.



7/9/2014

Robin Toy

STROBE TECHNOLOGIES LTD



SpamSnake

MTA AV & Spam Filter Install

Contents

[Introduction 2](#_Toc399397653)

[Version information 2](#_Toc399397654)

[Ubuntu Installation 3](#_Toc399397655)

[Install OS 3](#_Toc399397656)

[Configure OS 3](#_Toc399397657)

[Script Setup 5](#_Toc399397658)

[Base Setup 5](#_Toc399397659)

[Phase 1 Setup 5](#_Toc399397660)

[Install and Configure Mail Filter 6](#_Toc399397661)

[Install Additional Filters 6](#_Toc399397662)

[Spamassassin Configuration 6](#_Toc399397663)

[Phase 2 8](#_Toc399397664)

[Install Baruwa 8](#_Toc399397665)

[Install / Configure Webserver for Baruwa 11](#_Toc399397666)

[Install Additional Protection / Filters 13](#_Toc399397667)

[Install and configure SPF 13](#_Toc399397668)

[Install and Configure FuzzyOCR 13](#_Toc399397669)

[Filtering PDF, XLS and Phishing Spam with ClamAV (Sanesecurity Signatures) 17](#_Toc399397670)

[Greylisting Installing 18](#_Toc399397671)

[Make & Configure KAM 18](#_Toc399397672)

[Install ScamNailer 20](#_Toc399397673)

[Protect SpamSnake 29](#_Toc399397674)

[Install Firehol 29](#_Toc399397675)

[Create Update Scripts 29](#_Toc399397676)

[Configure Firehol 34](#_Toc399397677)

[Updating Software 34](#_Toc399397678)

[Branding 35](#_Toc399397679)

[Scheduling Updates and Tasks 35](#_Toc399397680)

# Introduction

This guide is almost a set by step guide how to create / configure SpamSnake based on Ubuntu LTS Server Edition. Thanks goes out to Rocky the author of The Perfect SpamSnake - Ubuntu 12.04 LTS Precise Pangolin, and Credit goes to the guys at HowToForge and the developers of MailScanner, Baruwa, Clamav, Nginx/Uwsgi, Mysql, Postfix, Spamassassin, Razor/Pyzor/DCC and Firehol.

# Version information

**SpamSnake Version:** V1.12.2.2 (Baruwa 1.x, Ubuntu 12.x, MailScanner 4.x, Strobe IT 2.x)

**Operating System:** Ubuntu 12.04 LTS Server Edition [Precise Pangolin] (32/64 bit)

**Web Server:** Nginx v1.1.19/Uwsgi v1.0.3

**Database Server:** MySQL v5.5.28

**Mail Server:** Postfix v2.9.3

**Caching DNS Server:** Dnsmasq 2.59

**Filter(s):** MailScanner v4.84.6-1 / Spamassassin v3.3.2-2

**Frontend(s):** Baruwa v1.1.2-4sn / Webmin 1.690

# Ubuntu Installation

To install the base OS ready for installing all the required add-ons please follow “Install OS” followed by “Configure OS”.

## Install OS

1. Insert ***Ubuntu Server CD***
2. Follow basic install screens
   1. Hostname = <Local Server Name> e.g ST-SVR04
3. Create user ***administrator***
4. Password of ***Passw0rd***
5. Encrypt your home directory? Select ***No***
6. Partitionaing method = ***Guided – use entire disk and set up LVM***
7. Update Method of ***No Auto Updates***
8. Choose ***OpenSSH*** on the software selection
9. Install the GRUB boot loader too master boot record? ***YES***
10. Remove CD and Continue to restart and launch the server

## Configure OS

1. Login with username administrator
2. Enable the root account and set it a password
   1. *sudo passwd root*
   2. Provide the password of ***IG88H1r3d***
3. Log out with the command *Exit*
4. Login with username root
5. Now to change the default OS Shell from Dash to Bash, to do this follow these steps:
   1. *dpkg-reconfigure dash*
   2. When asked to install Dash as shell select ***No***
6. Finally we need to install base tools and utilities.
   1. Get server IP with *ifconfig*
   2. Connect to the server using Putty on the listed IP within ifconfig on port 22
   3. Log out the server screen (not Putty) with the command *Exit*
7. Install base OS packages by running option a from the installer then exit installer
8. Edit the hosts file with *vi /etc/hosts*
9. Make the file look like (changing the RED for your settings)

*127.0.0.1 localhost.localdomain localhost*

*IP-Address FQDN Hostname*

*IP-Address External-Server-Name*

*# The following lines are desirable for IPv6 capable hosts*

*::1 localhost ip6-localhost ip6-loopback*

*fe00::0 ip6-localnet*

*ff00::0 ip6-mcastprefix*

*ff02::1 ip6-allnodes*

*ff02::2 ip6-allrouters*

*ff02::3 ip6-allhosts*

1. ?Copy the new name to hostname with *echo server1.example.com > /etc/hostname*
2. ?Reboot the server with *reboot now*

# Script Setup

Get script from ……

Save script to /usr/src/

Make script executable with *chmod +x snakesetup.sh*

# Base Setup

Launch setup script with *./snakesetup.sh*

Choose option **a**

# Phase 1 Setup

If you are not in the snake setup menu launch it with ./snakesetup.sh

Choose option **b**

During the install MySQL will ask for a root password, enter **5n@keSpam**

During the install of Postfix when asked for type of mail setup choose **Internet Site**

During the install of Postfix when asked for the systems mail name enter the *External Server Name*

# Install and Configure Mail Filter

This section is all about installing the mail filters and linking them all together to work through MailScanner as the main engine. These filters / applications include: -

* ClamAV
* DCC
* Pyzor & Razor
* Spamassassin

## Install Additional Filters

1. Configure ClamAV for Apparmor
   1. Edit the profile for clamd with vi by *vi /etc/apparmor.d/usr.sbin.clamd*
   2. Enter vi edit mode by pressing **I** (Insert)

#clamav

/var/spool/MailScanner/\*\* rw,

/var/spool/MailScanner/incoming/\*\* rw,

* 1. Press **Esc** to exit to command mode
  2. Save and quit vi with *:wq* (write then quit)

1. Reload Apparmor with */etc/init.d/apparmor reload*

## Spamassassin Configuration

1. Disable the default config file by renaming it with *mv /etc/spamassassin/local.cf /etc/spamassassin/local.cf.disabled*
2. Backup the MailScanner Spamassassin setting *by cp /opt/MailScanner/etc/spam.assassin.prefs.conf /opt/MailScanner/etc/spam.assassin.prefs.conf.back*
3. Create the MySQL database on the server where you intend on storing the bayesian information
   1. *mysql -u root –p* Press **Enter**
   2. *create database sa\_bayes;* Press **Enter**
   3. *GRANT ALL ON sa\_bayes.\* TO sa\_user@localhost IDENTIFIED BY 'sa\_password';* Press **Enter**
   4. *flush privileges;* Press **Enter**
   5. *exit;* Press **Enter**
4. Correct bayes SQL structure script
   1. Edit script with *vi /usr/share/doc/spamassassin/sql/bayes\_mysql.sql*
   2. Enter vi edit mode by pressing **I** (Insert)
   3. Change the word **TYPE** to **ENGINE** in all statements of the script
   4. Press **Esc** to exit to command mode
   5. Save and quit vi with
      1. *:wq* (write then quit)
5. Import the database structure by running the script with *mysql -u sa\_user -p sa\_bayes < /usr/share/doc/spamassassin/sql/bayes\_mysql.sql*
6. Enable DCC plugin by opening config file with *vi /etc/spamassassin/v310.pre*
7. Enter vi edit mode by pressing **I** (Insert)
8. Modify the file to have

loadplugin Mail::SpamAssassin::Plugin::DCC

1. Press **Esc** to exit to command mode
2. Save and quit vi with
   1. *:wq* (write then quit)
3. Run the following to create folders to prevent an error in a lint test
   1. *mkdir /var/www*
   2. *mkdir /var/www/.spamassassin*
4. Open the Spamassassin config with vi for numerous edits by *vi /opt/MailScanner/etc/spam.assassin.prefs.conf*
5. Enable Razor and Pyzor by adding the following to the top of the config file

#pyzor

use\_pyzor 1

pyzor\_options --homedir /var/lib/MailScanner/

#razor

use\_razor2 1

razor\_config /var/lib/MailScanner/.razor/razor-agent.conf

1. Fix the DCC Path so it reads as

dcc\_path /usr/bin/dccproc

1. Update the following to be correct and make changes to the RED text to meet your configuration requirements

bayes\_ignore\_header X-YOURDOMAIN-COM-SpamSnake

bayes\_ignore\_header X-YOURDOMAIN-COM- SpamSnake -SpamCheck

bayes\_ignore\_header X-YOURDOMAIN-COM- SpamSnake -SpamScore

bayes\_ignore\_header X-YOURDOMAIN-COM- SpamSnake -Information

#use\_auto\_whitelist 0

envelope\_sender\_header X-SpamSnake-From

header MS\_FOUND\_SPAMVIRUS exists:X-SpamSnake-SpamVirus-Report

1. Add SQL connection string to bottom of the file

bayes\_store\_module Mail::SpamAssassin::BayesStore::SQL

bayes\_sql\_dsn DBI:mysql:sa\_bayes:localhost

bayes\_sql\_username sa\_user

bayes\_sql\_password sa\_password

bayes\_sql\_override\_username root

1. Press **Esc** to exit to command mode
2. Save and quit vi with
   1. *:wq* (write then quit)
3. Install missing perl packages
   1. *perl -MCPAN -e shell* Press **Enter** (Auto = YES, Mirrors = YES)
   2. *install IP::Country::Fast* Press **Enter**
   3. *install Encode::Detect* Press **Enter**
   4. *install Crypt::OpenSSL::RSA* Press **Enter**
   5. *exit* Press **Enter**
4. Set the permissions to allow access for all components with
   1. *chown -R postfix:www-data /var/spool/postfix/hold*
   2. *chmod -R ug+rwx /var/spool/postfix/hold*

# Phase 2

## Install Baruwa

1. Install Baruwa & docs with
   1. During the install you will be asked for some information that is listed in the above information panel. Summary is: -
      1. Webserver to configure manually (nginx)
      2. Virtual Host name
      3. Choose whether to configure mysql automatically (YES)
      4. My Host
      5. Database administrator name/password
      6. Baruwa username/password/dbname
      7. RabbitMQ host/username/password
      8. Admin user/password/email for Baruwa Web
2. Edit the settings.py so we can edit the database settings *vi /etc/baruwa/settings.py*
3. Enter vi edit mode by pressing **I** (Insert)
4. Update the file to below changing the RED to match your settings

DATABASES = {

'default': {

# Add 'postgresql\_psycopg2', 'postgresql', 'mysql', 'sqlite3'

# or 'oracle'.

'ENGINE': 'django.db.backends.mysql',

# Or path to database file if using sqlite3.

'NAME': 'baruwa',

# Not used with sqlite3.

'USER': 'baruwa',

# Not used with sqlite3.

'PASSWORD': '5n@keSpam',

# Set to empty string for localhost. Not used with sqlite3.

'HOST': '',

# Set to empty string for default. Not used with sqlite3.

'PORT': '',

# Recommended for MySQL. See http://code.djangoproject.com/ticket/13906

'OPTIONS': {'init\_command': 'SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED'},

}

}

TIME\_ZONE = 'Europe/London'

DEFAULT\_FROM\_EMAIL = 'SpamSnake@Domain.name'

QUARANTINE\_REPORT\_HOSTURL = 'http://baruwa-alpha.local'

1. Press **Esc** to exit to command mode
2. Save and quit vi with
   1. *:wq* (write then quit)
3. Populate the Baruwa database by running
   1. *baruwa-admin syncdb --noinput*
   2. *for name in $(echo "accounts messages lists reports status fixups config"); do*
   3. *baruwa-admin migrate $name;*
   4. *done*
4. Open the MailScanner config file with *vi /etc/MailScanner/MailScanner.conf*
5. Enter vi edit mode by pressing **I** (Insert)
6. Update the MailScanner config to use Baruwa by changing the below lines to

Run As Group = celeryd

Quarantine User = celery

Quarantine Group = celery

Is Definitely Not Spam = &BaruwaWhitelist

Is Definitely Spam = &BaruwaBlacklist

Required SpamAssassin Score = &BaruwaLowScore

High SpamAssassin Score = &BaruwaHighScore

Always Looked Up Last = &BaruwaSQL

1. Press **Esc** to exit to command mode
2. Save and quit vi with
3. *:wq* (write then quit)
4. We now need to edit the Barwua config with *vi /opt/MailScanner/etc/conf.d/baruwa.conf*
5. Enter vi edit mode by pressing **I** (Insert)
6. Update the file to include

Quarantine User = celeryd

DB DSN = DBI:mysql:database=baruwa;host=localhost;port=3306

DB Username = baruwa

DB Password = 5n@keSpam

#Inline HTML Signature = htmlsigs.customize

#Inline Text Signature = textsigs.customize

#Signature Image Filename = sigimgfiles.customize

#Signature Image Filename = sigimgs.customize

1. Press **Esc** to exit to command mode
2. Save and quit vi with
3. *:wq* (write then quit)
4. Update MailScanner startup script with *vi /etc/init.d/mailscanner*
5. Enter vi edit mode by pressing **I** (Insert)
6. Update the file to

CONFFILE=/etc/MailScanner/MailScanner.conf

PIDFILE=/var/run/MailScanner/MailScanner.pid

check\_dir /var/spool/MailScanner ${user:-postfix} ${group:-celeryd}

check\_dir /var/lib/MailScanner ${user:-postfix} ${group:-celeryd}

check\_dir /var/run/MailScanner ${user:-postfix} ${group:-celeryd}

check\_dir /var/lock/subsys/MailScanner ${user:-postfix} ${group:-celeryd}

start-stop-daemon --start --quiet --startas $STARTAS --pidfile $PIDFILE --test > /dev/null \

start-stop-daemon --start --quiet --nicelevel $run\_nice --chuid postfix:celeryd --exec $DAEMON --pidfile "$PIDFILE" -- $DAEMON\_ARGS \

start-stop-daemon --stop --retry=TERM/30 --pidfile $PIDFILE

start-stop-daemon --stop --signal 1 --quiet --pidfile $PIDFILE

1. Press **Esc** to exit to command mode
2. Save and quit vi with
3. *:wq* (write then quit)
4. Add celeryd user to clamav group with *usermod -a -G celeryd clamav*
5. Change the group ownership of the quarantine folder and content with *chgrp -R celeryd /var/spool/MailScanner/quarantine*
6. Initialise signatures with *baruwa-admin initconfig*
7. Create the super user with *baruwa-admin createsuperuser*
8. Reboot the server with *shutdown -r now*

## Install / Configure Webserver for Baruwa

1. Install nginx and uwsgi by running
   1. *apt-get install nginx-full uwsgi uwsgi-plugin-python -y*
2. Create a new Baruwa config file with *vi /etc/uwsgi/apps-available/baruwa.ini*
3. Enter vi edit mode by pressing **I** (Insert)
4. Populate file with

[uwsgi]

workers = 2

chdir = /usr/share/pyshared/baruwa

env = DJANGO\_SETTINGS\_MODULE=baruwa.settings

module = django.core.handlers.wsgi:WSGIHandler()

1. Press **Esc** to exit to command mode
2. Save and quit vi with
3. *:wq* (write then quit)
4. Create Baruwa config with *vi /etc/nginx/sites-available/baruwa.conf*
5. Enter vi edit mode by pressing **I** (Insert)
6. Populate file with except that you need to change the RED to meet your settings

server {

listen 80;

server\_name example.com;

root /usr/share/pyshared/baruwa;

autoindex on;

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

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

location /static {

root /usr/share/pyshared/baruwa/static/;

}

# static resources

location ~\* ^.+\.(html|jpg|jpeg|gif|png|ico|css|zip|tgz|gz|rar|bz2|doc|xls|exe|pdf|ppt|txt|tar|mid|midi|wav|bmp|rtf|js)$

{

expires 30d;

break;

}

location / {

uwsgi\_pass unix:///var/run/uwsgi/app/baruwa/socket;

include uwsgi\_params;

}

}

1. Press **Esc** to exit to command mode
2. Save and quit vi with
3. *:wq* (write then quit)
4. Create Symlinks and remove default hosts with
   1. *ln -s /etc/nginx/sites-available/baruwa.conf /etc/nginx/sites-enabled/baruwa.conf*
   2. *ln -s /etc/uwsgi/apps-available/baruwa.ini /etc/uwsgi/apps-enabled/baruwa.ini*
   3. *rm -r /etc/nginx/sites-enabled/default*
   4. *cp /usr/share/doc/uwsgi-extra/nginx/uwsgi\_params /etc/nginx/uwsgi\_params ??????*
5. Restart Services with */etc/init.d/uwsgi restart && /etc/init.d/nginx restart*
6. Create a symlink to manage.py
   1. *ln -s /usr/share/pyshared/baruwa/manage.py /usr/bin/manage.py*
   2. *chmod +x /usr/bin/manage.py*
7. Start MailScanner with */etc/init.d/mailscanner start*

# Install Additional Protection / Filters

## Install and configure SPF

1. Install SPF engine with *apt-get install postfix-policyd-spf-python -y*
2. Edit the Postfix mail server config with *vi /etc/postfix/master.cf*
3. Enter vi edit mode by pressing **I** (Insert)
4. Add the following lines to the file
   1. \*Note: (The leading spaces before user=nobody are important so that Postfix knows that this line belongs to the previous one!)
   2. \*Note: We already added the entry for main.cf using the postfix setup script.

policy-spf unix - n n - - spawn

user=nobody argv=/usr/bin/policyd-spf

1. Press **Esc** to exit to command mode
2. Save and quit vi with
3. *:wq* (write then quit)
4. Restart Postfix with */etc/init.d/postfix restart*

## Install and Configure FuzzyOCR

1. FuzzyOCR has some prerequisites like ocrad and gocr that we can install with this command

apt-get install fuzzyocr netpbm gifsicle libungif-bin gocr ocrad libstring-approx-perl libmldbm-sync-perl libdigest-md5-perl libdbd-mysql-perl imagemagick tesseract-ocr -y

1. Download FuzzyOCR
   1. *wget http://users.own-hero.net/~decoder/fuzzyocr/fuzzyocr-3.6.0.tar.gz*
   2. *tar xvfz fuzzyocr-3.6.0.tar.gz && cd FuzzyOcr-3.6.0/*
2. Open the configuration file with *vi /etc/spamassassin/FuzzyOcr.cf*
3. Enter vi edit mode by pressing **I** (Insert)
4. uncomment the following lines

focr\_global\_wordlist /etc/spamassassin/FuzzyOcr.words

focr\_preprocessor\_file /etc/spamassassin/FuzzyOcr.preps

focr\_scanset\_file /etc/spamassassin/FuzzyOcr.scansets

focr\_enable\_image\_hashing 3

focr\_digest\_db /etc/spamassassin/FuzzyOcr.hashdb

focr\_db\_hash /etc/spamassassin/FuzzyOcr.db

focr\_db\_safe /etc/spamassassin/FuzzyOcr.safe.db

focr\_bin\_helper convert, tesseract

1. Comment out the path

#focr\_path\_bin /usr/local/netpbm/bin:/usr/local/bin:/usr/bin

1. Enable the following lines

focr\_mysql\_db FuzzyOcr

focr\_mysql\_hash Hash

focr\_mysql\_safe Safe

focr\_mysql\_user fuzzyocr

focr\_mysql\_pass fuzzyocr

focr\_mysql\_host localhost

focr\_mysql\_port 3306

focr\_mysql\_socket /var/run/mysqld/mysqld.sock

1. Press **Esc** to exit to command mode
2. Save and quit vi with
3. *:wq* (write then quit)
4. Correct SQL structure script
   1. Edit script with *vi FuzzyOcr.mysql*
   2. Enter vi edit mode by pressing **I** (Insert)
   3. Change the word **TYPE** to **ENGINE** in all statements of the script
   4. Press **Esc** to exit to command mode
   5. Save and quit vi with
      1. *:wq* (write then quit)
5. Import the database structure by running the script with *mysql -p < FuzzyOcr.mysql*
6. Set up FuzzyOcr Database Cleaner by creating a new file with *vi /usr/sbin/fuzzy-cleanmysql*
7. Enter vi edit mode by pressing **I** (Insert)
8. Populate the file with

#!/usr/bin/perl

#Script to clean out mysql tables of data. Default is to leave data in Safe for 1 day and Hash for 10 days.

#Fuzzyocr-cleanmysql

use Getopt::Long;

use DBI;

use MLDBM qw(DB\_File Storable);

my %Files = (

db\_hash => '/var/lib/fuzzyocr/FuzzyOcr.db',

db\_safe => '/var/lib/fuzzyocr/FuzzyOcr.safe.db',

);

use DBI;

$database = "FuzzyOcr";

$hostname = "localhost";

$socket = "/var/run/mysqld/mysqld.sock";

$port = "3306";

$username = "fuzzyocr";

$password = 'password';

# defaults

my $cfgfile = "/etc/spamassassin/FuzzyOcr.cf";

my %App;

my %age;

$age{'age'} = 10\*24; # 10 days

$age{'hash'} = $age{'age'};

$age{'safe'} = 0;

my $help = 0;

my $verbose = 0;

GetOptions( \%age,

'age=i',

'config=s' => \$cfgfile,

'hash=i',

'help' => \$help,

'safe=i',

'verbose' => \$verbose,

);

if ($help) {

print "Usage: fuzzy-cleanmysql [Options]\n";

print "\n";

print "Available options:\n";

print "--age=i Global age in hours to keep in db\n";

print "--config=s Specify location of FuzzyOcr.cf\n";

print " Default: /etc/spamassassin/FuzzyOcr.cf\n";

print "--hash=i Number of hours old to keep in Hash db\n";

print "--safe=i Number of hours old to keep in Safe db\n";

print "--verbose Show more informations\n";

print "\n";

exit 1;

}

# Convert hours to seconds

$age{'age'} \*= 60 \* 60;

$age{'hash'} \*= 60 \* 60;

$age{'safe'} \*= 60 \* 60;

$age{'safe'} = $age{'safe'} ? $age{'safe'} : $age{'age'};

# Read custom paths from FuzzyOcr.cf

my $app\_path = q(/usr/local/netpbm/bin:/usr/local/bin:/usr/bin);

open CONFIG, "< $cfgfile" or warn "Can't read configuration file, using defaults...\n";

while () {

chomp;

if ($\_ =~ m/^focr\_bin\_(\w+) (.+)/) {

$App{$1} = $2;

printf "Found custom path \"$2\" for application \"$1\"\n" if $verbose;

}

if ($\_ =~ m/^focr\_path\_bin (.+)/) {

$app\_path = $1;

printf "Found new path: \"$1\"\n" if $verbose;

}

if ($\_ =~ m/^focr\_enable\_image\_hashing (\d)/) {

$App{hashing\_type} = $1;

printf "Found DB Hashing\n" if ($verbose and $1 == 2);

printf "Found MySQL Hashing\n" if ($verbose and $1 == 3);

}

if ($\_ =~ m/^focr\_mysql\_(\w+) (.+)/) {

$MySQL{$1} = $2;

printf "Found MySQL option $1 => '$2'\n" if $verbose;

}

if ($\_ =~ m/^focr\_threshold\_max\_hash (.+)/) {

$App{max\_hash} = $1;

printf "Updated Thresold{max\_hash} = $1\n" if $verbose;

}

}

close CONFIG;

# make shure we have this threshold set

$App{max\_hash} = 5 unless defined $App{max\_hash};

# search path for bin\_util unless already specified in configuration file

foreach my $app (@bin\_utils) {

next if defined $App{$app};

foreach my $d (split(':',$app\_path)) {

if (-x "$d/$app") {

$App{$app} = "$d/$app";

last;

}

}

}

sub get\_ddb {

my %dopts = ( AutoCommit => 1 );

my $dsn = "DBI:mysql:database=$database";

if (defined $socket) {

$dsn .= ";mysql\_socket=$socket";

} else {

$dsn .= ";host=$hostname";

$dns .= ";port=$port" unless $port == 3306;

}

printf "Connecting to: $dsn\n" if $verbose;

return DBI->connect($dsn, $username, $password,\%dopts) or die("Could not connect!");

}

if ($App{hashing\_type} == 3) {

my $ddb = get\_ddb();

if ($ddb) {

my $sql;

foreach my $ff (sort keys %Files) {

$ff =~ s/db\_//;

$sqlbase = "FROM $MySQL{$ff} WHERE $MySQL{$ff}.\`check\` < ?";

my $timestamp = time;

$timestamp = $timestamp - $age{$ff};

$sql = "DELETE $sqlbase";

if ( $verbose ) {

printf "Delete from Table $MySQL{$ff}\n";

print "$sql, $timestamp\n";

print "Timestamp is ", scalar(localtime($timestamp)), "\n";

print "That's $age{$ff} seconds earlier than now.\n";

print "\n";

}

$ddb->do($sql,undef,$timestamp);

}

$ddb->disconnect;

}

}

1. Press **Esc** to exit to command mode
2. Save and quit vi with
3. *:wq* (write then quit)
4. Make the script executable with *chmod +x /usr/sbin/fuzzy-cleanmysql*

## Filtering PDF, XLS and Phishing Spam with ClamAV (Sanesecurity Signatures)

1. Install dependencies with *apt-get install curl rsync -y*
2. Make directory and enter it with *mkdir /usr/src/sanesecurity && cd /usr/src/sanesecurity*
3. Download Signatures *wget http://downloads.sourceforge.net/project/unofficial-sigs/clamav-unofficial-sigs-3.7.2.tar.gz*
4. Extract Signatures *tar -zxf clamav-unofficial-sigs-3.7.2.tar.gz && cd clamav-unofficial-sigs-3.7.2*
5. Move application script with *mv clamav-unofficial-sigs.sh /usr/sbin*
6. Move application config file with *mv clamav-unofficial-sigs.conf /etc/*
7. Make the application script executable with *chmod +x /usr/sbin/clamav-unofficial-sigs.sh*
8. Open the config file with *vi /etc/clamav-unofficial-sigs.conf*
9. Enter vi edit mode by pressing **I** (Insert)
10. Update the file with the following information

clam\_dbs="/var/lib/clamav"

clamd\_pid="/var/run/clamav/clamd.pid"

reload\_dbs="yes"

reload\_opt="kill -USR2 `cat $clamd\_pid`" #Signals PID to reload dbs

work\_dir="/var/lib/clamav"

user\_configuration\_complete="yes"

1. Press **Esc** to exit to command mode
2. Save and quit vi with
3. *:wq* (write then quit)

## Greylisting Installing

1. Change to source directory with *cd /usr/src*
2. Download with *wget http://www.kim-minh.com/pub/greyfix/greyfix-0.4.0.tar.gz*
3. Extract Greyfix with *tar -xf greyfix-0.4.0.tar.gz && cd greyfix-0.4.0*
4. Set variables for installation by running *./configure --localstatedir=/var*
5. Create installer with command *make*
6. Install Greyfix with *make install*
7. Edit the Postfix mail server config with *vi /etc/postfix/master.cf*
8. Enter vi edit mode by pressing **I** (Insert)
9. Add the following lines to the file

greyfix unix - n n - - spawn

user=nobody argv=/usr/local/sbin/greyfix --greylist-delay 60 -/ 24

1. Press **Esc** to exit to command mode
2. Save and quit vi with
3. *:wq* (write then quit)
4. Restart Postfix with */etc/init.d/postfix restart*

## Make & Configure KAM

1. Create a script file with *vi /etc/cron.daily/kam.sh*
2. Enter vi edit mode by pressing **I** (Insert)
3. Add the following lines to the file

#!/bin/bash

# Original version modified by Andrew MacLachlan (andrew@gdcon.net)

# Added additional MailScanner restarts on inital restart failure

# Made script run silently for normal (successful) operation

# Increased UPDATEMAXDELAY to 900 from 600

# Insert a random delay up to this value, to spread virus updates round

# the clock. 1800 seconds = 30 minutes.

# Set this to 0 to disable it.

UPDATEMAXDELAY=0

if [ -f /opt/MailScanner/var/MailScanner ] ; then

. /opt/MailScanner/var/MailScanner

fi

export UPDATEMAXDELAY

if [ "x$UPDATEMAXDELAY" = "x0" ]; then

:

else

logger -p mail.info -t KAM.cf.sh Delaying cron job up to $UPDATEMAXDELAY seconds

perl -e "sleep int(rand($UPDATEMAXDELAY));"

fi

# JKF Fetch KAM.cf

#echo Fetching KAM.cf...

cd /etc/mail/spamassassin

rm -f KAM.cf

wget -O KAM.cf http://www.peregrinehw.com/downloads/SpamAssassin/contrib/KAM.cf > /dev/null 2>&1

if [ "$?" = "0" ]; then

#echo It completed and fetched something

if ( tail -10 KAM.cf | grep -q '^#.\*EOF' ); then

# echo It succeeded so make a backup

cp -f KAM.cf KAM.cf.backup

else

echo ERROR: Could not find EOF marker

cp -f KAM.cf.backup KAM.cf

fi

else

echo It failed to complete properly

cp -f KAM.cf.backup KAM.cf

fi

#echo Reloading MailScanner and SpamAssassin configuration rules

/etc/init.d/mailscanner reload > /dev/null 2>&1

if [ $? != 0 ] ; then

echo "MailScanner reload failed - Retrying..."

/etc/init.d/mailscanner force-reload

if [ $? = 0 ] ; then

echo "MailScanner reload succeeded."

else

echo "Stopping MailScanner..."

/etc/init.d/mailscanner stop

echo "Waiting for a minute..."

perl -e "sleep 60;"

echo "Attemping to start MailScanner..."

/etc/init.d/mailscanner start

fi

fi

1. Press **Esc** to exit to command mode
2. Save and quit vi with
3. *:wq* (write then quit)
4. Make the script executable with *chmod +x /etc/cron.daily/kam.sh*

## Install ScamNailer

1. Create the update script with *vi /opt/MailScanner/bin/update\_scamnailer*
2. Enter vi edit mode by pressing **I** (Insert)
3. Add the following lines to the file

#!/usr/bin/perl

#

# (c) 2009 Julian Field ‹ScamNailer@ecs.soton.ac.uk›

# Version 2.05

#

# This file is the copyright of Julian Field ‹ScamNailer@ecs.soton.ac.uk›,

# and is made freely available to the entire world. If you intend to

# make any money from my work, please contact me for permission first!

# If you just want to use this script to help protect your own site's

# users, then you can use it and change it freely, but please keep my

# name and email address at the top.

#

use strict;

use File::Temp;

use Net::DNS::Resolver;

use LWP::UserAgent;

use FileHandle;

use DirHandle;

# Filename of list of extra addresses you have added, 1 per line.

# Does not matter if this file does not exist.

my $local\_extras = '/etc/MailScanner/ScamNailer.local.addresses';

# Output filename, goes into SpamAssassin. Can be over-ridden by just

# adding the output filename on the command-line when you run this script.

my $output\_filename = '/etc/mail/spamassassin/ScamNailer.cf';

# This is the location of the cache used by the DNS-based updates to the

# phishing database.

my $emailscurrent = '/var/cache/ScamNailer/';

# Set this next value to '' if ou are not using MailScanner.

# Or else change it to any command you need to run after updating the

# SpamAssassin rules, such as '/sbin/service spamd restart'.

my $mailscanner\_restart = '/etc/init.d/mailscanner force-reload';

# The SpamAssassin score to assign to the final rule that fires if any of

# the addresses hit. Multiple hits don't increase the score.

#

# I use a score of 0.1 with this in MailScanner.conf:

# SpamAssassin Rule Actions = SCAMNAILER=>not-deliver,store,forward postmaster@my-domain.com, header "X-Anti-Phish: Was to \_TO\_"

# If you don't understand that, read the section of MailScanner.conf about the

# "SpamAssassin Rule Actions" setting.

my $SA\_score = 4.0;

# How complicated to make each rule. 20 works just fine, leave it alone.

my $addresses\_per\_rule = 20;

my $quiet = 1 if grep /quiet|silent/, @ARGV;

if (grep /help/, @ARGV) {

print STDERR "Usage: $0 [ --quiet ]\n";

exit(1);

}

my($count, $rule\_num, @quoted, @addresses, @metarules);

#local(\*YPCAT, \*SACF);

local(\*SACF);

$output\_filename = $ARGV[0] if $ARGV[0]; # Use filename if they gave one

# First do all the addresses we read from DNS and anycast and only do the

# rest if needed.

if (GetPhishingUpdate()) {

open(SACF, ">$output\_filename") or die "Cannot write to $output\_filename $!";

print SACF "# ScamNailer rules\n";

print SACF "# Generated by $0 at " . `date` . "\n";

# Now read all the addresses we generated from GetPhishingUpdate().

open(PHISHIN, $emailscurrent . 'phishing.emails.list')

or die "Cannot read " . $emailscurrent . "phishing.emails.list, $!\n";

while(

<phishin>) {

chomp;

s/^\s+//g;

s/\s+$//g;

s/^#.\*$//g;

next if /^\s\*$/;

next unless /^[^@]+\@[^@]+$/;

push @addresses, $\_; # This is for the report

s/[^0-9a-z\_-]/\\$&/ig; # Quote every non-alnum

s/\\\\*/[0-9a-z\_.+-]\*/g; # Unquote any '\*' characters as they map to .\*

# Find all the numbers just before the @ and replace with them digit wildcards

s/([0-9a-z\_.+-])\d{1,3}\\\@/$1\\d+\\@/i;

#push @quoted, '(' . $\_ . ')';

push @quoted, $\_;

$count++;

if ($count % $addresses\_per\_rule == 0) {

# Put them in 10 addresses at a time

$rule\_num++;

# Put a start-of-line/non-address character at the front,

# and an end-of-line /non-address character at the end.

print SACF "header \_\_SCAMNAILER\_H$rule\_num ALL =~ /" .

'(^|[;:\s])(?:' . join('|',@quoted) . ')($|[^0-9a-z\_.+-])' .

"/i\n";

push @metarules, "\_\_SCAMNAILER\_H$rule\_num";

print SACF "uri \_\_SCAMNAILER\_B$rule\_num /" .

'^mailto:(?:' . join('|',@quoted) . ')$' .

"/i\n";

push @metarules, "\_\_SCAMNAILER\_B$rule\_num";

undef @quoted;

undef @addresses;

}

}

close PHISHIN;

# Put in all the leftovers, if any

if (@quoted) {

$rule\_num++;

print SACF "header \_\_SCAMNAILER\_H$rule\_num ALL =~ /" .

'(^|[;:\s])(?:' . join('|',@quoted) . ')($|[^0-9a-z\_.+-])' .

"/i\n";

push @metarules, "\_\_SCAMNAILER\_H$rule\_num";

print SACF "uri \_\_SCAMNAILER\_B$rule\_num /" .

'^mailto:(?:' . join('|',@quoted) . ')$' .

"/i\n";

push @metarules, "\_\_SCAMNAILER\_B$rule\_num";

}

print SACF "\n# ScamNailer combination rule\n\n";

print SACF "meta SCAMNAILER " . join(' || ',@metarules) . "\n";

print SACF "describe SCAMNAILER Mentions a spear-phishing address\n";

print SACF "score SCAMNAILER $SA\_score\n\n";

print SACF "# ScamNailer rules ($count) END\n";

close SACF;

# And finally restart MailScanner to use the new rules

$mailscanner\_restart .= " >/dev/null 2>&1" if $quiet;

system($mailscanner\_restart) if $mailscanner\_restart;

exit 0;

}

sub GetPhishingUpdate {

my $cache = $emailscurrent . 'cache/';

my $status = $emailscurrent . 'status';

my $urlbase = "http://www.mailscanner.tv/emails.";

my $target= $emailscurrent . 'phishing.emails.list';

my $query="emails.msupdate.greylist.bastionmail.com";

my $baseupdated = 0;

if (! -d $emailscurrent) {

print "Working directory is not present - making....." unless $quiet;

mkdir ($emailscurrent) or die "failed";

print " ok!\n" unless $quiet;

}

if (! -d $cache) {

print "Cache directory is not present - making....." unless $quiet;

mkdir ($cache) or die "failed";

print " ok!\n" unless $quiet;

}

if (! -s $target) {

open (FILE,">$target") or die

"Failed to open target file so creating a blank file";

print FILE "# Wibble";

close FILE;

} else {

# So that clean quarantine doesn't delete it!

utime(time(), time(), $emailscurrent);

}

my ($status\_base, $status\_update);

$status\_base=-1;

$status\_update=-1;

if (! -s $status) {

print "This is the first run of this program.....\n" unless $quiet;

} else {

print "Reading status from $status\n" unless $quiet;

open(STATUS\_FILE, $status) or die "Unable to open status file\n";

my $line=<status\_file>;

close (STATUS\_FILE);

# The status file is text.text

if ($line =~ /^(.+)\.(.+)$/) {

$status\_base=$1;

$status\_update=$2;

}

}

print "Checking that $cache$status\_base exists..." unless $quiet;

if ((! -s "$cache$status\_base") && (!($status\_base eq "-1"))) {

print " no - resetting....." unless $quiet;

$status\_base=-1;

}

print " ok\n" unless $quiet;

print "Checking that $cache$status\_base.$status\_update exists..." unless $quiet;

if ((! -s "$cache$status\_base.$status\_update") && ($status\_update>0)) {

print " no - resetting....." unless $quiet;

$status\_update=-1;

}

print " ok\n" unless $quiet;

my $currentbase = -1;

my $currentupdate = -1;

# Lets get the current version

my $res = Net::DNS::Resolver->new();

my $RR = $res->query($query, 'TXT');

my @result;

if ($RR) {

foreach my $rr ($RR->answer) {

my $text = $rr->rdatastr;

if ($text =~ /^"emails\.(.+)\.(.+)"$/) {

$currentbase=$1;

$currentupdate=$2;

last;

}

}

}

die "Failed to retrieve valid current details\n" if $currentbase eq "-1";

print "I am working with: Current: $currentbase - $currentupdate and Status: $status\_base - $status\_update\n" unless $quiet;

my $generate=0;

# Create a user agent object

my $ua = LWP::UserAgent->new;

$ua->agent("UpdateBadPhishingSites/0.1 ");

# Patch from Heinz.Knutzen@dataport.de

$ua->env\_proxy;

if (!($currentbase eq $status\_base)) {

print "This is base update\n" unless $quiet;

$status\_update = -1;

$baseupdated = 1;

# Create a request

#print "Getting $urlbase . $currentbase\n" unless $quiet;

my $req = HTTP::Request->new(GET => $urlbase.$currentbase);

# Pass request to the user agent and get a response back

my $res = $ua->request($req);

# Check the outcome of the response

if ($res->is\_success) {

open (FILE, ">$cache/$currentbase") or die "Unable to write base file ($cache/$currentbase)\n";

print FILE $res->content;

close (FILE);

} else {

warn "Unable to retrieve $urlbase.$currentbase :".$res->status\_line, "\n";

}

$generate=1;

} else {

print "No base update required\n" unless $quiet;

}

# Now see if the sub version is different

if (!($status\_update eq $currentupdate)) {

my %updates=();

print "Update required\n" unless $quiet;

if ($currentupdate‹$status\_update) {

# In the unlikely event we roll back a patch - we have to go from the base

print "Error!: $currentupdate<$status\_update\n" unless $quiet;

$generate = 1;

$status\_update = 0;

}

# If there are updates avaliable and we haven't donloaded them

# yet we need to reset the counter

if ($currentupdate>0) {

if ($status\_update<1) {

$status\_update=0;

}

my $i;

# Loop through each of the updates, retrieve it and then add

# the information into the update array

for ($i=$status\_update+1; $i<=$currentupdate; $i++) {

print "Retrieving $urlbase$currentbase.$i\n" unless $quiet;

#print "Getting $urlbase . $currentbase.$i\n" unless $quiet;

my $req = HTTP::Request->new(GET => $urlbase.$currentbase.".".$i);

my $res = $ua->request($req);

warn "Failed to retrieve $urlbase$currentbase.$i"

unless $res->is\_success;

my $line;

foreach $line (split("\n", $res->content)) {

# Is it an addition?

if ($line =~ /^\> (.+)$/) {

if (defined $updates{$1}) {

if ($updates{$1} eq "<") {

delete $updates{$1};

}

} else {

$updates{$1}=">";

}

}

# Is it an removal?

if ($line =~ /^\< (.+)$/) {

if (defined $updates{$1}) {

if ($updates{$1} eq ">") {

delete $updates{$1};

}

} else {

$updates{$1}="<";

}

}

}

}

# OK do we have a previous version to work from?

if ($status\_update>0) {

# Yes - we open the most recent version

open (FILE, "$cache$currentbase.$status\_update") or die

"Unable to open base file ($cache/$currentbase.$status\_update)\n";

} else { # No - we open the the base file

open (FILE, "$cache$currentbase") or die

"Unable to open base file ($cache/$currentbase)\n";

}

# Now open the new update file

print "$cache$currentbase.$currentupdate\n" unless $quiet;

open (FILEOUT, ">$cache$currentbase.$currentupdate") or die

"Unable to open new base file ($cache$currentbase.$currentupdate)\n";

# Loop through the base file (or most recent update)

while (<file>) {

chop;

my $line=$\_;

if (defined ($updates{$line})) {

# Does the line need removing?

if ($updates{$line} eq "<") {

$generate=1;

next;

}

# Is it marked as an addition but already present?

elsif ($updates{$line} eq ">") {

delete $updates{$line};

}

}

print FILEOUT $line."\n";

}

close (FILE);

my $line;

# Are there any additions left

foreach $line (keys %updates) {

if ($updates{$line} eq ">") {

print FILEOUT $line."\n" ;

$generate=1;

}

}

close (FILEOUT);

}

}

# Changes have been made

if ($generate) {

print "Updating live file $target\n" unless $quiet;

my $file="";

if ($currentupdate>0) {

$file="$cache/$currentbase.$currentupdate";

} else {

$file="$cache/$currentbase";

}

if ($file eq "") {

die "Unable to work out file!\n";

}

system ("mv -f $target $target.old");

system ("cp $file $target");

open(STATUS\_FILE, ">$status") or die "Unable to open status file\n";

print STATUS\_FILE "$currentbase.$currentupdate\n";

close (STATUS\_FILE);

}

my $queuedir = new DirHandle;

my $file;

my $match1 = "^" . $currentbase . "\$";

my $match2 = "^" . $currentbase . "." . $currentupdate . "\$";

$queuedir->open($cache) or die "Unable to do clean up\n";

while(defined($file = $queuedir->read())) {

next if $file eq '.' || $file eq '..';

next if $file =~ /$match1/;

next if $file =~ /$match2/;

print "Deleting cached file: $file.... " unless $quiet;

unlink($cache.$file) or die "failed";

print "ok\n" unless $quiet;

}

$queuedir->close();

$generate;

}

1. Press **Esc** to exit to command mode
2. Save and quit vi with
3. *:wq* (write then quit)
4. Make the script executable with *chmod +x /opt/MailScanner/bin/update\_scamnailer*

# Protect SpamSnake

Firehol is a stateful iptables packet filtering firewall configurator. It is abstracted, extensible, easy and powerful. It can handle any kind of firewall, but most importantly, it gives you the means to configure it, the same way you think of it.

## Install Firehol

1. Install Firehol with *apt-get install firehol -y*
2. Open the startup file with *vi /etc/default/firehol*
3. Enter vi edit mode by pressing **I** (Insert)
4. Change the following in the file

START\_FIREHOL=YES

1. Press **Esc** to exit to command mode
2. Save and quit vi with
3. *:wq* (write then quit)

## Create Update Scripts

1. Create IANA Update script with *vi /usr/sbin/get-iana*
2. Enter vi edit mode by pressing **I** (Insert)
3. Populate the file with

#!/bin/bash

# $Id: get-iana.sh,v 1.13 2010/09/12 13:55:00 jcb Exp $

#

# $Log: get-iana.sh,v $

# Revision 1.13 2010/09/12 13:55:00 jcb

# Updated for latest IANA reservations format.

#

# Revision 1.12 2008/03/17 22:08:43 ktsaou

# Updated for latest IANA reservations format.

#

# Revision 1.11 2007/06/13 14:40:04 ktsaou

# \*\*\* empty log message \*\*\*

#

# Revision 1.10 2007/05/05 23:38:31 ktsaou

# Added support for external definitions of:

#

# RESERVED\_IPS

# PRIVATE\_IPS

# MULTICAST\_IPS

# UNROUTABLE\_IPS

#

# in files under the same name in /etc/firehol/.

# Only RESERVED\_IPS is mandatory (firehol will complain if it is not there,

# but it will still work without it), and is also the only file that firehol

# checks how old is it. If it is 90+ days old, firehol will complain again.

#

# Changed the supplied get-iana.sh script to generate the RESERVED\_IPS file.

# FireHOL also instructs the user to use this script if the file is missing

# or is too old.

#

# Revision 1.9 2007/04/29 19:34:11 ktsaou

# \*\*\* empty log message \*\*\*

#

# Revision 1.8 2005/06/02 15:48:52 ktsaou

# Allowed 127.0.0.1 to be in RESERVED\_IPS

#

# Revision 1.7 2005/05/08 23:27:23 ktsaou

# Updated RESERVED\_IPS to current IANA reservations.

#

# Revision 1.6 2004/01/10 18:44:39 ktsaou

# Further optimized and reduced PRIVATE\_IPS using:

# http://www.vergenet.net/linux/aggregate/

#

# The supplied get-iana.sh uses .aggregate. if it finds it in the path.

# (aggregate is the name of this program when installed on Gentoo)

#

# Revision 1.5 2003/08/23 23:26:50 ktsaou

# Bug #793889:

# Change #!/bin/sh to #!/bin/bash to allow FireHOL run on systems that

# bash is not linked to /bin/sh.

#

# Revision 1.4 2002/10/27 12:44:42 ktsaou

# CVS test

#

#

# Program that downloads the IPv4 address space allocation by IANA

# and creates a list with all reserved address spaces.

#

IPV4\_ADDRESS\_SPACE\_URL="http://www.iana.org/assignments/ipv4-address-space/ipv4-address-space.txt"

# The program will match all rows in the file which start with a number, have a slash,

# followed by another number, for which the following pattern will also match on the

# same rows

IANA\_RESERVED="(RESERVED|UNALLOCATED)"

# which rows that are matched by the above, to ignore

# (i.e. not include them in RESERVED\_IPS)?

#IANA\_IGNORE="(Multicast|Private use|Loopback|Local Identification)"

IANA\_IGNORE="Multicast"

tempfile="/tmp/iana.$$.$RANDOM"

AGGREGATE="`which aggregate 2>/dev/null`"

if [ -z "${AGGREGATE}" ]

then

AGGREGATE="`which aggregate 2>/dev/null`"

fi

if [ -z "${AGGREGATE}" ]

then

echo >&2

echo >&2

echo >&2 "WARNING"

echo >&2 "Please install 'aggregate' to shrink the list of IPs."

echo >&2

echo >&2

fi

echo >&2

echo >&2 "Fetching IANA IPv4 Address Space, from:"

echo >&2 "${IPV4\_ADDRESS\_SPACE\_URL}"

echo >&2

wget -O - -proxy=off "${IPV4\_ADDRESS\_SPACE\_URL}" |\

egrep " \*[0-9]+/[0-9]+.\*${IANA\_RESERVED}" |\

egrep -vi "${IANA\_IGNORE}" |\

sed -e 's:^ \*\([0-9]\*/[0-9]\*\).\*:\1:' |\

(

while IFS="/" read range net

do

if [ ! $net -eq 8 ]

then

echo >&2 "Cannot handle network masks of $net bits ($range/$net)"

continue

fi

first=`echo $range | cut -d '-' -f 1`

first=`expr $first + 0`

last=`echo $range | cut -d '-' -f 2`

last=`expr $last + 0`

x=$first

while [ ! $x -gt $last ]

do

# test $x -ne 127 && echo "$x.0.0.0/$net"

echo "$x.0.0.0/$net"

x=$[x + 1]

done

done

) | \

(

if [ ! -z "${AGGREGATE}" -a -x "${AGGREGATE}" ]

then

"${AGGREGATE}"

else

cat

fi

) >"${tempfile}"

echo >&2

echo >&2

echo >&2 "FOUND THE FOLLOWING RESERVED IP RANGES:"

printf "RESERVED\_IPS=\""

i=0

for x in `cat ${tempfile}`

do

i=$[i + 1]

printf "${x} "

done

printf "\"\n"

if [ $i -eq 0 ]

then

echo >&2

echo >&2

echo >&2 "Failed to find reserved IPs."

echo >&2 "Possibly the file format has been changed, or I cannot fetch the URL."

echo >&2

rm -f ${tempfile}

exit 1

fi

echo >&2

echo >&2

echo >&2 "Differences between the fetched list and the list installed in"

echo >&2 "/etc/firehol/RESERVED\_IPS:"

echo >&2 "# diff /etc/firehol/RESERVED\_IPS ${tempfile}"

diff /etc/firehol/RESERVED\_IPS ${tempfile}

if [ $? -eq 0 ]

then

echo >&2

echo >&2 "No differences found."

echo >&2

rm -f ${tempfile}

exit 0

fi

echo >&2

echo >&2

echo >&2 "Would you like to save this list to /etc/firehol/RESERVED\_IPS"

echo >&2 "so that FireHOL will automatically use it from now on?"

echo >&2

while [ 1 = 1 ]

do

printf >&2 "yes or no > "

read x

case "${x}" in

yes) cp -f /etc/firehol/RESERVED\_IPS /etc/firehol/RESERVED\_IPS.old 2>/dev/null

cat "${tempfile}" >/etc/firehol/RESERVED\_IPS || exit 1

echo >&2 "New RESERVED\_IPS written to '/etc/firehol/RESERVED\_IPS'."

echo "Firehol will now be restart"

sleep 3

/etc/init.d/firehol restart

break

;;

no)

echo >&2 "Saved nothing."

break

;;

\*) echo >&2 "Cannot understand '${x}'."

;;

esac

done

rm -f ${tempfile}

1. Press **Esc** to exit to command mode
2. Save and quit vi with
3. *:wq* (write then quit)
4. Make the executable with *chmod +x /usr/sbin/get-iana*
5. Create update command with *vi /usr/sbin/update-iana*
6. Enter vi edit mode by pressing **I** (Insert)
7. Populate the file with

#!/bin/sh

/usr/sbin/get-iana < /etc/firehol/get-iana-answerfile

1. Press **Esc** to exit to command mode
2. Save and quit vi with
3. *:wq* (write then quit)
4. Make the script executable with *chmod +x /usr/sbin/update-iana*
5. Create update answer file with *vi /etc/firehol/get-iana-answerfile*
6. Enter vi edit mode by pressing **I** (Insert)
7. Populate the file with

yes

1. Press **Esc** to exit to command mode
2. Save and quit vi with
3. *:wq* (write then quit)

## Configure Firehol

1. Edit the config file with *vi /etc/firehol/firehol.conf*
2. Enter vi edit mode by pressing **I** (Insert)
3. Update the file with the following

version 5

# Accept all client traffic on any interface

interface any internet

protection strong

server "icmp ping ICMP ssh http https telnet webmin dns dcc echo smtp" accept

client all accept

1. Press **Esc** to exit to command mode
2. Save and quit vi with
3. *:wq* (write then quit)

## Updating Software

Once the install is complete it is a good idea to use Webmin to perform all the security and application updates on the filter.

# Branding

To make sure every box looks and talks the same we need to update some files by uploading then to the file system directly via Webmin, by doing this we can provide users with a wonderful look and correctly talking messages.

Go to Webmin

Expand Others

Open File Manager

Upload the files within the branding folder to the same locations on SpamSnake.

(MailScanner Reports, delete all and upload + Extract Reports.zip)

# Scheduling Updates and Tasks

This section of the guide describes how to

Execute As: root

Active: Yes

Command: /opt/MailScanner/bin/update\_phishing\_sites &> /dev/null

Decription: #Update MailScanner Phishing Sites

Times: 37 minutes, 5 Hours, All Days, All Months, All Weekdays

Execute As: root

Active: Yes

Command: /opt/MailScanner/bin/update\_bad\_phishing\_sites &> /dev/null

Decription: #Update MailScanner Bad Phishing Sites

Times: 7 minutes, All Hours, All Days, All Months, All Weekdays

Execute As: root

Active: Yes

Command: /opt/MailScanner/bin/clean.quarantine &> /dev/null

Decription: #Clean MailScanner Quarantine

Times: 58 minutes, 23 Hours, All Days, All Months, All Weekdays

Execute As: root

Active: Yes

Command: /opt/MailScanner/bin/update\_virus\_scanners &> /dev/null

Decription: #Update Virus Scanners

Times: 42 minutes, All Hours, All Days, All Months, All Weekdays

Execute As: root

Active: Yes

Command: /opt/MailScanner/bin/check\_mailscanner &> /dev/null

Decription: N/A

Times: 3,23,43 minutes, All Hours, All Days, All Months, All Weekdays

Execute As: root

Active: Yes

Command: /usr/bin/sa-learn --force-expire --sync –p /opt/MailScanner/etc/spam.assassin.prefs.conf

Decription: #Spamassassin Learning

Times: 30 minutes, 01 Hours, All Days, All Months, All Weekdays

Execute As: root

Active: Yes

Command: manage.py cleanquarantine &> /dev/null

Decription: #Clean quarantine

Times: Daily

Execute As: root

Active: Yes

Command: manage.py sendquarantinereports &> /dev/null

Decription: #Send quarantine reports

Times: Daily

Execute As: root

Active: Yes

Command: manage.py dbclean &> /dev/null

Decription: #Clean maillog

Times: Monthly

Execute As: root

Active: Yes

Command: manage.py updatesarules &> /dev/null

Decription: #Update spamassassin rules

Times: Weekly

Execute As: root

Active: No

Command: manage.py sendpdfreports &> /dev/null

Decription: #Send PDF Reports

Times: Daily

Execute As: root

Active: Yes

Command: /usr/sbin/fuzzy-cleanmysql &> /dev/null

Decription: #FuzzyOcr DB cleaner

Times: Weekly

Execute As: root

Active: Yes

Command: /usr/sbin/clamav-unofficial-sigs.sh -c /etc/clamav-unofficial-sigs.conf &> /dev/null

Decription: N/A

Times: 00 minutes, 04 Hours, All Days, All Months, All Weekdays

Execute As: root

Active: Yes

Command: /usr/sbin/update\_scamnailer &> /dev/null

Decription: #Update Scamnailer

Times: Daily

Execute As: root

Active: Yes

Command: /usr/sbin/update-iana &> /dev/null

Decription: #Update firehol reserved ips

Times: Monthly

<http://www.mailscanner.info/tnef.html>

Rich Text scanning issue