HOW TO STOP SPAM – DEAD IN ITS TRACKS!!

(WHM®/cPanel® 11.48.1 – stock configuration)

For years I suffered and tolerated out of control spam. I tried tinkering with some spam assassin (user_prefs) settings and RBL's in the Exim Configuration Manager, but with very limited success. I even considered purchasing a managed spam control system; but why should we pay for a 3rd party solution when we're already paying for cPanel which includes fully integrated SpamAssassin … which by the way is really supposed to be pretty good? So I finally decided to roll up my sleeves and really tackle the problem.

RECOMMENDED READING:
(giving credit where credit is due! ☺)

Click here to skip to WHM/cPanel Settings …

1. Building a Poor Man's Barracuda Spam Firewall: Most of what I implemented came from this guide; in fact, without this guide I may have never got started on this project. It took some time to work through and thoroughly understand each step. In this guide, I will provide a summary list of settings, organized in a way that hopefully can be quickly and easily entered in WHM/cPanel.

2. Access Control Lists (ACL's): The syntax for how ACL's work in Exim is not intuitive at all. It's even more difficult to grasp without some basic understanding of how the SMTP protocol works. I liked this particular resource because it gives a fairly detailed explanation of several ACL rules. In a nutshell, a rule consists a "verb" and associated "modifiers" and "conditions". NOTE: all modifiers and conditions that follow a verb, are associated with that verb … until a new verb is encountered.

3. Simple Greylisting in Exim: The greylisting system that I developed more or less follows the logic in this example. I really liked most of the features and its overall implementation scheme in general. The problem, though, with this particular example is that it requires that Exim be built from source with the dbm library included, which in cPanel it is not. So, although I followed the logic in this example, I implemented it instead with Perl because I am not interested in having to rebuild Exim from source every time cPanel updates.

4. Simple Greylisting in Perl: Fortunately, the cPanel stock Exim build does in fact include a Perl interpreter; so we can use Perl without having to rebuild Exim from source. But as I have literally zero experience with Perl, I found the greylisting section of this guide to be invaluable. I basically copied most of the syntax, and then completely re-wrote the subroutines to follow the logic in the example in item no. 3 above. I also chose to keep the database(s) in MySQL, as in this example, versus SQLite because I like the fact that I can use phpMyAdmin to monitor the databases while debugging my code.

5. Looking Up MX Records in Perl: In general, I found myself over at the perlmonks.org website quite a lot as I learned my way around the Perl scripting language. The greylisting system that I developed, includes "automatic" whitelisting (exemption from greylisting) for mailservers belonging to email recipients of my users' outbound email. This snippet was extremely helpful since I know practically nothing about how DNS query or formatted or returned.

6. Exim Specification - Table of Contents: I think you could spend an entire month reading the documents at this link. For this project, I found the details in chapter 11 (String Expansions) and chapter 42 (Access Control Lists) to be particularly helpful.

7. Apache SpamAssassin v3.3.x - Default Test Scores: This page contains a VERY comprehensive list of SpamAssassin default scores for hundereds of tests that are performed on each message. And of course, the default scores can be changed for each user account as desired by editing ~/.spamassassin/user_prefs, or for the entire server by editing /etc/mail/spamassassin/local.cf. One way to see which tests might be of interest to you is to examine the bottom of the headers of particular email messages to see which tests (and scores) are being triggered in your mail. At the time that this guide is written, the current version of SpamAssassin is 3.4.0, but I could only find this list for version 3.3.x.

8. Nolisting: After implementing all of the above, I was quite happy with the spam across the entire server being reduced to pretty much zero. For a short while after, I routinely monitored the exim_mainlog, exim_greylistlog, greylist database, and mail delivery reports to make sure that everything seemed to be working fine. It was interesting to see in realtime all of the traffic coming to the mail server, and the significant amount of activity associated with catching the spam. And then while writing this document, I stumbled across “Nolisting” and was immediately fascinated with the concept. Knownhost gives us 2 dedicated IP numbers, so I decided to give this a try! I will update this document with stats later: summary of rejection at SMTP and summary of greylisting spam (before/after nolisting).

Server Settings and Modifications

(WHM®/cPanel® 11.48.1 – stock configuration)

Rather than copy/paste code in this document, feel free to get the files here: Mailserver Setup Files

I've tried not to clutter this section with too much explanation. For simplicity, actual settings or modifications to be implemented are indicated in red below. For those who are interested, more verbose explanation and reasoning for each step is provided at the Appendix near the end of this document.

1. Install Clam AntiVirus (ClamAV®):

   In WHM » Home » cPanel » Manage Plugins, check the box "Install and keep updated" next to ClamAV, and then click "Save". This not only installs ClamAV, but it also configures Exim to use it.

2. Setup and Configure a Caching Nameserver:

   In my setup, I estimate that about 70% or so of the spam that hits my server gets rejected at SMTP time (by Exim) because either the sender ip number, sender domain, or a message body domain link is found in a DNS block list (RBL). This is very important: DNS block list (RBL) lookups will NOT work if you do not have a properly configured caching nameserver!

   – In WHM » Home » Service Configuration » Nameserver Selection, select the radio button next to "BIND", and then click the "Save" button.

   – In WHM » Home » Network Setup » Resolver Configuration, click the "Proceed" button to get to the nameserver input screen. Set the "Primary Resolver" to the localhost, 127.0.0.1, and the others to whatever you prefer (KnownHost default is to use Google servers 8.8.8.8 and 8.8.4.4). Then press the "Continue Button".

   – The stock cPanel BIND version tries to use IPv6 transport, even though the server host does not have IPv6 connectivity, thus resulting in failed DNS block (RBL) list lookups, come-on cPanel, seriously?!

   Set BIND to use IPv4 only (no IPv6). Add the following line to the file '/etc/sysconfig/named':

   Code:

   ```
   OPTIONS="-4"
   ```

   – Make sure that the Net::DNS Perl module is installed. I think it would be very unusual if it weren't, and probably other important things would be broken. But just in case …

   In WHM » Home » Software » Install a Perl Module, scroll down and find "Net::DNS" (mine is version 0.83). IF it's not installed, then click on the "Show Available Perl Module(s)" button near the top of the page, and then click on the "Install" link next to Net::DNS in the list.

3. Exim Configuration – Basic Settings

In WHM » Home » Service Configuration » Exim Configuration Manager, select the [Basic Editor] tab, and edit the following settings in the various sub-tabs.

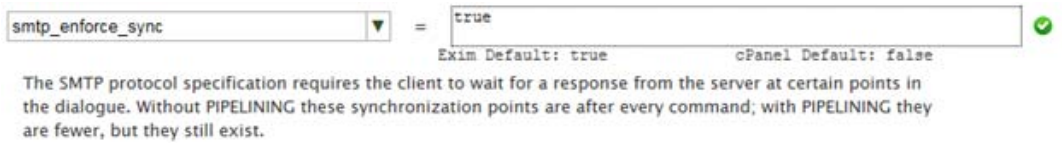
ACL Options	
SpamAssassin reject spam score threshold	10
Dictionary attack protection	On
Reject remote mail sent to server's hostname	On
Ratelimit suspicious SMTP servers	On
SpamAssassin: ratelimit spam score threshold	5
Require HELO before MAIL	On
Require remote (hostname/IP) HELO	On
Require remote (domain) HELO	On
Require RFC-compliant HELO	On
Max msg recipients (soft limit)	25
Max msg recipients before disconnect (hard limit)	25
Filters	
System filter file	/etc/cpanel_exim_system_filter (*default*)
Attachments: Filter msgs w/dangerous attachments	On
Mail	
Log sender rates in the exim mainlog	On
Sender verification callouts	Off (Setting On can get you blacklisted)
Sender verification	On
Security	
Require clients to connect with SSL or issue the STARTTLS command before they are allowed to authenticate with the server	On
Apache SpamAssassin™ Options	
Apache SpamAssassin Forced Global ON	On

NOTE: A detailed explanation and reasoning for each setting is given at Appendix A, near the end of this document.

4. Exim Configuration – Advanced Settings (delay and sync)

In WHM » Home » Service Configuration » Exim Configuration Manager, select the [Advanced Editor] tab, and then make the following changes:

– Search for the “smtp_enforce_sync” in the “Combined Exim Configuration” block and changed its value = true.



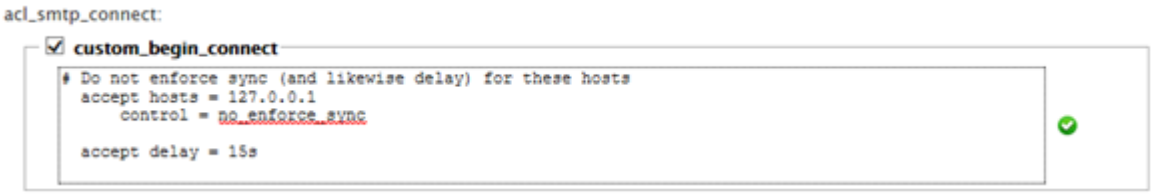
– Search for the “custom_begin_connect” block in the acl_smtp_connect ACL and add the following rule(s):

Code:

```
# Do not enforce sync (and likewise delay) for these hosts
  accept hosts = 127.0.0.1
       control = no_enforce_sync

  accept delay = 15s
```

It should look like this:



Scroll all the way down to the bottom of the Advanced Editor window/frame, and click on the “Save” button.

5. Tweak Settings

In WHM » Home » Server Configuration » Tweak Settings, select the [Mail] tab

– On the first line, change “Initial default/catch-all forwarder destination” to “Fail”.

– At the 5th line from the bottom, change “Enable Apache SpamAssassin™ Spam Box delivery for messages marked as spam (user configurable) to “On”.

6. Exim Configuration – Advanced Settings - RBL’s (DNS blocklists)

You may be aware already that the Exim Configuration Manager’s [Basic Editor] tab has a [RBLs] sub-tab. For reasons described in the Appendix, I do not manage my RBL’s there; instead …

In WHM » Home » Service Configuration » Exim Configuration Manager, select [Advanced Editor] tab; then search for the “custom_begin_rbl” block in the acl_smtp_rcpt ACL and add the following rule(s):

Code:

```
# spamhaus zen
  deny message = JunkMail rejected - $sender_fullhost - $sender_address_domain - is in the $dnslist_domain dnsbl, see: $dnslist_text
       hosts = +backupmx_hosts
       dnslists = zen.spamhaus.org
  warn dnslists = zen.spamhaus.org
       set acl_m8 = 1
       set acl_m9 = "JunkMail rejected - $sender_fullhost - $sender_address_domain - is in the $dnslist_domain dnsbl, see: $dnslist_text"
  warn condition = ${if eq {${acl_m8}}{1}{1}{0}}
       ratelimit = 0 / 1h / strict / per_conn
       log_message = "Increment Connection Ratelimit - $sender_fullhost - $sender_address_domain - because of $dnslist_domain match"
  drop condition = ${if eq {${acl_m8}}{1}{1}{0}}
       message = ${acl_m9}

# spamhaus DBL - Domain name blocking list
  deny message = JunkMail rejected - $sender_fullhost - $sender_address_domain - is listed on Spamhaus DBL. see: $dnslist_text
       hosts = +backupmx_hosts
       dnslists = dbl.spamhaus.org/<,$sender_address_domain
  warn dnslists = dbl.spamhaus.org/<,$sender_address_domain
       set acl_m8 = 1
       set acl_m9 = "JunkMail rejected - $sender_fullhost - $sender_address_domain - is in $dnslist_domain, see: $dnslist_text"
  warn condition = ${if eq {${acl_m8}}{1}{1}{0}}
       ratelimit = 0 / 1h / strict / per_conn
       log_message = "Increment Connection Ratelimit - $sender_fullhost - $sender_address_domain - because of $dnslist_domain dnsbl match"
  drop condition = ${if eq {${acl_m8}}{1}{1}{0}}
       message = ${acl_m9}

# spamcop
  deny message = JunkMail rejected - $sender_fullhost - $sender_address_domain - is in the $dnslist_domain dnsbl, see: $dnslist_text
       hosts = +backupmx_hosts
       dnslists = bl.spamcop.net
  warn dnslists = bl.spamcop.net
       set acl_m8 = 1
       set acl_m9 = "JunkMail rejected - $sender_fullhost - $sender_address_domain - is in the $dnslist_domain dnsbl, see: $dnslist_text"
  warn condition = ${if eq {${acl_m8}}{1}{1}{0}}
       ratelimit = 0 / 1h / strict / per_conn
       log_message = "Increment Connection Ratelimit - $sender_fullhost - $sender_address_domain - because of $dnslist_domain match"
  drop condition = ${if eq {${acl_m8}}{1}{1}{0}}
       message = ${acl_m9}

# URIBL - black
  deny message = JunkMail rejected - $sender_fullhost - $sender_address_domain - is in the $dnslist_domain dnsbl, see: $dnslist_text
       hosts = +backupmx_hosts
       dnslists = black.uribl.com/<,$sender_address_domain
  warn dnslists = black.uribl.com/<,$sender_address_domain
       set acl_m8 = 1
       set acl_m9 = "JunkMail rejected - $sender_fullhost - $sender_address_domain - is in $dnslist_domain, see: $dnslist_text"
  warn condition = ${if eq {${acl_m8}}{1}{1}{0}}
       ratelimit = 0 / 1h / strict / per_conn
       log_message = "Increment Connection Ratelimit - $sender_fullhost - $sender_address_domain - because of $dnslist_domain dnsbl match"
  drop condition = ${if eq {${acl_m8}}{1}{1}{0}}
       message = ${acl_m9}

# barracudacentral
  deny message = JunkMail rejected - $sender_fullhost - $sender_address_domain - is in the $dnslist_domain dnsbl, see: http://www.barracudacentral.org/rbl
       hosts = +backupmx_hosts
       dnslists = bb.barracudacentral.org
  warn dnslists = bb.barracudacentral.org
       set acl_m8 = 1
       set acl_m9 = "JunkMail rejected - $sender_fullhost - $sender_address_domain - is in the $dnslist_domain dnsbl, see: http://www.barracudacentral.org/rbl"
  warn condition = ${if eq {${acl_m8}}{1}{1}{0}}
       ratelimit = 0 / 1h / strict / per_conn
       log_message = "Increment Connection Ratelimit - $sender_fullhost - $sender_address_domain - because of $dnslist_domain match"
```

```
  drop condition = ${if eq {${acl_m8}}{1}{1}{0}}
      message = ${acl_m9}

# SpamRATS - NoPtr
  deny message = JunkMail rejected - $sender_fullhost - $sender_address_domain - is in the $dnslist_domain dnsbl, see: $dnslist_text
      hosts = +backupmx_hosts
      dnslists = noptr.spamrats.com
  warn dnslists = noptr.spamrats.com
      set acl_m8 = 1
      set acl_m9 = "JunkMail rejected - $sender_fullhost - $sender_address_domain - is in $dnslist_domain, see: $dnslist_text"
  warn condition = ${if eq {${acl_m8}}{1}{1}{0}}
      ratelimit = 0 / 1h / strict / per_conn
      log_message = "Increment Connection Ratelimit - $sender_fullhost - $sender_address_domain - because of $dnslist_domain dnsbl match"
  drop condition = ${if eq {${acl_m8}}{1}{1}{0}}
      message = ${acl_m9}
```

Scroll all the way down to the bottom of the Advanced Editor window/frame, and click on the "Save" button.

To date with these DNS block lists (RBL), I have ZERO false positives. Since the messages are "rejected" at SMTP time, none of these messages are delivered to a spam or junkmail folder; and with rejection bounce-back, hopefully some recipients are removed from a few spam lists (they have to clean-up their lists occasionally right?). To review the rejected mail notices, look in WHM » Home » Email » Mail Delivery Reports, click the "Run Report" button, click on the "Advanced Search" link next to the search box, and then uncheck the box next to "Show Deliveries".

7. SpamAssassin – Install and Enable DCC, Pyzor, Razor2, and iXhash Modules

– Open outbound ports for DCC, Pyzor, and Razor2. In WHM » Home » Plugins » ConfigServer Security & Firewall, click on the "Firewall Configuration" button. Then select "IPv4 Port Settings in the dropdown selector near the top of the page and add ports as follows:

Razor2:	port 2703	TCP_OUT
DCC	port 6277	UDP_OUT
Pyzor	port 24441	UDP_OUT

Scroll down to the bottom of the page and click on the "Change" button. Then click on the "Restart csf+lfd" button.

– Download and install DCC. At the time that this guide is written, the current version of DCC is 1.3.155. If the version is changes, then change the directory name in line 4 below as required. In an SSH terminal as root user, enter the following:

Code:

```
cd /usr/local/src
wget http://www.rhyolite.com/dcc/source/dcc.tar.Z
tar -zxvf dcc.tar.Z
cd dcc-1.3.155
./configure
make
make install
```

- Download and install Pyzor. At the time that this guide is written, the current version of Pyzor is 0.7.0. If a newer version is available, change the directory/filename(s) in line 3 and 5 below as required. In an SSH terminal as root user, enter the following:

  Code:

```
yum install python-setuptools
cd /usr/local/src
wget http://sourceforge.net/projects/pyzor/files/pyzor/0.7.0/pyzor-0.7.0.tar.bz2
tar -zxvf pyzor*.tar.gz
cd pyzor-0.7.0
python setup.py build
python setup.py install
/usr/bin/pyzor discover
/usr/bin/pyzor ping        # This tests that Pyzor can connect to its server
                           # Should return "public.pyzor.org:24441  (200, 'OK')"
```

- Download and install Razor2. At the time that this guide is written, the current version of Razor2 is 2.84. If a newer version is available, change the directory/filename(s) in line 2 and 4 below as required. In an SSH terminal as root user, enter the following:

  Code:

```
cd /usr/local/src
wget http://sourceforge.net/projects/razor/files/razor-agents/2.84/razor-agents-2.84.tar.bz2
tar -jxvf razor-agents*.bz2
cd razor-agents-2.84
perl Makefile.PL
make
make test
make install
```

- Download and install iXhash. At the time that this guide is written, the current version of iXhash is 1.5.5. If a newer version is available, change the directory/filename(s) in line 2 and 4 below. In an SSH terminal as root user, enter the following:

  Code:

```
cd /usr/local/src
wget http://sourceforge.net/projects/ixhash/files/iXhash/iXhash-1.5.5/iXhash-1.5.5.tgz
tar -zxvf  iXhash*.tgz
cd iXhash-1.5.5
cp iXhash/iXhash.cf /etc/mail/spamassassin
cp iXhash/iXhash.pm /etc/mail/spamassassin
/usr/local/cpanel/3rdparty/bin/spamassassin -D IXHASH < iXhash.eml  # This tests iXhash
                                                                   # Should list an IXHASH
                                                                   # test(GENERIC)in the
                                                                   # X-Spam-Status
```

- Edit the iXhash configuration file. Comment out ctyme.ixhash.net config lines in the /etc/mail/spamassassin/iXhash.cf configuration file:

  Code:

```
…
# ctyme.ixhash.net seems to be down (03/2015), comment out all ctyme.ixhash.net config
# body      CTYME_IXHASH eval:ixhashtest('ctyme.ixhash.net')
# describe  CTYME_IXHASH BiXhash found @ ctyme.ixhash.net
# tflags    CTYME_IXHASH net
# adjust as you seem fit
# score     CTYME_IXHASH 0.1
…
```

- Uncomment the following lines in the /etc/mail/spamassassin/v310.pre configuration file:

  Code:

```
loadplugin Mail::SpamAssassin::Plugin::DCC
loadplugin Mail::SpamAssassin::Plugin::Pyzor
loadplugin Mail::SpamAssassin::Plugin::Razor2
```

– Add these lines to the /etc/mail/spamassassin/local.cf configuration file:

Code:

```
use_ucc 1
ucc_timeout 10
use_pyzor 1
```

– Restart the SpamAssasson Dameon

Code:

```
/scripts/restartsrv_spamd
```

For what it's worth, I'm thinking about disabling Razor2. We get a non-trivial number of good email ending up in our junk folders due to this rule. I haven't been able to figure out why yet, but in the meantime we've been whitelisting (see no. 8 below) senders who get flagged by Razor2.

8. Update SpamAssassin Rule Scoring: Any of the hundreds of 'default' rule scores can be modified in a SpamAssassin preference file. I'm pretty sure that user preferences overrides server preferences.

Server preference file: /etc/mail/spamassassin/local.cf
User preference file: ~/.spamassassin/user_prefs

I do not use any scoring preferences in the local.cf file, but rather give each user a 'recommended' user_prefs file and encourage them to edit themselves if desired.

Code:

```
# Lower score stops more mail and potentially false positive
# We're doing ok flagging as spam with 3.0 or higher
required_score            3.0


# Whitelist
# This works by assigning a score of -100 or something like that
whitelist_from       *@myfavoritedomain.com
whitelist_from       *@iliketheseguystoo.org


# Bayes scores
score BAYES_40            1.0
score BAYES_50            2.0
score BAYES_60            3.0
score BAYES_80            4.0
score BAYES_95            5.0
score BAYES_99            6.0

# Give credit for being in whitelists
score RCVD_IN_DNSWL_NONE         -0.5
score RCVD_IN_DNSWL_LOW          -1.0
score RCVD_IN_DNSWL_MED          -1.5
score RCVD_IN_DNSWL_HI           -2.0
score RCVD_IN_MSPIKE_H2          -0.5
score RCVD_IN_MSPIKE_H3          -1.0
score RCVD_IN_MSPIKE_H4          -1.5
score RCVD_IN_MSPIKE_H5          -2.0
score RCVD_IN_IADB_LISTED         0.0
score RCVD_IN_IADB_VOUCHED        0.0
score RCVD_IN_IADB_DK             0.0
score RCVD_IN_IADB_RDNS           0.0

# Razor2 scores
# Penalty only once
score RAZOR2_CHECK                      1.0
score RAZOR2_CF_RANGE_51_100            0.0
score RAZOR2_CF_RANGE_E4_51_100         0.0
score RAZOR2_CF_RANGE_E8_51_100         0.0
```

```
# iXhash scores
# score GENERIC_IXHASH          0.5
# score NIXSPAM_IXHASH          0.5
# score HOSTEUROPE_IXHASH       0.5

# These are 'really' no longer needed because we now reject mail for
# SPF failures at SMTP time
score SPF_FAIL            1.0
score SPF_PASS            0.0
score SPF_NEUTRAL         0.0

# Some of these are 'really' no longer needed because we now reject mail
# for many of these RBL's at SMTP time
# Some are still useful though (e.g., URIBL_RED, URIBL_GREY, JP_SURBL)
score URIBL_BLACK        10.0
score URIBL_RED           1.5
score URIBL_GREY          1.5
score URIBL_SBL          10.0
score RCVD_IN_SBL        10.0
score RCVD_IN_XBL        10.0
score RCVD_IN_PBL        10.0
score URIBL_DBL_SPAM     10.0
score URIBL_JP_SURBL     10.0

# Spamcop
# Some false positives (?)
# Used to ccasionally block Hotmail
# Now rejecting at SMTP time
score RCVD_IN_BL_SPAMCOP_NET 0 1.246 0 1.347

# These (along with required_score = 3) helps to catch few that sneak thru
score HTML_MESSAGE        1.0
score UNPARSEABLE_RELAY   1.0
score T_REMOTE_IMAGE      1.0
```

9. Change Default Spam Folder Name (optional): It seems that Exim/SpamAssassin wants to put spam into a folder called “Spam”. While most IMAP servers, and IMAP mail clients for that matter, default to use a folder called “Junk”. Since all of my users use IMAP, I decided to change the Exim/SpamAssassin default folder to “Junk” to automatically be compatible with all of the user email clients’ defaults.

   – In WHM » Home » Service Configuration » Exim Configuration Manager, select the [Advanced Editor] tab, and then search for the “DIRECTORSTART” section and add the following filter rule:

   Code:

```
#
#  CUSTOM - Account level filtering for everything but the main account
#
custom_central_filter:
    driver = redirect
    allow_filter
    no_check_local_user
    file = /etc/cpanel_exim_system_filter_custom
    directory_transport = address_directory
    file_transport = address_file
    domains = +user_domains
    pipe_transport = virtual_address_pipe
    reply_transport = address_reply
    router_home_directory = ${extract{5}{:}{${lookup
passwd{${lookup{$domain}lsearch*{/etc/userdomains}{$value}}}{$value}}}}
    user = "${lookup{$domain}lsearch* {/etc/userdomains}{$value}}"
    allow_fail
    no_verify
```

   – Create the custom filter file: ~etc/cpanel_exim_system_filter_custom
   And add the following lines into that file. On line 8, the custom spam folder name is “Junk”, but can be changed to whatever you want:

Code:

```
#
# Exim filter
#
if not first_delivery and error_message then finish endif

# Custom spam box
if $h_X-Spam-Status: contains "Yes" then
    if "${if exists {$home/mail/$domain/$local_part/.Junk}{yes}{no}}" contains "yes"
        then
        save "$home/mail/$domain/$local_part/.Junk/" 660
        finish
    endif
endif
```

Scroll all the way down to the bottom of the Advanced Editor window/frame, and click on the "Save" button.

10. Implement Greylisting: If you've made it this far, then congratulations!! Hopefully your incoming spam is now dramatically reduced. At this point in the project, I was very pleased with the results. Most of our email accounts' spam dropped to zero. You may want to run your system for a while with these settings to see how it goes before moving on to greylisting.

    On our system, at this point in the project we had only one or two email accounts that were still getting some spam – approximately 5 - 10 per day per account yet with more than half of those ending up in the junk folder. We determined that for these few particularly stubborn spam, we needed a 0.5 to 2.0+ minutes delay in scanning the message … long enough for the DNS block lists (RBLs) to catch up to these senders. Greylisting does exactly that! We set ours to greylist 5 minutes and observed that the VAST majority of these senders never retry … and the few that do, end up getting caught in an RBL. ☺ Perfect‼

    – Add 'Time::Duration' module to the cPanel Perl installation:

      Code:

      ```
      /usr/local/cpanel/3rdparty/perl/514/bin/cpan –I Time::Duration
      ```

      If you're on a different version of WHM, then your cPanel Perl version may be different, in which case the "514" in the path may be a different number. Change the path as required …

– Create the MySQL database and tables. In WHM » Home » SQL Services » phpMyAdmin, select the [SQL] tab and enter the following SQL statement (or via MySQL command line interface if you prefer):

Code:

```
CREATE DATABASE eximgreylist;
CREATE USER 'eximgreylist'@'localhost' IDENTIFIED BY 'my_chosen_password';  ###  Create a unique password here ###
GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP ON eximgreylist.* TO 'eximgreylist'@'localhost';

USE eximgreylist;
CREATE TABLE `greylist` (
   `id`             varchar(20)                    NOT NULL,
   `relay_ip`       varchar(16)                                default NULL,
   `helo`           varchar(255)                               default NULL,
   `mail_from`      varchar(255)                               default NULL,
   `rcpt_to`        varchar(255)                               default NULL,
   `block_expires`  datetime                       NOT NULL    default '0000-00-00 00:00:00',
   `record_expires` datetime                       NOT NULL    default '0000-00-00 00:00:00',
   `blocked_count`  bigint(20)                     NOT NULL    default '0',
   `origin_type`    enum('MANUAL','AUTO')          NOT NULL    default 'MANUAL',
   `create_time`    datetime                       NOT NULL    default '0000-00-00 00:00:00',
   `last_update`    timestamp                      NOT NULL    default CURRENT_TIMESTAMP on update CURRENT_TIMESTAMP,
   PRIMARY KEY  (`id`),
   KEY `relay_ip_helo` (`relay_ip`, `helo`)
 );

USE eximgreylist;
CREATE TABLE `spamlist` (
   `relay_ip`       varchar(16)                    NOT NULL,
   `helo`           varchar(255)                   NOT NULL,
   `blocked_count`  bigint(20)                     NOT NULL    default '0',
   `origin_type`    enum('MANUAL','AUTO')          NOT NULL    default 'MANUAL',
   `create_time`    datetime                       NOT NULL    default '0000-00-00 00:00:00',
   `last_update`    timestamp                      NOT NULL    default CURRENT_TIMESTAMP on update CURRENT_TIMESTAMP,
   PRIMARY KEY  (`relay_ip`, `helo`)
 );

USE eximgreylist;
CREATE TABLE `whitelist` (
   `relay_ip`       varchar(16)                    NOT NULL,
   `helo`           varchar(255)                   NOT NULL,
   `record_expires` datetime                       NOT NULL    default '0000-00-00 00:00:00',
   `passed_count`   bigint(20)                     NOT NULL    default '0',
   `origin_type`    enum('MANUAL','AUTO','SMTP')   NOT NULL    default 'MANUAL',
   `create_time`    datetime                       NOT NULL    default '0000-00-00 00:00:00',
   `last_update`    timestamp                      NOT NULL    default CURRENT_TIMESTAMP on update CURRENT_TIMESTAMP,
   PRIMARY KEY  (`relay_ip`, `helo`)
 );
```

– Create Perl script with greylisting subroutines. As root user, create file /etc/exim.greylist.pl as follows:

Code:

```
use DBI;
use Time::Duration;
use Mail::Address;
use Net::DNS;

sub greylist() {

    my $timeout     = 5;  # minutes                  ###  Choose your greylisting duration              ###
    my $expiration  = 8;  # hours                    ###  Choose your greylisting exipiration           ###
    my $debug       = 1;  # 1=yes, 0=no              ###  Affects amount of data written to logfile     ###
    my $date        = localtime;
    my $myhost      = "localhost";
    my $mydb        = "eximgreylist";
    my $myuser      = "eximgreylist";
    my $mypasswd    = "my_chosen_password";          ###  Enter unique password created for MySQL DB user ###
    my $logfile     = "/var/log/exim_greylistlog";
    open(LOG, ">>$logfile");

    my $recipient_addr  = Exim::expand_string('$recipients');
    my $sender_addr     = Exim::expand_string('$sender_address');
    my $relay_host      = Exim::expand_string('$sender_host_address');
    my $relay_helo      = Exim::expand_string('$sender_helo_name');
    my $grey_id         = Exim::expand_string('$acl_m_grey_id');
    my $grey_status;

    $recipient_addr =~ s/\'/\\\'/g;
    $sender_addr    =~ s/\'/\\\'/g;
    $relay_host     =~ s/\'/\\\'/g;
    $relay_helo     =~ s/\'/\\\'/g;
    $grey_id        =~ s/\'/\\\'/g;

    if ($debug) {
        print LOG "$date  Exim variables:\n";
        print LOG "                              - receipent_addr = $recipient_addr\n";
        print LOG "                              - sender_addr    = $sender_addr\n";
        print LOG "                              - relay_host     = $relay_host\n";
        print LOG "                              - relay_helo     = $relay_helo\n";
        print LOG "                              - grey_id        = $grey_id\n";
    }

    # Open the greylist MySQL database
    my $isp = DBI->connect("DBI:mysql:$mydb:$myhost", "$myuser", "$mypasswd") || print LOG "$date  Can't connect to MySQL
database\n";
    my $query;
    my $sth;
    my @whitelisted;
    my @greylisted;

    # Search whitelist for host
    $query = "SELECT UNIX_TIMESTAMP()-UNIX_TIMESTAMP(record_expires), relay_ip, helo FROM whitelist WHERE
relay_ip='$relay_host' and helo='$relay_helo'";
    $sth = $isp->prepare($query) || print LOG "$date  Whitelist query prepare error: $query\n";
```

```
    $sth->execute || print LOG "$date  Whitelist query execute error: $query\n";
    @whitelisted = $sth->fetchrow_array;
    $sth->finish;

    # Search greylist for message
    $query = "SELECT UNIX_TIMESTAMP(block_expires)-UNIX_TIMESTAMP(), relay_ip, helo FROM greylist WHERE id='$grey_id'";
    $sth = $isp->prepare($query) || print LOG "$date  Greylist query prepare error: $query\n";
    $sth->execute || print LOG "$date  Greylist query execute error: $query\n";
    @greylisted = $sth->fetchrow_array;
    $sth->finish;

    if ($debug) {
        print LOG "$date  Database variables:\n";
        print LOG "                           - whitelisted    = @whitelisted\n";
        print LOG "                           - greylisted     = @greylisted\n";
    }

    unless (@whitelisted) {
        print LOG "$date  Host $relay_host ($relay_helo) not found in whitelist\n";

        unless(@greylisted) {
            print LOG "$date  Msg $grey_id not found in greylist\n";
            print LOG "                         Greylisting msg $grey_id\n";
            $query = "INSERT INTO greylist VALUES ('$grey_id', '$relay_host', '$relay_helo', '$sender_addr',
'$recipient_addr', DATE_ADD(NOW(), INTERVAL $timeout MINUTE), DATE_ADD(NOW(), INTERVAL $expiration HOUR), 1, 'AUTO', NOW(),
NULL)";
            $sth = $isp->prepare($query) || print LOG "$date  Greylist insert prepare error: $query\n";
            $sth->execute || print LOG "$date  Greylist insert execute error: $query\n";
            $sth->finish;
            $grey_status = $timeout*60;
        }
        else {
            print LOG "$date  Msg $grey_id found in greylist\n";

            if ($greylisted[0]<0) {
                print LOG "$date  Msg $grey_id greylist time has expired\n";
                print LOG "                         Whitelisting host $relay_host ($relay_helo)\n";
                $query = "INSERT INTO whitelist VALUES ('$relay_host', '$relay_helo', DATE_ADD(NOW(), INTERVAL $expiration
DAY), 1, 'AUTO', NOW(), NULL)";
                $sth = $isp->prepare($query) || print LOG "$date  Whitelist insert prepare error: $query\n";
                $sth->execute || print LOG "$date  Whitelist insert execute error: $query\n";
                $sth->finish;

                if ($greylisted[1] ne $relay_host or $greylisted[2] ne $relay_helo) {
                    print LOG "$date  Msg originally sent from different host $greylisted[1] ($greylisted[2])\n";
                    print LOG "                         Whitelisting host $greylisted[1] ($greylisted[2])\n";
                    $query = "INSERT INTO whitelist VALUES ('$greylisted[1]', '$greylisted[2]', DATE_ADD(NOW(), INTERVAL
$expiration DAY), 1, 'AUTO', NOW(), NULL)";
                    $sth = $isp->prepare($query) || print LOG "$date  Whitelist insert prepare error: $query\n";
                    $sth->execute || print LOG "$date  Whitelist insert execute error: $query\n";
                    $sth->finish;
                }

                print LOG "$date  Removing msg $grey_id from greylist\n";
                $query = "DELETE FROM greylist WHERE id='$grey_id'";
```

```
                $sth = $isp->prepare($query) || print LOG "$date  Greylist delete prepare error: $query\n";
                $sth->execute || print LOG "$date  Greylist delete execute error: $query\n";
                $sth->finish;
            }
            else {
                print LOG "$date  Msg $grey_id greylist time has not expired\n";
                $query = "UPDATE greylist SET blocked_count=blocked_count+1 WHERE id='$grey_id'";
                $sth = $isp->prepare($query) || print LOG "$date  Greylist update prepare error: $query\n";
                $sth->execute || print LOG "$date  Greylist update execute error: $query\n";
                $sth->finish;
                $grey_status = $greylisted[0];
            }
        }
    }
    else {
        print LOG "$date  Host $relay_host ($relay_helo) found in whitelist\n";

        if (@greylisted) {
            print LOG "$date  Msg $grey_id previously greylisted\n";

            # Can have host in both greylist and whitelist when 2 or more messages are greylisted and then expire
            if ($whitelisted[1] ne $relay_host or $whitelisted[2] ne $relay_helo) {
                print LOG "                          Whitelisting previously greylisted host $greylisted[1]
($greylisted[2])\n";
                $query = "INSERT INTO whitelist VALUES ('$greylisted[1]', '$greylisted[2]', DATE_ADD(NOW(), INTERVAL
$expiration DAY), 1, 'AUTO', NOW(), NULL)";
                $sth = $isp->prepare($query) || print LOG "$date  Whitelist insert prepare error: $query\n";
                $sth->execute || print LOG "$date  Whitelist insert execute error: $query\n";
                $sth->finish;
            }
            print LOG "$date  Removing msg $grey_id from greylist\n";
            $query = "DELETE FROM greylist WHERE id='$grey_id'";
            $sth = $isp->prepare($query) || print LOG "$date  Greylist delete prepare error: $query\n";
            $sth->execute || print LOG "$date  Greylist delete execute error: $query\n";
            $sth->finish;
        }
        print LOG "$date  Updating whitelist expiration for host $relay_host ($relay_helo)\n";
        $query = "UPDATE whitelist SET record_expires=DATE_ADD(NOW(), INTERVAL $expiration DAY),
passed_count=passed_count+1 WHERE relay_ip='$relay_host' and helo='$relay_helo'";
        $sth = $isp->prepare($query) || print LOG "$date  Whitelist update prepare error: $query\n";
        $sth->execute || print LOG "$date  Whitelist update execute error: $query\n";
        $sth->finish;
    }
    if ($grey_status) {
        $grey_status = duration($grey_status);
        print LOG "$date  Return grey_status = $grey_status\n";
    }
    else {
        print LOG "$date  Return grey_status = NOT_GREY\n";
    }
    close(LOG);
    $isp->disconnect;
    return ($grey_status);
}
```

```
sub smtp_whitelist() {

    my $expiration  = 60; # days
    my $debug       = 1;  # 1=yes, 0=no
    my $date        = localtime;
    my $myhost      = "localhost";
    my $mydb        = "eximgreylist";
    my $myuser      = "eximgreylist";
    my $mypasswd    = "my_chosen_password";          ###  Enter unique password created for MySQL DB user ###
    my $logfile     = "/var/log/exim_greylistlog";
    open(LOG, ">>$logfile");

    my @recipients = split(/,/, Exim::expand_string('$recipients'));
    my $recipient;
    my $rcpt_domain;
    my $rcpt_domain_mx1;
    my $rcpt_domain_mx2;
    my $rcpt_ip_mx;
    my @mx;
    my @mx_host;

    # Open the greylist MySQL database
    my $isp = DBI->connect("DBI:mysql:$mydb:$myhost", "$myuser", "$mypasswd") || print LOG "$date  Can't connect to MySQL
database\n";
    my $query;
    my $sth;
    my @whitelisted;

    # Setup DNS resolver
    my $res = new Net::DNS::Resolver;
    $res->nameservers("127.0.0.1");

    foreach $recipient (@recipients) {

        # Escape single quotes if present
        $recipient =~ s/\'/\\\'/g;

        # Get domain name from email address
        my ($addr) = Mail::Address->parse($recipient);
        $rcpt_domain = $addr->host;

        # Lookup MX record for domain name
        print LOG "$date  Checking recipient $recipient\n";
        @mx = mx($res, $rcpt_domain);
        if (@mx) {
            foreach my $rr (@mx) {
                $rcpt_domain_mx1 = $rr->exchange;
                @mx_host = &mx_fwd_lookup($rcpt_domain_mx1,);
                $rcpt_ip_mx = @mx_host[0];
                $rcpt_domain_mx2 = @mx_host[1];
                chop($rcpt_domain_mx2);
                print LOG "                          -> $rcpt_domain_mx1 -> $rcpt_ip_mx -> $rcpt_domain_mx2\n";

                # Do not white list MX ip/domain if DNS lookup returns "NXDOMAIN"
                unless ($rcpt_domain_mx2 =~ /^NXDOM/) {
```

```
                    # Check for whitelisting at dnswl.org
                    if (mx_dnswl_lookup($rcpt_ip_mx)==1) {
                        print LOG ", whitelisted at dnswl.org\n";
                    }
                    else {

                        # Check for duplicates in whitelist
                        $query = "SELECT COUNT(*) FROM whitelist WHERE relay_ip='$rcpt_ip_mx' and helo='$rcpt_domain_mx2'";
                        $sth = $isp->prepare($query) || print LOG "$date  Whitelist count prepare error: $query\n";
                        $sth->execute || print LOG "$date  Whitelist count execute error: $query\n";
                        @whitelisted = $sth->fetchrow_array;
                        $sth->finish;

                        if (@whitelisted[0] > 0) {
                            print LOG ", previously whitelisted\n";
                        }
                        else {
                            print LOG ", adding to whitelist\n";
                            $query = "INSERT INTO whitelist VALUES ('$rcpt_ip_mx', '$rcpt_domain_mx2', DATE_ADD(NOW(),
INTERVAL $expiration DAY), 1, 'SMTP', NOW(), NULL)";
                            $sth = $isp->prepare($query) || print LOG "$date  Whitelist insert prepare error: $query\n";
                            $sth->execute || print LOG "$date  Whitelist insert execute error: $query\n";
                            $sth->finish;
                        }
                    }
                }
            }
        }
        else {
            print LOG "                          -> No MX records found for $rcpt_domain: ", $res->errorstring, "\n";
        }
    }
    close(LOG);
    $isp->disconnect;
}

sub mx_fwd_lookup {
    my $rcpt_domain_mx1 = $_[0];
    my $res = new Net::DNS::Resolver;
    my $query = $res->search($rcpt_domain_mx1);
    if ($query) {
        foreach my $rr ($query->answer) {
            next unless $rr->type eq "A";
            my $rcpt_rev = &mx_rev_lookup($rr->address,);
            my $rcpt_ip_mx = $rr->address,;
            my @lookedup = ($rcpt_ip_mx,$rcpt_rev);
            return @lookedup;
        }
    }
    else {
        return $res->errorstring,;
    }
}
```

```
sub mx_rev_lookup {
    my $rcpt_ip_mx = $_[0];
    my $res = new Net::DNS::Resolver;
    my $query = $res->search($rcpt_ip_mx);
    if ($query) {
        foreach my $rr ($query->answer) {
            next unless $rr->type eq "PTR";
            return $rr->rdatastr,;
        }
    }
    else {
        return $res->errorstring,;
    }
}

sub mx_dnswl_lookup {
    my $rcpt_ip_mx = $_[0];
    my @numbers = split (/\./, $rcpt_ip_mx);
    my $res = new Net::DNS::Resolver;
    my $query = $res->search(join(".", reverse 'org', 'dnswl', 'list', @numbers));
    if ($query) {
        foreach my $rr ($query->answer) {
            next unless $rr->type eq "A";
            my $dnswl_ip = $rr->address,;
            print LOG "                              dnswl.org lookup = $dnswl_ip";
            return 1;
        }
    }
    else {
        print LOG "                             dnswl.org query empty";
        return 0;
    }
}

sub rcpt_spamlist() {

    my $debug       = 1;
    my $date        = localtime;
    my $myhost      = "localhost";
    my $mydb        = "eximgreylist";
    my $myuser      = "eximgreylist";
    my $mypasswd    = "my_chosen_password";           ###  Enter unique password created for MySQL DB user ###
    my $logfile     = "/var/log/exim_greylistlog";
    open(LOG, ">>$logfile");

    my $relay_host  = Exim::expand_string('$sender_host_address');
    my $relay_helo  = Exim::expand_string('$sender_helo_name');
    my $spam_status = 0;

    $relay_host =~ s/\'/\\\'/g;
    $relay_helo =~ s/\'/\\\'/g;

    # Open the greylist MySQL database
    my $isp = DBI->connect("DBI:mysql:$mydb:$myhost", "$myuser", "$mypasswd") || print LOG "$date  Can't connect to MySQL
database\n";
```

```
    my $query;
    my $sth;
    my @spamlisted;

    # Check spamlist
    $query = "SELECT COUNT(*) FROM spamlist WHERE relay_ip='$relay_host' and helo='$relay_helo'";
    $sth = $isp->prepare($query) || print LOG "$date  Spamlist count prepare error: $query\n";
    $sth->execute || print LOG "$date  Spamlist count execute error: $query\n";
    @spamlisted = $sth->fetchrow_array;
    $sth->finish;


    if (@spamlisted[0] > 0) {
        print LOG "$date  Host $relay_host ($relay_helo) found in grey spamlist\n";
        $query = "UPDATE spamlist SET blocked_count=blocked_count+1 WHERE relay_ip='$relay_host' and helo='$relay_helo'";
        $sth = $isp->prepare($query) || print LOG "$date  Spamlist update prepare error: $query\n";
        $sth->execute || print LOG "$date  Spamlist update execute error: $query\n";
        $sth->finish;
        $spam_status = 1;
    }
    close(LOG);
    $isp->disconnect;
    return ($spam_status);
}

sub rcpt_ptr() {

    my $debug = 1;
    my $date = localtime;
    my $myhost = "localhost";
    my $logfile = "/var/log/exim_greylistlog";
    open(LOG, ">>$logfile");

    my $relay_host  = Exim::expand_string('$sender_host_address');
    my $relay_helo  = Exim::expand_string('$sender_helo_name');
    my $relay_rev;
    $relay_host =~ s/\'/\\\'/g;
    $relay_helo =~ s/\'/\\\'/g;

    # Setup DNS resolver
    my $res = new Net::DNS::Resolver;
    $res->nameservers("127.0.0.1");

    print LOG "$date  Checking reverse PTR for host $relay_host\n";
    $relay_rev = &ptr_rev_lookup($relay_host);
    chop($relay_rev);
    print LOG "$date  -> $relay_helo -> $relay_host -> $relay_rev\n";

    unless ($relay_rev =~ /^NX/) {
        print LOG "$date  Yes rDNS, returning (1)\n";
        close(LOG);
        return (1);
    }
    close(LOG);
}
```

```
sub ptr_rev_lookup {
    my $relay_host = $_[0];
    my $res = new Net::DNS::Resolver;
    my $query = $res->search($relay_host);
    if ($query) {
        foreach my $rr ($query->answer) {
            next unless $rr->type eq "PTR";
            return $rr->rdatastr;
        }
    }
    else {
        return $res->errorstring,;
    }
}

1;
```

– Create Perl script for greylist database cleanup. As root user, create file /etc/exim.greylist.clean.pl as follows:

Code:

```
#!/usr/bin/perl/ -w

use strict;
use DBI;

my $date = localtime;
my $mydb = "eximgreylist";
my $myhost = "localhost";
my $myuser = "eximgreylist";
my $mypasswd    = "my_chosen_password";          ###  Enter unique password created for MySQL DB user ###
my $logfile = "/var/log/exim_greylistlog";
open(LOG, ">>$logfile");

my $i;
my $grey_id;
my $relay_host;
my $relay_helo;
my $blocked;
my $created;

# Open the greylist MySQL database
my $isp = DBI->connect("DBI:mysql:$mydb:$myhost", "$myuser", "$mypasswd") || print LOG "$date  Can't connect to MySQL
database\n";
my $query;
my $sth;
my $expired;

$query = "SELECT id, relay_ip, helo, blocked_count, create_time FROM greylist WHERE ((UNIX_TIMESTAMP() -
UNIX_TIMESTAMP(record_expires) > 0))";
$sth = $isp->prepare($query) || print LOG "$date  DB clean query prepare error: $query\n";
$sth->execute || print LOG "$date  DB clean query execute error: $query\n";
$expired = $sth->fetchall_arrayref || print LOG "$date  DB clean query fetch error: $query\n";
$sth->finish;
```

```
foreach $i (0 .. $#{$expired}) {

    $grey_id    = $expired->[$i][0];
    $relay_host = $expired->[$i][1];
    $relay_helo = $expired->[$i][2];
    $blocked    = $expired->[$i][3];
    $created    = $expired->[$i][4];

    print LOG "$date  Processing expired grey record no. $i, $relay_host ($relay_helo)\n";

    $relay_host =~ s/\'/\\\'/g;
    $relay_helo =~ s/\'/\\\'/g;
    $blocked    =~ s/\'/\\\'/g;
    $created    =~ s/\'/\\\'/g;

    $query = "INSERT INTO spamlist VALUES ('$relay_host', '$relay_helo', $blocked, 'AUTO', '$created', NULL)";
    $sth = $isp->prepare($query) || print LOG "$date  DB clean insert prepare error: $query\n";
    $sth->execute || print LOG "$date  DB clean insert execute error: $query\n";
    $sth->finish;

    $query = "DELETE FROM greylist WHERE id='$grey_id'";
    $sth = $isp->prepare($query) || print LOG "$date  DB clean delete prepare error: $query\n";
    $sth->execute || print LOG "$date  DB clean delete execute error: $query\n";
    $sth->finish;
}

close(LOG);
$isp->disconnect;
```

- Set (or confirm) Perl script file permissions. In an SSH terminal as root user, enter the following:

  Code:

```
chmod 644 /etc/exim.greylist.pl
chmod 644 /etc/exim.greylist.clean.pl
```

- Setup greylist database cleanup cron. In an SSH terminal as root user, enter the following (use whatever timing you prefer, mine is set for daily at 1:30 AM):

  Code:

```
Crontab -e
30 1 * * * /usr/bin/perl  /etc/exim.greylist.clean.pl
```

- Add greylist perl script to Exim perl startup parameter. In WHM » Home » Service Configuration » Exim Configuration Manager, select the [Advanced Editor] tab, and then search for the “perl_startup” variable definition. Change the value as follows:

  Code:

```
{do '/etc/exim.pl'; do '/etc/exim.greylist.pl';}
```

  Scroll all the way down to the bottom of the Advanced Editor window/frame, and click on the “Save” button.

- Add exim_greylistlog to the log rotation schedule . As root user, create file /etc/logrotate.d/exim_greylist as follows (I configured mine the same as the other exim logs)

  Code:

```
/var/log/exim_greylistlog {
    create 0640 mailnull mail
    compress
    postrotate
    /usr/bin/killall -HUP exim
    endscript
}
```

- Add various ACL's to trigger greylist subroutine(s). In WHM » Home » Service Configuration » Exim Configuration Manager, select the [Advanced Editor] tab, and then add ACL rules as desired:

  Automatically exempt from greylisting, mailservers that any of our users send email to. Do not use this rule unless you trust all of your users to send email to only trustworthy recipients. You may not want to use this rule if you are a reseller; or you could add a condition that includes a list of 'trusted' users. Enter this rule in the "custom_begin_smtp_predata" block in acl_smtp_predata.

  Code:

```
# Whitelisting - sendmail recipients (acl_smtp_predata)
  warn authenticated = *
       set acl_m_white = ${perl{smtp_whitelist}}
  accept
```

  Flag this message greylisting if it contains any HTML parts. Enter this rule in the "custom_begin_smtp_mime" block in acl_smtp_mime.

  Code:

```
# HTML mail (in acl_smtp_mime)
  warn !condition = $mime_is_rfc822
       condition = $mime_is_coverletter
       condition = ${if eq {$mime_content_type}{text/html}{1}}
       set acl_m_greylistreasons_short = \040-HTML mail$acl_m_greylistreasons_short
       set acl_m_greylistreasons = \040\040- Message appears to have HTML content, not just
plain text\n$acl_m_greylistreasons
  accept
```

  Flag this message for greylisting if it has no Message-ID. Enter this rule in the "custom_end_check_message_pre" block in acl_smtp_data.

  Code:

```
# No Message-ID (beginning of acl_smtp_data)
  warn condition = ${if !def:h_Message-ID: {1}}
       set acl_m_greylistreasons_short = \040-no Message-ID$acl_m_greylistreasons_short
       set acl_m_greylistreasons = \040\040- Message has no Message-
ID\n$acl_m_greylistreasons
```

  Flag this message for greylisting if it has any MIME errors. Enter this rule in the "custom_end_check_message_pre" block in acl_smtp_data.

  Code:

```
# MIME errors (beginning of acl_smtp_data)
  warn demime = *
       condition = ${if >{$demime_errorlevel}{0}{1}{0}}
       set acl_m_greylistreasons_short = \040-MIME error(s)$acl_m_greylistreasons_short
       set acl_m_greylistreasons = \040\040- Message contains one or more MIME errors
($demime_reason)\n$acl_m_greylistreasons
```

Flag this message for greylisting if it is a fake “In-Reply-To” message. Enter this rule in the “custom_end_check_message_pre” block in acl_smtp_data.

Code:

```
# Fake replies (beginning of acl_smtp_data)
  warn  condition = ${if and {                                  \
                        {match {${lc:$h_subject:}}{^re:}}   \
                        {!def:h_References:}                  \
                        {!def:h_In-Reply-To:}                 \
                        } {1}{0}}
        set acl_m_greylistreasons_short = \040-fake reply$acl_m_greylistreasons_short
        set acl_m_greylistreasons = \040\040- Message has 'Re:' in Subject: but there is
no References: or In-Reply-To:\n$acl_m_greylistreasons
```

Flag this message for greylisting if SpamAssassin detects any ‘unparseable relay’ lines. Enter this rule in the “custom_end_spam” block in acl_smtp_data.

Code:

```
# Unparseable relay lines (in acl_smtp_data after spam scan)
  warn condition = ${if match {$spam_report}{UNPARSEABLE_RELAY}{1}}
# warn condition =  ${if match {$spam_report}{T_TVD_FW_GRAPHIC_ID2}{1}}
      set acl_m_greylistreasons_short = \040-unparseable relay$acl_m_greylistreasons_short
      set acl_m_greylistreasons = \040\040- Message has unparseable relay
lines\n$acl_m_greylistreasons
```

Flag this message for greylisting if SpamAssassin score is greater than 3.0 (or change “3” in line 2 below to some other spam score value). THIS RULE HAS NOT BEEN TESTED. Enter this rule in the “custom_end_spam” block in acl_smtp_data.

Code:

```
# Spam score >= 3.0 (in acl_smtp_data after spam scan)
  warn condition = ${if >={$spam_score_int}{3} {1}}
       set acl_m_greylistreasons_short = \040-spam score$acl_m_greylistreasons_short
       set acl_m_greylistreasons = \040\040- Message flagged as possible
spam\n$acl_m_greylistreasons
```

Flag this message for greylisting if the host is found in a DNS block list (RBL) … one that we don’t already reject on. THIS RULE HAS NOT BEEN TESTED. Enter this rule in the “custom_begin_rbl” block in acl_smtp_rcpt.

Code:

```
# Sending host in RBL (after rbl checks in acl_smtp_rcpt)
  warn dnslists = grey.uribl.com
       set acl_m_greylistreasons_short = \040-grey.uribl.com$acl_m_greylistreasons_short
       set acl_m_greylistreasons = \040\040- Host listed in $dnslist_domain dnsbl:
$dnslist_text\n$acl_m_greylistreasons
  warn dnslists = SOME.OTHER.RBL.COM
       set acl_m_greylistreasons_short = \040-
SOME.OTHER.RBL.COM$acl_m_greylistreasons_short
       set acl_m_greylistreasons = \040\040- Host listed in $dnslist_domain dnsbl:
$dnslist_text\n$acl_m_greylistreasons
```

Invoke the greylisting subroutine if any of the above greylisting conditions are met and if the host is not whitelisted at www.dnswl.org. Enter this rule in the “custom_begin_check_message_post” block in acl_smtp_data.

Code:

```
# Whitelisting (end of acl_smtp_data)
  warn dnslists = list.dnswl.org
       set acl_m_greylistreasons =

# Greylisting (end of acl_smtp_data)
  warn !condition = ${if eq {$acl_m_greylistreasons}{}{1}}
       set acl_m_grey_id = ${hash{20}{62}{$sender_address$recipients$h_message-id:}}
       set acl_m_grey = ${perl{greylist}}
  defer !condition = ${if eq {$acl_m_grey}{0}{1}}
```

```
        log_message = "Greylisted for $acl_m_grey:$acl_m_greylistreasons_short"
        message = Your mail was considered suspicious for the following
reason(s):\n$acl_m_greylistreasons\n\n \
         As a precaution, your mailserver has been "greylisted" for a short period     \
         of time ($acl_m_grey remaining). Your mailserver should keep the              \
         mail in its queue and automatically attempt to resend the message after the   \
         greylisting period has expired. After your message has been successfully      \
         received, then your mailserver will be permanently "whitelisted".  We        \
         apologize for any inconvenience.
```

Defer (or reject if desired) this message if the sending host has been previously greylisted and never attempted to resend (spamlist). Enter this rule in the "custom_end_rbl" block in acl_smtp_rcpt.

Code:

```
# Grey spamlist (after rbl checks in acl_smtp_rcpt)
  defer !condition = ${if eq {${perl{rcpt_spamlist}}}{0}{1}}
        !local_parts = abuse : postmaster
        message = Message deferred - $sender_fullhost - $sender_address_domain - is in our
grey spamlist. For questions, email abuse@<mydomain>.com
        log_message = "Deferred: $sender_fullhost - $sender_address_domain in grey
spamlist"
```

Change the email address in the ACL above to your own preferred 'abuse' or 'webmaster' email address.

- Check for reverse DNS (optional): We are already rejecting messages from hosts that are in the noptr.spamrats.com DNS block list. That block list is a collection of IP addresses that don't have an rDNS and are sending out a flood of connections. But just to see if it misses any hosts with no rDNS, I included a subroutine in exim.greylist.pl that performs an rDNS lookup. Enter this rule in the "custom_begin_rbl" block in acl_smtp_rcpt.

  Code:

```
# Verify rDNS (after rbl checks in acl_smtp_rcpt)
  warn !local_parts = abuse : postmaster
       set acl_m_ptr = ${perl{rcpt_ptr}}
  drop !condition =  ${if eq {${acl_m_ptr}}{1}{1}}
       message = Message rejected - no reverse DNS for $sender_host_address, please fix
your reverse PTR. For questions, email abuse@powerproductsandservices.com
       log_message = "Rejected: $sender_host_address has no rDNS"
```

11. Implement Nolisting:

WARNING!!

Do not edit your DNS zones unless you really know what you're doing.

On our server, the anti-spam settings and modifications listed above have been 100% successful in stopping spam. So for us, implementation of nolisting is strictly aimed at reducing traffic and server load.

- Create two (2) mailserver subdomains.

  For each domain that receives mail on the server, edit its DNS zone, creating two (2) mailserver subdomains. In WHM » Home » DNS Functions » Edit DNS Zone, select a domain and click the "Edit" button (be sure to delete or overwrite any old mailserver records).

  In the sample DNS zone editing page shown below, the DNS zone domain being edited is "yourdomain.com" (note: this may be a sub-account domain).

And the two IP addresses assigned to the server are:

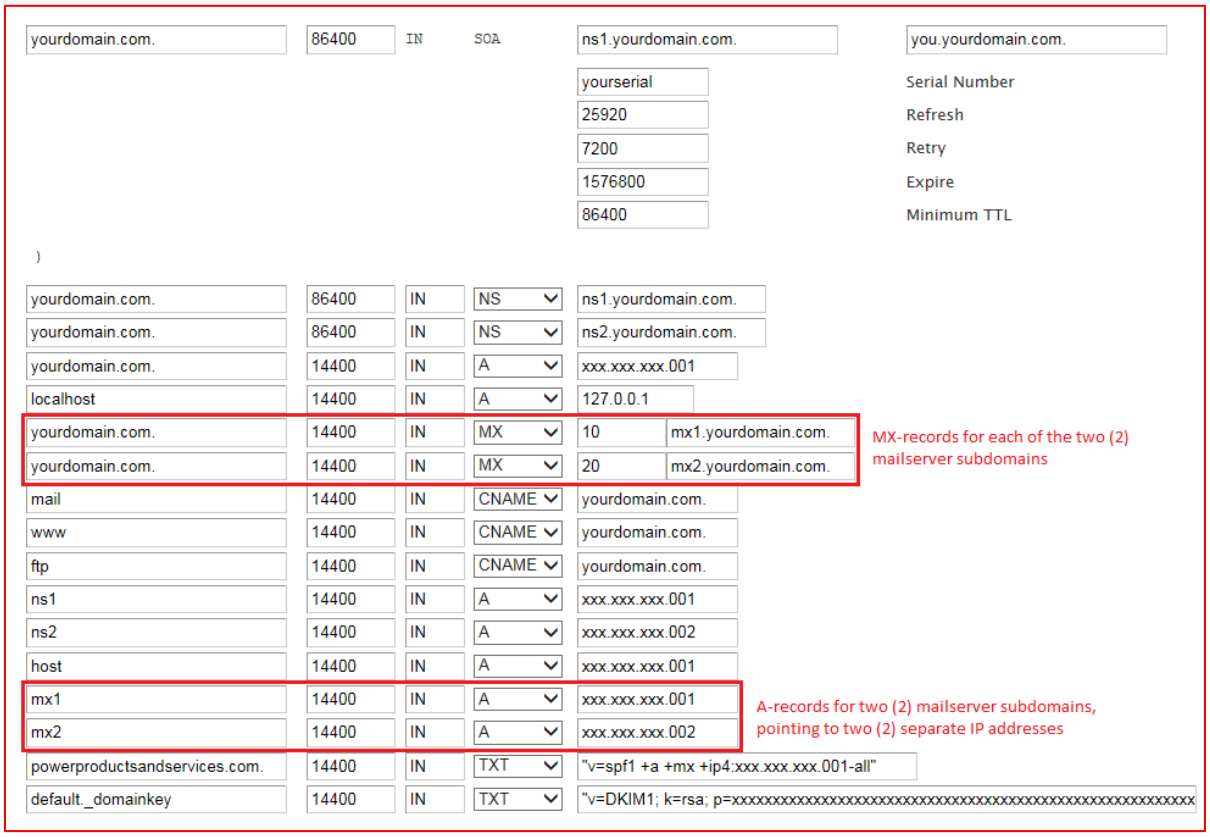
xxx.xxx.xxx.001 » change to your server primary IP address

xxx.xxx.xxx.002 » change to your server secondary IP address

And the two mailserver subdomains are:

10 mx1.yourdomain.com » choose any primary MX subdomain name that you like

20 mx2.yourdomain.com » choose any secondary MX subdomain name that you like



– Block port 25 of the primary MX IP address. Create file /etc/csf /csfpre.sh and enter this iptable command in that file, substituting your primary MX IP address for 'xxx.xxx.xxx.001':

Code:

```
# Run iptable commands before csf configures iptables
iptables -I INPUT -p tcp -d xxx.xxx.xxx.001 --dport 25 -j REJECT --reject-with tcp-reset
iptables -I INPUT -p tcp -d xxx.xxx.xxx.001 --dport 26 -j REJECT --reject-with tcp-reset
```

– Confirm that it's working:

From any computer outside your server's firewall:

- Open a telnet session to "mx1.yourdomain.com 25" and observe failed connection.
- Open a telnet session to "mx2.yourdomain.com 25" and observe successful connection.
- Send an email to yourself (still from outside your server's firewall), and observe in the /var/log/exim_mainlog that the email message is received on your secondary MX IP address instead of on your primary MX IP address.

12. NOW!! – Enjoy your nice, shiny, spam-free mailserver:

    Open an SSH terminal session (preferably in a large window) and run “tail -f /var/log/exim_mainlog /var/log/exim_greylistlog”, grab a beer, sit back, and watch all of that spam get shot down!! ☺

Rather than copy/paste all of the code in this document, feel free to pick it up here: Mailserver Setup Files

Please feel free to let me know if you think any corrections should be made to this document or if you have any additional information or insight to contribute. You can PM me over at the knownhost.com forums here, or send me an email me at contact form.

Cheers!
Robert Medure
https://powerproductsandservices.com

P.S. APPENDIX WITH MORE DETAILED EXPLANATION/REASONING COMING SOON

Need to add code to clean-up the log file (similar to other exim logs).
I was thinking about adding some code to this database clean-up script to also "blacklist" (in Exim) hosts that never resend after greylisting. This could be done by writing their IP's to the /etc/spammeripblocks file (this list is normally managed in the WHM » Home » Service Configuration » Exim Configuration Manager, [Basic Editor] tab, [Access Lists] sub-tab, "Blacklisted SMTP IP addresses"). But to date, not a single 'greylist spammer' has attempted send spam again from the same ip/hostname.

(they get blocked anywayIP numbers of type "SMTP" (obtained from outgoing mail – recipient mailserver hosts). This could be accomplished by writing those ip numbers to the /etc/spammera section to this script to write these ip numbers into the /etc/spetup greylist database cleanup cron. Add this line to the /var/spool/cron/root cron file (use whatever timing you prefer):