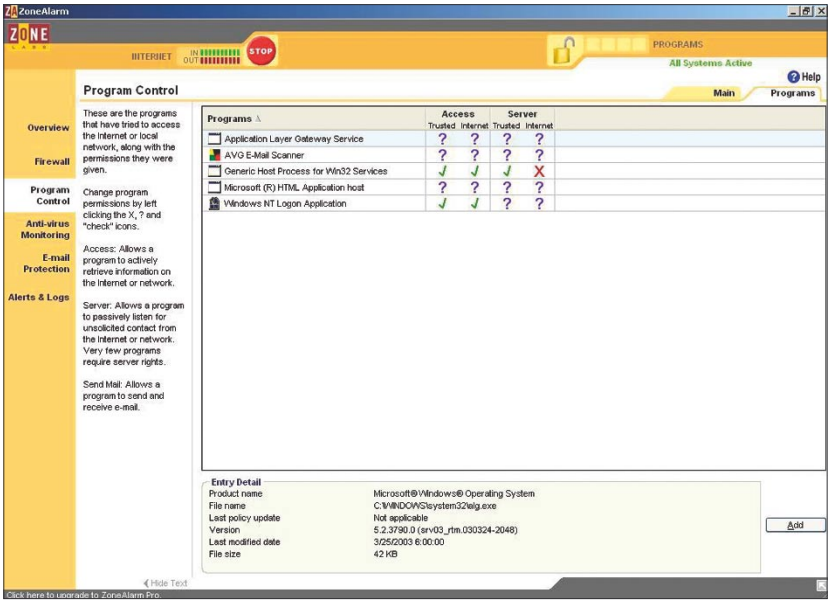


Figure 3. *ZoneAlarm's program control*

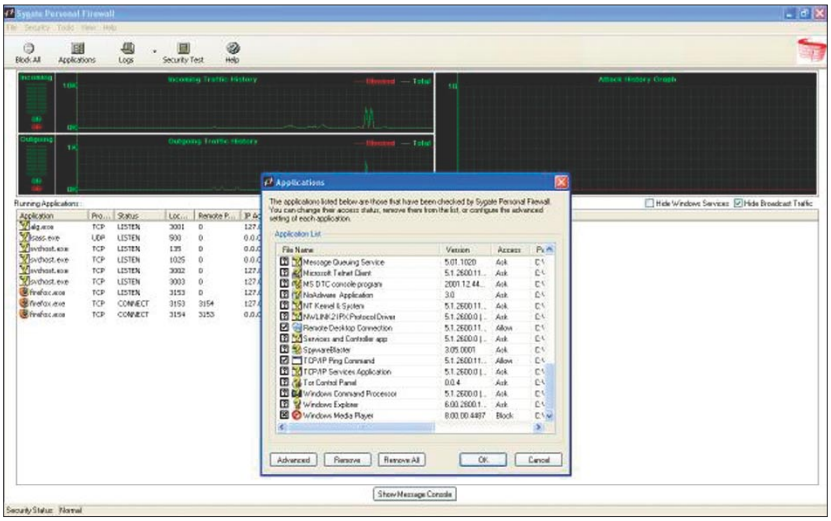


Figure 4. *Sygate Personal Firewall – applications*

About the Author

Josh Sawyer has over 10 years experience with computers and electronic systems and holds numerous certifications, including the MCSE MCSA MCP Network+ Security+ A+, CCNA. He is currently a graduate student at East Carolina University.

CLUB .PRO

Zero Day Consulting

ZDC specializes in penetration testing, hacking, and forensics for medium to large organizations. We pride ourselves in providing comprehensive reporting and mitigation to assist in meeting the toughest of compliance and regulatory standards.

bcausey@zerodayconsulting.com

Digital Armaments

The corporate goal of Digital Armaments is Defense in Information Security. Digital armaments believes in information sharing and is leader in the 0day market. Digital Armaments provides a package of unique Intelligence service, including the possibility to get exclusive access to specific vulnerabilities.

www.digitalarmaments.com

Eltima Software

Eltima Software is a software Development Company, specializing primarily in serial communication, security and flash software. We develop solutions for serial and virtual communication, implementing both into our software. Among our other products are monitoring solutions, system utilities, Java tools and software for mobile phones.

web address: http://www.eltima.com
e-mail: info@eltima.com

First Base Technologies

We have provided pragmatic, vendor-neutral information security testing services since 1989. We understand every element of networks - hardware, software and protocols - and combine ethical hacking techniques with vulnerability scanning and ISO 27001 to give you a truly comprehensive review of business risks.

www.firstbase.co.uk

@ Mediaservice.net

@ Mediaservice.net is a European vendor-neutral company for IT Security Testing. Founded in 1997, through our internal Tiger Team we offer security services (Proactive Security, ISECOM Security Training Authority for the OSSTMM methodology), supplying an extremely rare professional security consulting approach.

e-mail: info@mediaservice.net

@ PSS Srl

@ PSS is a consulting company focused on Computer Forensics: classic IT assets (servers, workstations) up to the latest smartphones analysis. Andrea Ghirardini, founder, has been the first CISSP in his country, author of many C.F. publications, owning a deep C.F. cases background, both for LEAs and the private sector.

e-mail: info@pss.net

If you want to become our partner – join our CLUB .PRO!
To find out more, e-mail us at

en@hakin9.org

CLUB .PRO

Matthew Jonkman

Making Firewalls Smarter

We've all had firewalls for a good long while now. They're a must, a necessity, they can save lives. Well, maybe not save lives, but they can save jobs, OUR jobs. In the most basic terms, a firewall allows certain types of traffic to certain places. The firewall doesn't care what's in that traffic; just that it's on the correct port.

In the not too distant past that was fine. We trusted that everyone surfing the company website was just looking for the number of the regional salesguy so they could spend money. And all was well in the world. But things changed, PHP, CGI, Perl and others allowed hundreds of web applications to creep into our websites. Online banking, online ordering, shopping, online employee directories, the list goes on. Things are much more complex, and getting more so exponentially. We have data we only want some people to see, and application that may or may not contain vulnerabilities. It's nearly impossible to be sure things are bulletproof. Add to that the barrage of vulnerabilities that even those of us that do security for a living struggle to keep pace with.

So our firewalls are still there, letting everybody talk to the webserver, as long as they use the expected port and protocol. In the meantime a myriad of other tools came and matured. Things like Intrusion Detection made it suddenly possible to see if what those people browsing our website were REALLY after, and low and behold, it wasn't always the salesguy's phone number! They're checking my code for vulnerabilities in that handy dandy online ordering system someone wrote 3 years ago! That's nice of them. I wonder if they'll let me know if they find any holes? Hmmm… did I turn off *register_globals* in php?

We've been doing a lot of things to make our firewalls smarter. And of course Intrusion Detection has done great things to move us toward that. There are a number of commercial products out there that will go inline and stop the bad stuff before it gets to the firewall. Even a few open source projects (*Snort_Inline*, etc). But these generally operate on an island, that is they block individual attacks in individual sessions, and only on the entry point at which they sit. The attacker is free to try again to see if they get stopped a second time. If their session is dropped, oh well. Try again and fragment this time, or just move to another portion of the company perimeter and try on a new host.

I recommend something a little smarter for the firewall that provides a real consequence to the attacker, and something geared more toward your entire organization. Distributed Blocking in conjunction with Snort and other data feeds is the way of the future. Gauging the intent of a host asking for information from your organization is a real possibility. If they've not shown any hostile intent, then we let them keep talking to us. If they show hostile intent we block them, all traffic at all perimeter entrance points. Plain andw simple, black and white. There are a number of advantages to this approach.

The vast majority of the crud that's bouncing off of our perimeters and polluting our weblogs and firewall drop logs are automated scans. These are scripts usually run from a compromised host that are set to scour a large IP block for a few known vulnerabilities, record that and save it for the attacker who'll come back later to exploit manually. Or worms hoping to find the IIS or MS exploit of the day exposed for an automated infection. Often they'll try 5-10 URLs or netbios exploits against every IP sequentially in your public IP space.

While if you're even reasonably security aware, you're likely not going to have any of these vulnerabilities, not exposed at least. So most of us would brush this off to *Who cares? Let them knock, they're not getting in.* Here's where the philosophy of distributed blocking comes into play. If when we see that first exploit against the first IP in your network we block that host, the attacker doesn't know you exist. And more importantly, they don't know if you do somehow have one of the vulnerabilities of the day exposed.

So, by the fact that traffic from a certain IP was clearly attack traffic, we can assume that anything else from that IP will very likely be hostile. Even if not hostile, we'll certainly want to talk to that individual about why they're trying to attack us before we'd let them in to browse the company employee directory. We can judge their intent as hostile, and thus we want nothing more to do with that IP. That little bit of information is valuable. We know that the system at IP `xx.xx.xx.xx` is hostile. Why should we let it talk to other

About the Author

Please share your thoughts. You can email Matt directly at *jonkman@bleedingthreats.net*. You can hop into the forums on *http://www.bleedingthreats.net* and start a topic. Or use the Bleeding-sigs email list also available on the website to start a discussion.

systems at other branches or locations of our company? We already know its intent is not good, so let's stop it everywhere. Blammo, Distributed Blocking. But how do we get that bit of information shared amongst other devices, especially when we don't have all the same type of device, and we're not managing them all at one spot?

A tool like Snortsam is the answer to implement distributed blocking for free. Snortsam works via a hub where your input systems send blocks to. Take a snort IDS sensor for example. It saw that IIS attack and judged it hostile. You've defined that if someone trips that signature it's reliable enough that I consider it hostile traffic. So you define that I want to block traffic from any host that tries it. Your Snort (if compiled with the snortsam output plugin) can pass that information to your Snortsam hub running on a system inside your network. It according to the rules you write distributes that bit of information to every perimeter access device on your network within milliseconds. Instantly your entire organization goes dark to that attacker.

That's the basics, you operate with a central hub, or pair of hubs for redundancy. Sensing devices pass block information to the hub, the hub distributes that information to blocking devices such as your firewalls and routers. The hub maintains state and whitelists.

There are complexities of course. Automated blocking is not something to take on lightly or without controls. For one, we need to have something in place to make sure if we do have an accidental block that it's not going to be there for long. We have timing information controlled by the central hub to mitigate this risk. When you setup your rulesets on the snort sensor for example, you cay that if this signature is tripped block the source IP for 1 hour. You go longer for signatures that are absolutely foolproof, and shorter for the ones that might be in a gray area, or can false positive on occasion. That way, if you block a real client accidentally, but you only define that block as 5 minutes, then the inconvenience is minimized. You have to take care to block only on very reliable signatures of course, and those that are not prone to source spoofing (udp, icmp, etc).

Snortsam maintains a state table on the hub. When a block is started it notes the time. When that IPs block has run it's time period, and there haven't been any subsequent blocks since to extend that time, Snortsam tells each device to remove the block. You can choose to have a block last just minutes, years, or anything in between.

Next, we need a whitelist. There are a number of partners, locations, networks, and IPs that you always want to be able to communicate with, even if they do throw a few viruses or exploits your way. You'd still know about it, as your sensing devices will register alerts. But when that block is passed to the snortsam hub it'll compare to it's whitelist, and if the host is on there silently discard that block and not distribute to other devices.

So we have timing information to remove a block in a reasonable time, a whitelist to keep us from blocking required traffic, and a hub to distribute blocks to all of our perimeter devices. What else can we do with this infrastructure? Two very cool things, among many others. We can block IPs before they strike, and we can share blocks with other trusted organizations.

Identifying hostile IPs before they strike us is not as difficult as it sounds. There are many data sources that are trustworthy and can tell us about the intent of certain IPs or IP blocks. Spamhaus runs the DROP list of known professional spammers. Dshield runs a Top 10 Attackers list updated daily. Shadowserver runs a list of known Command and Control Servers for Botnets. If you trust those sources then you can reliably say you don't want to talk to those IPs. So a simple bit of perl to once a day download each of these lists and feed them to your snortsam hub with a 24 hour expiration will give you a huge advantage. Your network will always look like it doesn't exist to these known hostile hosts.

So we've got this very nice list of blocked IPs and a steady stream of blocks and removes flowing. If we're learning these things, why not share with my buddy? If you have a sister organization, trusted friend, or even just a neighbor that is of a similar security mind as yourself, you can link your snortsam hubs and share blocks.

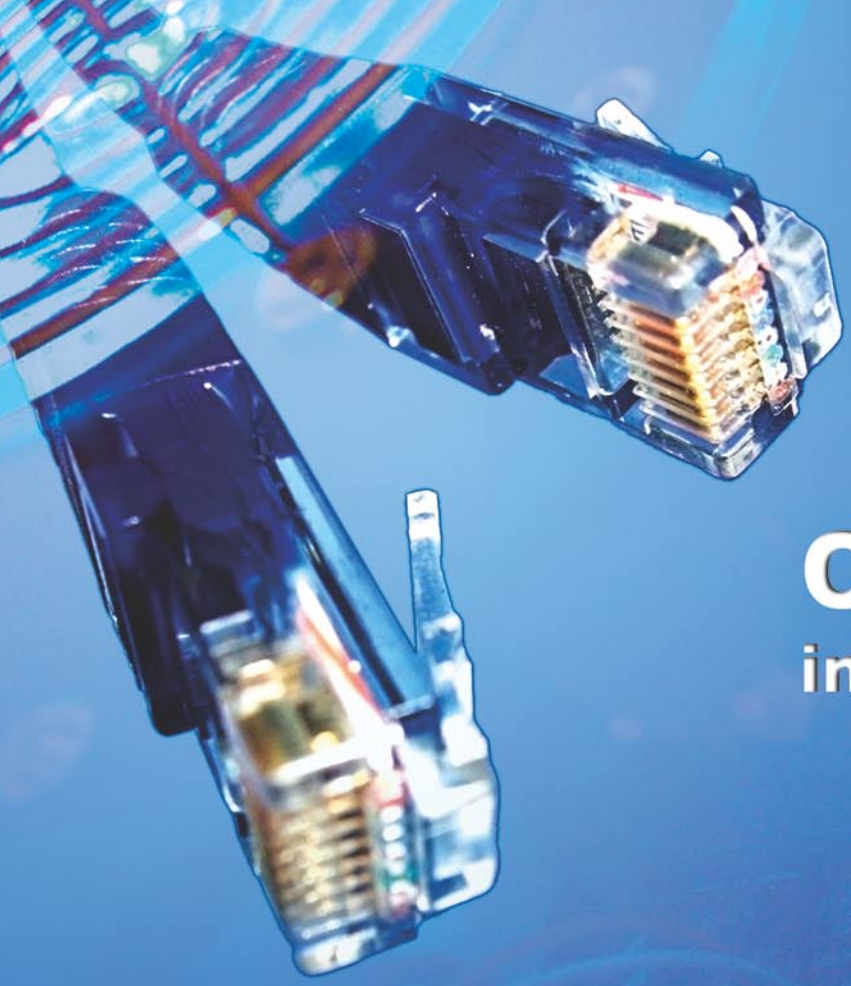
This is safer than you might think. You can filter the blocks you accept from a sister hub, down the snort SID or reason for the block, and even adjust the time you prefer to block in that case. So even if you differ in philosophy in what's a reliable signature or block reason, you can still share hostile intent. And above all your local whitelist still overrides all.

Entrapment is another cool thing you can do with distributed blocking. This is one of my favorites. If you own some IP space and have a spare IP or more, setup snort signatures to trip on any traffic to any unused IP space. Anyone that has a real legitimate non-hostile reason to communicate with you should never hit that dead IP, or the unused IPs in your block. If they do, they're scanning, looking for vulnerabilities, or just plain screwing around looking to learn things they don't need to know. Block 'em!

One step further, if you're into honeypots, setup a Nepenthes sensor in that unused space. You can feed the logs to a Snortsam hub, or just let your IDS trip on the attacks to it and block everyone that falls for the bait. Quick and easy, black and white. If you're exploiting my honeypot, your intent is hostile.

If you like this concept there are a number of ways to execute, and some very good HowTos out there. How you implement, what you use, and how you build it is very dependant on your architecture and security posture. You can find what you need by visiting the Snortsam website, *http://www.snortsam.net* or Bleedingthreats.net. Snortsam has plugins to push blocks and removes to every major firewall out there including Checkpoint, PIX, ISA, many routers, every major open source firewall (IPF, PF, IPTables, etc) and even some smart switches via snmp to down an interface. It's tried and tested, mature, reliable.

I can tell you from years of this being implemented in a managed security environment sharing blocks across hundreds of sensors and firewalls that it works. You have to consider a lot of things when you implement, but once it's up and running it'll make an incredible difference in the state of your security. ●



hakin9

Coming up
in the next issue...

The main subject of the next issue of *hakin9* will be:

- ✓ Excellent writing on Malware within the .NET-framework
- ✓ Continuation of Oracle security topic

Also inside:

- ✓ Free CD with useful applications and tools
- ✓ Advanced technical articles directed to the IT security specialists
- ✓ Presentation of most popular security tools
- ✓ Interesting techniques of protecting and attacking computer systems

hakin9 is a bi-monthly. It means 6 issues of *hakin9* a year! Each one full of precious guidelines, useful hints and essential information necessary to be even more efficient IT security professional.

Next issue of *hakin9* available in July!

LINUX BAZAR

SUPPLIES (OSS) OPEN SOURCE SOFTWARE CDs/DVDs & BOOKS.

Games
Utilities
Applications
Database
Training Videos
Multimedia
Linux Distributions
Website Hosting
Linux Technical Support
Custom Downloads
OSS Gift Items / Memorablia
Freeware / Shareware
Windows - Development Tools
Technical Books & Magazines
eBay2eStore - eBay Listings to eStore

Pay in currency of your choice.

We accept Internet Banking and all Credit Cards.

We ship worldwide.

LinuxBazar

sales@linuxbazar.com www.linuxbazar.com

The Official Event of the Ruby on Rails Community

The RailsConf Experience

- An entire conference dedicated to Ruby on Rails
- A program designed around all levels of Rails expertise
- The most innovative and sucessful Rails experts and companies showcased onstage
- A forum for a wide variety of business people seeking to collaborate and solve similar problems
- A gathering place for the worldwide Rails community, including an important network of experts, alpha geeks and innovators
- Hands-on and in-depth guides for learning how to best employ rails in a variety of situations
- Ample opportunity for all participants to connect, contribute and collaborate throughout the event
- Hear the latest updates, get information and meet the core Rails development team

RAILSCONF

MAY 17-20, 2007
PORTLAND, OREGON, USA
conferences.oreilly.com/rails

Use code RCØ7LNX+ and save 10% on registration fees

RAILSCONF EUROPE

17-19 SEPTEMBER 2007
BERLIN, GERMANY
conferences.oreilly.com/railseurope

Registration opens soon.

If you're using Rails, make plans now to attend RailsConf 2007

Co-presented by by O'Reilly Media, Inc. and Ruby Central, Inc.