**Table of Contents**

Preface

Requirements

Philosophy

OpenBSD

Pre Installation Tasks

Disk Partitioning

Base Installation

Post Installation Tasks

Patching

Rebuilding the Kernel

Install Packages

Install LibreSSL

Network Configuration

rc.conf.local

rc.shutdown

rc.securelevel

OpenSSH

Mail Server

Database Postgresql

nginx + Phusion Passenger

ircd-hybrid

AIDE

Packet Filtering

KDE 4

GNOME 3

Nessus

Snort

Yara

Utilities

Hacking Tools

Pre Production Settings

Commands Reference

About the Author

Preface

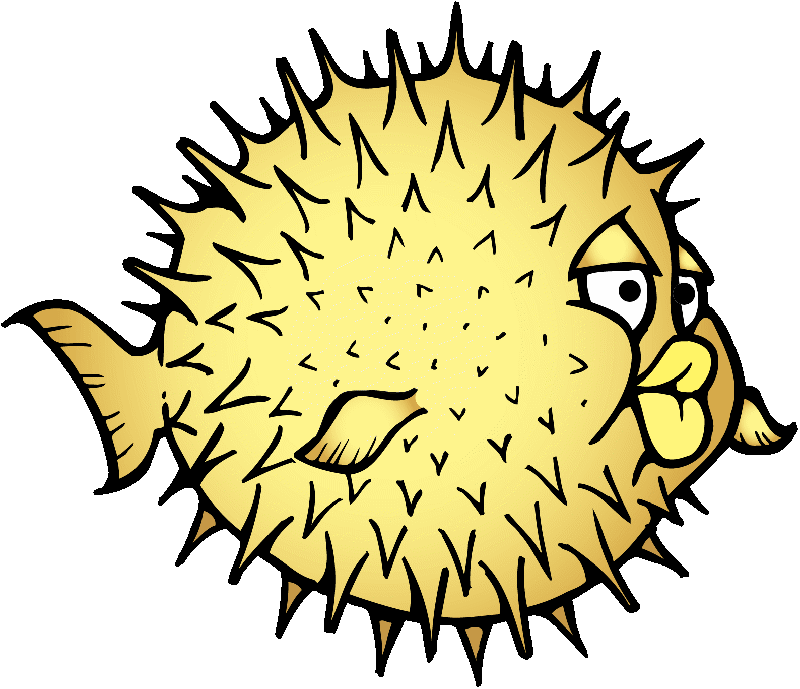
This book is intended for both advanced linux users and security professionals who want to build a super secure server for any online service. OpenBSD can be used also as a firewall itself, webserver, DNS or anything you want to keep secure and online.

Theo de Raadt is a software engineer who lives in Calgary, Canada. He is the founder of the OpenBSD project in 1994. In 2010, there were Reports of FBI back door in OpenBSD community. It all came to a head, quickly. All of the "potentially contaminated" commits were reviewed, some code was replaced, everything was re-verified, and Theo maintains that none of the "back doors" implanted were exploitable unless the user "did something stupid".

The notion of "users doing something stupid" is key to the OpenBSD philosophy. The OS comes out of the box with almost nothing listening on open ports. Not quite "black box opaque" but close. In that state, it's safe as houses.

In theory, if you're using OpenBSD, it's because you're a security professional, and it's your weapon of choice, so the odds are that you're not going to make those mistakes, because you're a professional.

Requirements and Philosophy



This class is for people who know already Linux or Unix operating systems, it is intended for advanced sysadmin and security professionals.

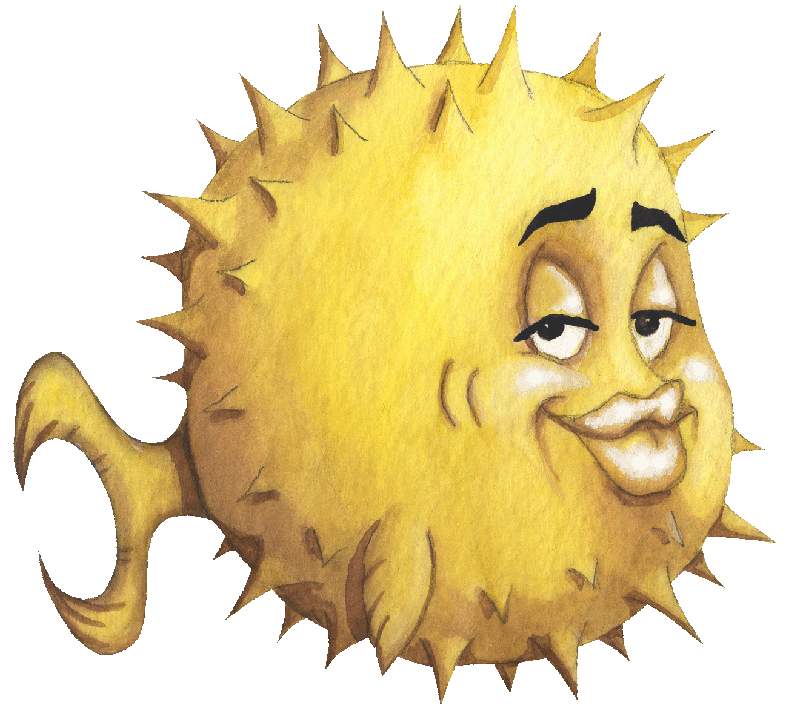
Whaat you need? A desktop, a remote server, or a notebook which is your own property. A Virtual box is good also for the class, but not for staging or production environments.

In his book, and class, I do not provide a VM with OpenBSD on it, because would break the philosophy.

In matter of computer security you don't have to trust not even your master, the VM could be compromised and give the false feeling of security. For this reason you have to build everything with your hands, checking what you do, following the instructions.

The philosophy behind the OpenBSD project is unique and awesome, the only OS focused on Security makes it the facto perfect server for online applications, databases, and communications.

Pre Installation Tasks



Before starting the installation you have to be sure the hardware is seen by the kernel, because adding external firmware, with sources not checked by the OpenBSD team will violate the Proactive Security on which OpenBSD is based on.

Next to booting OpenBSD for the first time (reading and understanding the messages for errors) you must have a hard disk to dedicate to the installation. A partition only will break the philosophy and installing it on a VM is good for making practise but not for a serious setup.

Anyone who has access to the OpenBSD partition, or the host OS of the Virtual Machine could both have access to the resources compromising it easily.

After the boot ends, you will be asked to proceed with the installation or go to the shell. Choose the shell because we need to check the disk first.

Annotate its dimension in term of size, cilinders and blocks. and then give the command:

# fdisk -u /dev/sda

Ipotizing sda is your HDD device and it is new the command will rewrite a new MBR (Master Boot Record) preserving the existing partitions.

If the HDD is not new, you should phisically delete every single byte stored on it, without caring if it's full of data or empty. To do this launch this command and go to take a coffee because it will take a while (depending on the size of the disk)

# dd if=/dev/zer of=/dev/sda

This command will erase everything writing the byte zero in every block, like the device goes to the market when it's new from fabric.

To send the USR1 signal to the *dd* prcoess:

$ kill -USR1 1337

Note that as soon as the USR1 signal is detected, *dd* will print out the current statistics to its STDERR.

$ dd if=/dev/random of=/dev/null bs=1K count=100

0+14 records in

0+14 records out

204 bytes (204 B) copied, 24.92 seconds, 0.0 kB/s

$ watch -n 10 kill -USR1 1337

Disk Partitioning



I am going to comment my current hard disk configuration for a 250GB SSD.

Filesystem Size Used Avail Capacity Mounted on

/dev/sd1a 9.8G 846M 8.5G 9% /

/dev/sd1k 17.2G 629M 15.8G 4% /home

/dev/sd1d 3.9G 2.2M 3.7G 0% /tmp

/dev/sd1f 2.0G 678M 1.2G 35% /usr

/dev/sd1g 1005M 178M 777M 19% /usr/X11R6

/dev/sd1h 19.7G 6.1G 12.6G 33% /usr/local

/dev/sd1j 5.9G 2.0K 5.6G 0% /usr/obj

/dev/sd1i 2.0G 2.0K 1.9G 0% /usr/src

/dev/sd1e 19.7G 42.1M 18.6G 0% /var

OpenBSD installs its components in many differents partitions each one of them has its own purpose and we have to set the correct size and permissions that may vary.

If you are the sysadmin reserve a bigger partition for the root privilege. 1GB is not enough (info based on the automatic partitioning) 10Gb of higher will be good for root operations.

Now the destination use of the host makes the diffence in partitioning. If you have to manage a lot of users, the /home partition should be large enough to allow storing and serve all users. For a database server, which datafiles goes under /var you should reserve a big slice of the disk to it. Another example could regard the /tmp directory, which time ago was due to make it at least 5GB of space to allow a DVD copy, but nowadays there is no need to keep it so large (unless you have to use it as desktop and copy BlueRay disks, for example)

On a live server, put on the internet 24/7 you don't have to install the GUI (gnome/kde/etc) and the /usr/X11R6 partition you see on the automatic partitioning could be deleted without problems. 2/3Gb for the /usr/src is more than enough, but keep in mind that leaving the sources on the server can lead some security problems. An attacker, once suceeded to get privileges on the machine could backdoor a binary file or even the kernel and compromise the host. Cleaning off all the tools and source that can facilitate the operations of an attacker is a part of the host security.

Base Install

**Welcome to the OpenBSD/amd64 6.1 installation program.**

(I)nstall, (U)pgrade, (A)utoinstall or (S)hell?

*I{Enter}*

**Choose your keyboard Layout ('?' or 'L' for list) [default]**

L{Enter}

it{Enter}

**System hostname? (short form, e.g. 'foo')**

base{Enter}

Available network interfaces are: em0 vlan0.

**Which network interface do you wish to configure? (or 'done') [em0]**

{Enter}

**IP adress v4 for em0? (or 'dhcp' or 'none') [dhcp]**

{Enter}

**IPv6 address for em0? (or 'rtsol' or 'none') [none]**

{Enter}

**IPv6 prefix length for em0? [64]**

{Enter}

Available network interfaces are: em0 vlan0

**Which network interface do you wish to configure? (or 'done') [done]**

{Enter}

**Password for root account? (will not echo)**

(rootpassword){Enter}

**Password for root account? (again)**

(rootpassword){Enter}

**Start sshd(8) by default? [yes]**

{Enter}

**Do you expect to run the X Window System? [yes]**

{Enter}

**Do you want the X Window System to be started by xenodm(1)? [no]**

yes{Enter}

**Setup a user? (Enter a lower-case loginname, or 'no')** [no]

d4x{Enter}

**Full name for user d4x? [d4x]**

Davide Gonnella

**Password for user d4x? (will not echo)**

(user password){Enter}

**Password for user d4x? (again)**

(user password){Enter}

WARNING: root is targeted by password guessing attacks, pubkeys are safer.

**Allow root ssh login? (yes, no, prohibit-password)** [no]

{Enter}

**What timezone are you in? ('?' for list)** [Europe/Amsterdam]

{Enter}

**Available disks are: wd0**

Which disk is the root disk? ('?' for details) [wd0]

{Enter}

No valid MBR or GPT.

**Use (W)hole disk MBR, whole disk (G)PT or (E)dit?** [whole]

{Enter}

Before starting the install process you have to collect and prepare some info you will be asked during the process. In detail write down:

***- Keyboard layout***

***- Hostname***

***- Domain***

***- Timezone***

***- IP address***

***- Subnet mask***

***- broadcast address***

***- gateway IP***

***- DNS server IP(s)***

***- root password***

***- user name and password***

Years ago the prcess was complex and not much smooth, but recently the team added a script that let you go through the installation quite easily, also if there is no graphic interface for it.

Sources and Patching

After the installation you may want to apply all the patches for the chosen version and architecture.

Initially you must download the sources and extract them in /usr/src. In our scenario for the 6.2 version, amd64 architecture we will download in /root the archives.

Download the source code

wget <http://ftp.openbsd.org/pub/OpenBSD/6.2/src.tar.gz>

Download system files

wget <http://ftp.openbsd.org/pub/OpenBSD/6.2/sys.tar.gz>

Download Ports

wget <http://ftp.openbsd.org/pub/OpenBSD/6.2/ports.tar.gz>

Download X sources

wget<http://ftp.openbsd.org/pub/OpenBSD/6.2/xenocara.tar.gz>

Download the hash of each file to check integrity

wget <http://ftp.openbsd.org/pub/OpenBSD/6.2/SHA256>

Now check the integrity of the source by hand, open a console (terminal) and the SHA256 file.

Give the sha256 for each file and check the returned hash if corresponds with the one specified in the SHA256.

# sha256 src.tar.gz

SHA256(src.tar.gz) = 732d9eadb07aebb0ca08b2f94bd0695250b5ff7621942b061e9f4dfbb94a3550

If there is match the file integrity check is positive, otherwise you have to download the file again, maybe from another mirror and check it again.

After having checked all the files we can go to the next step which is to extract every archive, the patches corresponding to the version and architecture, and see what there is to do to fix the binaries or the kernel.

The current version 6.2 is quite new at the time and there are not many patches but we will go through both scenarios for a full patching experience.

**# cd /sys/arch/amd64/conf**

**# cp GENERIC CUSTOM**

**# vi CUSTOM**

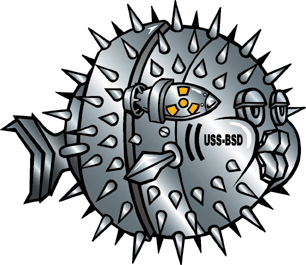
**# make your changes**

**# config CUSTOM**

**# cd ../compile/CUSTOM**

**# make**

Rebuild the Kernel



Install Packages



Choose your nearest OpenBSD mirror from this list:

Export the environment varialble, or even better add it to your .profile file in the home directory.

#export PKG\_PATH=<http://ftp.openbsd.org/pub/OpenBSD/6.2/packages/amd64>



Download the latest version of LibreSSL (currently 2.6.2)

# wget http://ftp.openbsd.org/pub/OpenBSD/LibreSSL/libressl-2.6.2.tar.gz

# wget http://ftp.openbsd.org/pub/OpenBSD/LibreSSL/libressl-2.6.2.tar.gz.asc

Check the integrity of the downloaded archive

Extract the archive contents into a temporary location

# tar -zxvf libressl-2.6.2.tar.gz

Enter the directory

# cd libressl-2.6.2

Configure LibreSSL without omitting to disable asm or you get many errors.

# ./configure --disable-asm

Let LibreSSL check for all its dependancies before building it

# make check

Compile LibreSSL

# make install

Delete the old openssl

# rm /usr/bin/openssl

Replace openssl with libressl creating the following symlink

# ln -sf /usr/local/bin/openssl /usr/bin/openssl

**Network Configuration**

Create a file in etc with interface name as extension for the file

# touch /etc/hostname.em0

inet 185.2.81.27 255.255.255.0 none

# vi /etc/resolv.conf

lookup file bind

nameserver 93.190.136.10

# vi /etc/myname

# vi /etc/mygate

The */etc/rc.d* directory contains shell scripts to start, stop, and reconfigure daemon programs (“services”).

Services installed from [packages(7)](https://man.openbsd.org/packages.7) may be started at boot time in the order specified by the pkg\_scripts variable from [rc.conf(8)](https://man.openbsd.org/rc.conf.8); the order will be reversed during shutdown.

rc.conf.local

edit **/etc/rc.shutdown**

*powerdown=yes*

edit **/etc/rc.securelevel**

sysctl kern.securelevel

edit **/etc/rc.local**

#create temporary folder for nginx

mkdir /tmp/nginx

mkdir /tmp/nginx/client

# Add a boot call for Postgresql

if [ -x /usr/local/bin/pg\_ctl ]; then

su -l \_postgresql -c "nohup /usr/local/bin/pg\_ctl start \

-D /var/postgresql/data -l /var/postgresql/logfile \

-o '-D /var/postgresql/data'"

echo ' postgresql '

fi

OpenSSH

###################################

### d4x337 SSH SERVER /etc/ssh/sshd\_config

#################################################

#

Port 22

Protocol 2

AddressFamily inet

#ListenAddress 127.0.0.1

#See the questions section for setting up the gatekeeper

#ForceCommand /tools/ssh\_gatekeeper.sh

AllowUsers calomel@10.10.10.3 calomel@192.168.\*

AllowGroups calomel

AllowTcpForwarding yes

#AuthorizedKeysFile .ssh/authorized\_keys (need to be be commented for OpenSSH 5.4)

Banner /etc/banner

ChallengeResponseAuthentication no

Ciphers aes256-ctr,aes192-ctr,aes128-ctr

ClientAliveInterval 15

ClientAliveCountMax 3

Compression yes

GatewayPorts no

LogLevel VERBOSE

LoginGraceTime 50s

MACs hmac-sha2-512-96,hmac-sha2-512,hmac-sha2-256-96,hmac-sha2-256,hmac-sha1-96,hmac-sha1

MaxAuthTries 6

MaxStartups 10

PasswordAuthentication yes

PermitEmptyPasswords no

#PermitOpen localhost:80

PermitRootLogin no

PermitUserEnvironment no

PidFile /var/run/sshd.pid

PrintLastLog yes

PrintMotd no

PubkeyAuthentication yes

StrictModes yes

Subsystem sftp /usr/libexec/sftp-server

SyslogFacility AUTH

TCPKeepAlive no

UseDNS no

UseLogin no

UsePrivilegeSeparation yes

X11DisplayOffset 10

X11Forwarding no

X11UseLocalhost yes

#Match User anoncvs

# X11Forwarding no

# AllowTcpForwarding no

# ForceCommand cvs server

#

####################################################

### d4x337 ssh SERVER /etc/ssh/sshd\_config

####################################################

**Build a Mail Server**

The Mail server we are going to build is based on Postgresql, Postfix and other technologies we will install. First we have to start installing a library for Postfix since the standard one is not suitable for what we have to do.

# pkg\_add -iv courier-authlib

and we choose the one that support both Postgresql and Sasl2 (option 16)

then we can install the other packages:

#pkg\_add -iv courier-imap courier-pop3 postfix

Also for Postfix we have to choose the most recent one that support bot Postgresql and Sasl2

# /usr/local/sbin/mkpop3dcert

[ ... ]

# /etc/rc.d/courier\_pop3\_ssl start

courier\_pop3\_ssl(ok)

# /usr/local/sbin/mkimapdcert

# /etc/rc.d/courier\_authdaemond start

courier\_authdaemond(ok)

# /etc/rc.d/courier\_imap\_ssl start

courier\_imap\_ssl(ok)

edit authdaemonrc and leave only the following module

**authmodulelist="authpgsql"**

#touch /etc/authdaemond

PGSQL\_HOST 127.0.0.1

PGSQL\_PORT 5432

PGSQL\_USERNAME vmail

PGSQL\_PASSWORD vmail

PGSQL\_DATABASE tortuga

PGSQL\_CHARACTER\_SET UTF8

PGSQL\_USER\_TABLE mail\_accounts

PGSQL\_CRYPT\_PWFIELD crypt

PGSQL\_CLEAR\_PWFIELD password

DEFAULT\_DOMAIN d4x337.com

PGSQL\_UID\_FIELD uid

PGSQL\_GID\_FIELD gid

PGSQL\_LOGIN\_FIELD login

PGSQL\_HOME\_FIELD home

PGSQL\_NAME\_FIELD name

PGSQL\_MAILDIR\_FIELD maildir

PGSQL\_QUOTA\_FIELD quota

#PGSQL\_SELECT\_CLAUSE select address='%u'

edit /etc/postfix/clamav-milter

MilterSocket /var/spool/postfix/var/run/milter-clamav/clamav.sock

User \_postfix

ClamdSocket tcp:127.0.0.1:3310

LogSyslog yes

LogFacility LOG\_MAIL

**/etc/postfix/pgsql\_virtual\_domains.cf**

user = vmail

password = vmail

dbname = mail

hosts = 127.0.0.1

query = SELECT domain FROM domains WHERE domain='%s'

**/etc/postfix/pgsql\_virtual\_alias\_maps.cf**

user = vmail

password = vmail

dbname = mail

hosts = 127.0.0.1

query = SELECT alias FROM alias\_maps WHERE account='%s'

**/etc/postfix/pgsql\_virtual\_mailboxes.cf**

user = vmail

password = vmail

dbname = mail

hosts = 127.0.0.1

query = SELECT maildir FROM users WHERE login='%s'

## config file for freshclam

DatabaseDirectory /var/db/clamav

DatabaseOwner \_clamav

DNSDatabaseInfo current.cvd.clamav.net

DatabaseMirror db.it.clamav.net

DatabaseMirror database.clamav.net

MaxAttempts 3

checks 24

After editing freshclam file try to update virus definitions by hand with:

# freshclam

If everything goes fine we can it to crontab calling the update every hour.

#crontab -e –u root

#Every hour at 26 min - Update Antivirus definitions

14 \* \* \* \* /usr/local/bin/freshclam >/dev/null 2>&1

**Postgresql as Database**

Install Postgresql adding its packages

**# pkg\_add -iv postgresql\_server**

Use the user \_postgresql to initizialize the DB that usually goes in /var/postgresql/data

**# initdb -D /var/postgresql/data**

**# psql template1**

Change password for user postgresql

Create a databases

Add Postgresql to startup scripts

Now we can add a script I have created to backup the postgresql database

I usually put these scripts under a folder in /root called jobs, so:

# mkdir /root/jobs

# touch /root/jobs/postgres\_backup.sh

Add this content to the just created file

#!/bin/bash

DATABASE\_NAME=$1

USERNAME=$2

HOST=$3

WORKING\_DIR=$4

RUNNING\_MODE=$5

PG\_DUMP=/usr/local/bin/pg\_dump

if [ "$RUNNING\_MODE" = "--dated" ]

then

DATA=`date +'%h\_%d\_%y'`

WORKING\_DIR="${WORKING\_DIR}"/"${DATABASE\_NAME}"

OUTFILE="${DATABASE\_NAME}\_${DATA}.sql"

else

DATA=`date +'%a'`

if [ "$RUNNING\_MODE" = "--isolated" ]

then

WORKING\_DIR="${WORKING\_DIR}"/"${DATABASE\_NAME}/${DATA}"

OUTFILE="${DATABASE\_NAME}.sql"

else

# data = giorno della settimana

WORKING\_DIR="${WORKING\_DIR}"/"${DATA}"

OUTFILE="${DATABASE\_NAME}.sql"

fi

fi

# Creo la directory se non esiste

if ! test -d "$WORKING\_DIR"

then

mkdir "$WORKING\_DIR" > /dev/null 2>&1

if [ $? -ne 0 ]

then

echo "Errore nella creazione della directory $WORKING\_DIR"

exit

fi

fi

# entro nella directory di lavoro

cd "$WORKING\_DIR"

START\_TIME=`date` # istante di inizio

# esecuzione del backup effettivo

$PG\_DUMP --create --column-inserts -f $OUTFILE --encoding=UTF-8 -U $USERNAME -h $HOST $DATABASE\_NAME

END\_TIME=`date`

#crontab -e -u root

#Every day at 7.00 am - database vmail dump backup

0 7 \* \* \* /bin/sh

/root/jobs/postgres\_backup.sh vmail \_postgresql 127.0.0.1 /root/vmail\_backup –dated

it will run every day in the morning at 7am and will create a compressed file named with the date house of the creation process in the root directorty (or other of your choice)

**nginx + Phusion Passenger**

# $OpenBSD: nginx.conf

#user www;

worker\_processes 8;

worker\_priority 15;

error\_log /var/log/nginx/error.log;

pid /var/run/nginx.pid;

worker\_rlimit\_nofile 1024;

events {

worker\_connections 1024;

}

http {

client\_body\_timeout 5s; # maximum time between packets the client can pause when sending nginx any data

client\_header\_timeout 5s; # maximum time the client has to send the entire header to nginx

keepalive\_timeout 75s; # timeout which a single keep-alive client connection will stay open

send\_timeout 15s; # maximum time between packets nginx is allowed to pause when sending the client data

client\_max\_body\_size 10M;

access\_log /var/log/nginx/access.log;

include /etc/nginx/mime.types;

default\_type application/octet-stream;

log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

'$status $body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" "$http\_x\_forwarded\_for"';

passenger\_root /usr/local/lib/ruby/gems/2.2/gems/passenger-5.1.1;

passenger\_ruby /usr/local/bin/ruby22;

passenger\_default\_user www;

passenger\_default\_group www;

ssl\_ciphers ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA:ECDHE-RSA-AES256-SHA;

ssl\_ecdh\_curve secp384r1; # 384 bit prime modulus curve efficiently supports ECDHE ssl\_ciphers up to a SHA384 hash

ssl\_prefer\_server\_ciphers on; # the preferred ciphers are listed on the server by "ssl\_ciphers"

#ssl\_protocols TLSv1.2 TLSv1.1 TLSv1; # protocols, the order is unimportant

ssl\_protocols TLSv1.2; # protocols, the order is unimportant

ssl\_session\_cache shared:SSL:10m;

ssl\_session\_timeout 128s; # how long before the client and server must renegotiate the ssl key

ssl\_stapling on; # staple the ssl cert to the initial reply returned to the client for speed

ssl\_stapling\_verify on;

server {

listen 80;

server\_name d4x337.com www.d4x337.com;

root /var/www/apps/d4x337/public;

}

server {

listen 80;

server\_name tortuga.tech www.tortuga.tech;

return 301 https://www.tortuga.tech$request\_uri;

}

server {

listen 80;

server\_name tigetfoods.online www.tigerfoods.online;

root /var/www/apps/d4xshop/public;

passenger\_enabled on;

location ^~ /assets/ {

gzip\_static on;

expires max;

add\_header Cache-Control public;

}

}

server {

listen 443;

ssl on;

# ssl\_certificate /etc/nginx/ssl/www\_tortuga\_tech.crt;

ssl\_certificate /etc/nginx/ssl/ssl-bundle.crt;

ssl\_certificate\_key /etc/nginx/ssl/www.tortuga.tech.key;

server\_name tortuga.tech;

passenger\_enabled on;

# passenger\_app\_env=production;

# passenger\_user www;

# passenger\_group www;

server\_name tortuga.tech www.tortuga.tech;

root /var/www/apps/tortuga00/public;

location ^~ /assets/ {

gzip\_static on;

expires max;

add\_header Cache-Control public;

}

error\_page 500 502 503 504 /500.html;

keepalive\_timeout 10;

}

include /etc/nginx/conf.d/\*.conf;

}

**ircd-hybrid**

**Packet Filtering**



Packet Filtering on OpenBSD deserve its own book, due to its capabilities and the varierty of configurations

## OpenBSD pf.conf by d4x337 ####################################

# Required order: options, queueing, translation and filtering.

################ Macros #######################################################

### Interfaces ###

ExtIf ="bge0"

### Queues, States and Types ###

IcmpType ="icmp-type 8 code 0"

IcmpMTUd ="icmp-type 3 code 4"

# SshQueue ="(ssh\_bulk, ssh\_login)"

#SynState ="flags S/SA synproxy state"

TcpState ="flags S/SA modulate state"

UdpState ="keep state"

### Ports ###

# FtpPort ="8021"

SshPort ="41337"

### Stateful Tracking Options (STO) ###

FtpSTO ="(tcp.established 7200)"

ExtIfSTO ="(max 9000, source-track rule, max-src-conn 2000, max-src-nodes 14)"

IntIfSTO ="(max 150, source-track rule, max-src-conn 50, max-src-nodes 14, max-src-conn-rate 75/20)"

SmtpSTO ="(max 200, source-track rule, max-src-states 50, max-src-nodes 50, max-src-conn-rate 30/10, overload <BLOCKTEMP> flush global)"

SshSTO ="(max 5, source-track rule, max-src-states 5, max-src-nodes 5, max-src-conn-rate 5/60)"

WebSTO ="(max 500, source-track rule, max-src-states 50, max-src-nodes 75, max-src-conn-rate 120/100, overload <BLOCKTEMP> flush global)"

### Tables ###

table <BLOCKTEMP> counters

table <BLOCKPERM> counters file "/etc/pf\_permban"

table <spamd-white>

################ Options ######################################################

### Misc Options

set skip on lo

# set debug urgent

set reassemble yes

set block-policy drop

set loginterface $ExtIf

set state-policy if-bound

set fingerprints "/etc/pf.os"

set ruleset-optimization none

### Timeout Options for normal operations

set optimization normal

set timeout { tcp.established 600, tcp.closing 60 }

### Timeout Options for anti SYN DDoS with short timeouts and increased states

set optimization aggressive

set timeout { adaptive.end 120000, interval 2, tcp.tsdiff 5, tcp.first 5, tcp.closing 5, tcp.closed 5, tcp.finwait 5, tcp.established 4200}

set limit { states 100000, src-nodes 100000 }

################ Queueing ####################################################

queue wan on bge0 bandwidth 100M

queue ack parent wan bandwidth 30M

queue dns parent wan bandwidth 1M

queue ssh parent wan bandwidth 30M

queue http parent wan bandwidth 38M default

queue mail parent wan bandwidth 10M

queue spamd parent wan bandwidth 1M

################ Translation and Filtering ###################################

# Blocking spoofed packets: enable "set state-policy if-bound" above

block drop in quick on !$ExtIf inet from (bge0:network) to any

# Block to/from illegal sources/destinations

block drop in quick on $ExtIf inet proto tcp from <BLOCKTEMP> to any

block drop in quick on $ExtIf inet proto tcp from <BLOCKPERM> to any

block drop in quick on $ExtIf inet proto udp from <BLOCKTEMP> to any

block drop in quick on $ExtIf inet proto udp from <BLOCKPERM> to any

block drop in quick on $ExtIf inet from any to 255.255.255.255

block drop in log quick on $ExtIf inet from urpf-failed to any

block drop in log quick on $ExtIf inet from no-route to any

# BLOCK all in on external interface by default and log

# block log on $ExtIf

# Packet normalization ( "scrubbing" )

# remove "min-ttl 64" if you need native traceroute functions or just use "traceroute -I" instead

# match log on $ExtIf all scrub (random-id min-ttl 64 set-tos reliability reassemble tcp max-mss 1440)

# $ExtIf inbound

pass in log on $ExtIf inet proto tcp from !($ExtIf) to ($ExtIf) port https $TcpState $WebSTO queue (http, ack) rdr-to lo0

pass in log on $ExtIf inet proto tcp from !($ExtIf) to ($ExtIf) port www $TcpState $WebSTO queue (http, ack) rdr-to lo0

pass in log on $ExtIf inet proto tcp from <spamd-white> to ($ExtIf) port smtp $TcpState $SmtpSTO queue (mail, ack) rdr-to lo0

pass in log on $ExtIf inet proto tcp from !<spamd-white> to ($ExtIf) port smtp $TcpState $SmtpSTO queue (spamd) rdr-to lo0 port spamd

# pass in log on $ExtIf inet proto tcp from !($ExtIf) to ($ExtIf) port ssh $TcpState $SshSTO queue $SshQueue rdr-to lo0 port $SshPort

#pass in log on $ExtIf inet proto icmp from !($ExtIf) to ($ExtIf) $IcmpType $UdpState

#pass in log on $ExtIf inet proto icmp from !($ExtIf) to ($ExtIf) $IcmpMTUd $UdpState

# $ExtIf outbound

pass out log on $ExtIf inet proto tcp from ($ExtIf) to !($ExtIf) $TcpState $ExtIfSTO queue (ack) tagged EGRESS

# pass out log on $ExtIf inet proto tcp from ($ExtIf) to !($ExtIf) port ssh $TcpState $ExtIfSTO queue $SshQueue tagged EGRESS

pass out log on $ExtIf inet proto tcp from ($ExtIf) to !($ExtIf) port ftp $TcpState $FtpSTO queue (ack) tagged EGRESS

pass out log on $ExtIf inet proto udp from ($ExtIf) to !($ExtIf) $UdpState $ExtIfSTO queue (ack) tagged EGRESS

pass out log on $ExtIf inet proto udp from ($ExtIf) to !($ExtIf) port domain $UdpState $ExtIfSTO queue (dns) tagged EGRESS

pass out log on $ExtIf inet proto icmp from ($ExtIf) to !($ExtIf) $UdpState $ExtIfSTO queue (http) tagged EGRESS

pass

############## END of pf.conf by d4x337 ################

**KDE4 for Desktop use**

Installing KDE4 is easy because it will download automatically all dependencies:

# pkg\_add -iv kde4

startkde4

**Gnome for Desktop use**

Edit /etc/rc.conf.local and enable/disable these startup scripts

xdm\_flags=NO

gdm\_enable=YES

gnome\_enable=YES

multicast\_host=YES

pkg\_scripts="messagebus"

restart the host or use /etc/netstart

To start gnome you have to launch X with the binary of gnome session, like the command below:

**$ startx /usr/local/bin/gnome-session**

**OpenVPN**

VPNs are becoming increasingly popular, as they allow companies to join the LANs of their branches or subsidiaries into a single private network (site-to-site VPNs) or to provide mobile employees, such as sales people, access to their corporate network from outside the premises (remote-access VPNs), thus making accessing and sharing internal information much easier.

OpenVPN is a SSL-based VPN solution, operating at the application layer and probably

the strongest contender for IPsec, thanks to its robustness, ease of use and portability.

edit /etc/rc.securelevel

# sysctl net.inet.esp.enable

net.inet.esp.enable=1

# sysctl net.inet.ah.enable

net.inet.ah.enable=1

# sysctl net.inet.ip.forwarding=1

# sysctl net.inet.ipcomp.enable=1

net.inet.ipcomp.enable: 0 -> 1

# ifconfig enc0 up

# mkdir -p /etc/openvpn/private

# chmod 700 /etc/openvpn/private

Hacking Tools

#pkg\_add -iv firewalk, nmap, hping, fping, hydra dsniff

Metasploit

crontab

# crontab -e -u root

SSL Certificates

# openssl req -x509 -days 365 -newkey rsa:4096 \

> -keyout /etc/ssl/private/ca.key \

> -out /etc/ssl/ca.crt

# openssl req -new -key /etc/isakmpd/private/local.key \

> -out /etc/isakmpd/private/1.2.3.4.csr

Partitions Permissions

Before going to production is a good practice to set stricter permissions on every partitions, in order to resize the attack surface for some vectors that abuse users permissions.

/etc/fstab

**Before permissions hardening**

9d1482c3ba14.b none swap sw

9d1482c3ba14.a / ffs rw 1 1

9d1482c3ba14.k /home ffs rw,nodev,nosuid 1 2

9d1482c3ba14.d /tmp ffs rw,nodev,nosuid 1 2

9d1482c3ba14.f /usr ffs rw,nodev 1 2

9d1482c3ba14.g /usr/X11R6 ffs rw,nodev 1 2

9d1482c3ba14.h /usr/local ffs rw,nodev 1 2

9d1482c3ba14.j /usr/obj ffs rw,nodev,nosuid 1 2

9d1482c3ba14.i /usr/src ffs rw,nodev,nosuid 1 2

9d1482c3ba14.e /var ffs rw,nodev,nosuid 1 2

**After permissions hardening**

9d1482c3ba14.b none swap sw

9d1482c3ba14.a / ffs rw 1 1

9d1482c3ba14.k /home ffs rw,nodev,nosuid,**noexec** 1 2

9d1482c3ba14.d /tmp ffs rw,nodev,nosuid,**noexec** 1 2

9d1482c3ba14.f /usr ffs **ro**,nodev 1 2

9d1482c3ba14.g /usr/X11R6 ffs **ro**,nodev 1 2

9d1482c3ba14.h /usr/local ffs **ro**,nodev 1 2

9d1482c3ba14.j /usr/obj ffs **ro**,nodev,nosuid 1 2

9d1482c3ba14.i /usr/src ffs **ro**,nodev,nosuid 1 2

9d1482c3ba14.e /var ffs rw,nodev,nosuid,**noexec** 1 2

**sappnd** set the system append-only flag

**schg** set the system immutable flag

AIDE

(Advanced Intrusion Detection Environment)

Quick Commands Reference

| **uptime** | Show the up time from last reboot |
| --- | --- |
| **ls -al** | formatted listing with hidden files |
| **cd [dir]** | change directory to [dir] |
| **cd -** | change to home |
| **pwd** | show current directory |
| **mkdir [dir]** | create a new directory |
| **rm *[file]*** | delete a file |
| **rm -rf [dir]** | delete a directory recursively |
| **cp [fileA] [fileB]** | copy a file |
| **mv [fileA] [fileB]** | rename a file or directory |
| **ln -s** | create a symbolic link to a file |
| **touch [file]** | create a file |
| **cat > [file]** | place standard output to a file |
| **more [file]** | output the content of a file |
| **head [file]** | output the first 10 lines of a file |
| **tail [file]** | output the last 10 lines of a file |
| **tail -f [file]** | output the last 10 line of a file as it grows |
| **tail -n** *[rows] [file]* | output the last defined rows of a file |
| **top** | display all running processes and the details |
| **ps** | display your current active processes |
| **kill [pid]** | kill a process with pid |
| **kill -9 [pid]** | force the killing of a process by pid |
| **fg** | brings the most recent job to foreground |
| **chmod** *[octal][file]* | changes the permissions of a file |
| **chown [user]:[group] [file]** | set the ownership of a file to a user and group |
| **date** | show the current date and time |
| **cal** | show this month’s calendar |
| **w** | display who is online |
| **whoami** | display who you are logged in as |
| **uname -a** | show kernel version |
| **finger** *[user]* | display information about the user |
| **cat /proc/cpuinfo** | display CPU information |
| **cat /proc/meminfo** | display memory information |
| **df -h** | show disk usage for every partition |
| **du** | show directory space usage |
| **man** *[command]* | show the manual for a command |
| **free** | show memory and swap usage |
| **ping** *[host]* | ping a host and show the results |
| **dig** *[domain]* | get DNS information for a domain |
| **dig -x** *[host]* | reverse lookup host |
| **diff** *[fileA] [fileB]* | show differences between two files |
| **wget** *[file]* | download a file |
| **wget -c** *[file]* | continue a stopped donwload |
| **gunzip** *[file]* | decompress a file .gz |
| **tar -czvf** arch dir | compress directory into a compressed archive |
| **tar -xvf** | extract files from a compressed archive |
| **pkg\_add** | add a packages from repository |
| **pkg\_delete** | delete a installed package |
| **pkg\_info** | list all the installed packages |
| **grep [pattern] [files]** | search for pattern in all files |
| **grep -r [pattern] [dir]** | search recursively for pattern in a directory |
| **locate [file]** | find all instances of a file |
| **killall [proc]** | Kill all processes named proc |
| **[command]|grep [pattern]** | search for patterns in the output of a command |
| **netstat** | show network connections statistics |
| **history** | show the last given commands |
| **CTRL+C** | halt the current command |
| **CTRL+Z** | stop the current command |
| **CTRL+D** | log out the current session |
| **CTRL+W** | erase one word in the current line |
| **CTRL+U** | erase the whole word |
| **CTRL+R** | type to bring up a recent command |
| **!!** | repeats the last command |
| **exit** | log out the current session |