Using BGP for realtime import and export of OpenBSD spamd entries

Peter Hessler
phessler@openbsd.org
OpenBSD

Bob Beck
beck@openbsd.org
OpenBSD

1 Introduction

In the battle against spam, many mail server admins collect and distribute IP addresses of systems that have sent them spam. However, distribution of these lists are traditionally limited to 2 methods. Method #1 is periodically downloading this list from a source, usually a web server - which is subject to load issues on the target web server. #2 is a real-time lookup against an external provider (such as dns-rbls) so your response time is dependent upon how fast they respond or timeout.

OpenBSD spamd[1] is typically used as a daemon to stop the "low hanging fruit" of spam, by subjecting previously unseen hosts to `greylisting` to allow time to identify it they are a real mailserver. Real mailservers are whitelisted locally after having passed greylisting, with their connections to the real mailservers monitored via `spamlogd`[2] . As such spamd keeps a list in a database of what it believes to be currently active "real" mailservers.

This paper suggests and discusses a 3rd solution: using BGP[3] to distribute the IP addresses in a real-time manner. By doing so we can take advantage of OpenBSD `spamd's` information to distribute two useful lists via BGP:

1. Each participant can share their TRAPPED entries from spamd(8) - hosts which the local spamd has determined should not be allowed to pass greylisting. Other hosts can use these lists to also prevent such hosts from passing greylisting in the short term.

2. By taking advantage of the information kept in `spamdb` - each participant can share a subset of their WHITE entries from `spamdb`, chosen based on criteria that makes them very likely to be real mail servers that are continuing to exchange mail with the participating domain on a regular basis. By doing this all participants can use this information to build a `bypass` table in `pf`[4] so that all such "real mailservers" talking to any participant in the network are not subjected to greylisting.

Having a greater amount of information is, of course, a great boon to a mail server administrator. This paper will show how an admin can use blacklist entries to not only block access from badly behaving mail servers, but, more importantly, allow access from so-called "known good" mail servers.

1.1 Traditional use of spamd(8)

Traditionally, OpenBSD users will use the spamd(8) daemon included with OpenBSD. This daemon will keep track of hosts it has communicated with, and put them in one of 3 lists. GREY, WHITE, and TRAPPED which are tracked in spamd's `spamdb`[5] database.

Whitelisted (WHITE) hosts do not talk to spamd(8), and are instead sent to the real mailserver.

Greylisted (GREY) hosts are not on the WHITE or TRAPPED lists. Normally these are hosts for which no SMTP[6] activity has been seen previously. These connections are redirected to spamd(8), and are given a temporary failure message when they try to deliver mail. Greylisted become Whitelisted after they retry delivery of the same message more than 3 times and are still retrying after 30 minutes of delay.

Trapped (TRAPPED) hosts are hosts which have been identified as doing bad things during the time they are Greylisted. Typically this is mailing to an unknown or spamtrap user, mailing with invalid smtp usage, or other spam signature activity. These hosts can be TRAPPED, typically for 24 hours, and will not be allowed to pass Greylisting during that time. This can be quite powerful because the trapping is only applied to hosts for which mail has never been exchanged before. As an example, it is not unusual for a legitimate mail server to mail to a large number of unknown users. However, it *is* unusual for a real mail server for which we have never exchanged mail with before to suddenly start mailing unknown users on its first communication with us.

Spamd also has the ability to use external blacklists, where clients on such a list will be given a rejection message specific to that list. This allows the default Greylisting behaviour to be combined with external lists of spam sources. The `spamd-setup(8)`[7] utility sends external blacklist data to `spamd`, as well as configuring mail rejection messages for blacklist entries. This utility uses method #1 to retrieve the list of blacklist entries.

In our case, we use BGP to distribute TRAPPED lists so that they may be used as external BLACK lists - as well as distributing selected WHITE entries so they can be shared among sites.

2 Definitions: Client vs Route Server vs Spamd Source

In this paper, we will discuss a reference implementation network. The authors will implement this network and it will be available for public use at "`rs.bgp-spamd.net`".

There are three important parts of the reference implementation network.

`Spamd Source`: These systems are feeding the Route Server with IP addresses fed from GREYTRAP and WHITE lists. They are the source of our `spamd` list information. Client systems are not able connect directly to the Spamd Source systems, their information will be sent via the Route Server.

Our reference implementation uses: University of Alberta (aka uatraps), Bob Beck (aka obtuse.com), and Peter Hansteen (aka bsdly.net) as our Spamd Sources.

`Route Server`: This system is the center hub of the network. Both the Spamd Sources and the Client systems connect. This system sorts and redistributes the BGP feeds from the Spamd Sources and distributes them to the Clients.

Our implementation uses the reference server, "rs.bgp-spamd.net"

`Client`: Any end-user.

Originally using sources from OpenBSD's `/etc/mail/spamd.conf`

3 Using BGP to distribute spamd lists

3.1 Basic explanation of BGP

In a traditional BGP network, Router A will send a list of IP networks assigned to itself, to all of its peers. Router B will also distribute its IP networks, as well as the IP networks assigned to Router A. Router A can mark specific IP networks with attributes, known as Communities, as a way to communicate with Router B some intentions for

these routes.

Some common examples of community meanings include "do not redistribute to a specific geographical area", "Prefix this route 3 times (make it less desirable to the peers)", and "Blackhole this network".

3.2 Our use of BGP

In this paper, we will use the fact that BGP is designed to distribute IP networks with community attributes to distribute TRAPPED and certain WHITE entries from `spamd`. We want to do this for two reasons:

1. We distribute TRAPPED entries from trusted sources to use as BLACK lists - we are assuming that our trusted sources have a reasonable criteria for trapping hosts, and that TRAPPED hosts are short lived.
2. We distribute a subset of the WHITE entries from `spamd` - Our goal is to distribute entries that we are confident are "real" mailservers based on the information in the `spamdb` database. This list of "likely good mailservers" can be used by participants to establish a `bgp-spamd-bypass` table managed by `bgpd`[8] in `pf`. The advantage of this is that "real" mailservers communicating regularly with *any* participant will not be subjected to greylisting delays at *all* participants.

We have chosen some arbitrary BGP Communities for use in marking BLACK list and WHITE entries. The authors have chosen `$AS:666` for BLACK list entries, and `$AS:42` for WHITE entries. In this case, `$AS` will be expanded to the AS of the originating system. For additional filtering capabilities we will also copy this Community and mark it with the AS on the Route Server.

While it is possible for a regular BGP router to be a member of this experiment, the authors recommend against it. Clients MUST NOT make routing decisions based on the routes that we send, a large amount of host-specific routes will be injected into the our BGP network, and care needs to be taken to ensure that the behaviour of one BGP network does not affect the behaviour of the other. With our sample configuration files, client and server entities do not need to be directly connected, and can easily be routed over the regular internet, even through NAT.

3.3 Distributing spamd bypass lists

When a previously unseen host hits OpenBSD spamd, it is normally subject to greylisting - meaning initial connections will be delayed and forced to retry for 30 minutes. After 30 minutes, if the same tuple[9] of mail is still being retried the connecting host passes greylisting, and is whitelisted. Whitelisted hosts are kept for 30 days from the time traffic from them is last seen, and spamd counts number of connections passed, keeping whitelisted hosts from being subject to greylisting again as long as they continue to periodically send mail.

Some sites already use the recommendation of a `<nospamd>` table to bypass spamd for trusted networks - often those may include things like Google's mailserver networks, etc.

What we want for a spamd bypass list from other sources is not to know "this host passed greylisting" - but rather "I am pretty sure this is a real mail server". Using BGP puts a new and powerful dimension on this, as participating systems can send real-time information about hosts that send mail to us.

OpenBSD spamd's database stores the time entries were first whitelisted as well as how many connections have been seen to pass to the real mailservers - this can be seen in the `spamdb` output.

We want to avoid putting "one off" servers - potentially spam software that retries connections, in the spamd bypass list - So instead what we do is we feed BGP a list of whitelist connections that have been around considerably longer than our expiry period (of 30 days), and who have made more than a trivial number of smtp connections. At the moment we have chosen any host that has been whitelisted for more than 75 days, and who has made at least 10 successful smtp connections. These hosts we share out in BGP so that they can be used by Client systems to bypass spamd checks as they can be considered "trusted".

The power of this is then "real" mailservers who frequently exchange mail with one participant will not see greylisting delays with other participants who share the same bypass list distributed by BGP - effectively a well chosen bypass list coming from BGP amounts to a dynamic list of established mailservers communicating with all participants contributing to the list.

In our sample configuration, Client systems WILL NOT be allowed to make changes to the distributed lists. The Route Server WILL reject and ignore all submitted routes from Client systems, and all Spamd Sources are configured to reject connections from unknown systems. Additionally, the connections between the Spamd Sources and the Route Server are protected with a variety of extra mechanisms, to further prevent generic BGP hijacking or TCP layer attacks. These policies are to guarantee the integrity of the IP address lists, and to make sure that incorrect information is not distributed.

3.4 Blacklist Source Selection Criteria

In selecting our sources for addresses to use for our blacklist, the authors chose to be very conservative. All upstream Blacklist Sources ONLY select IP addresses that have sent emails to specific spamtrap addresses or have otherwise been marked as spam by the mail server administrator. These IP addresses are automatically expired after 24 hours. If the same IP address attempts to send an email to a spamtrap address during that time, the time counter will be reset.

While manual adding of addresses is possible, this is generally avoided.

The list of Spamd sources was selected by the authors, to be trusted systems with personally known administrators. The authors are concerned about invalid or malicious information being added to the network so care has been made that all information injected into this network will be based on trusted information.

All IPv4 addresses marked with the BLACK list community are /32 routes, which are IPv4 host-specific routes. This prevents us from accidentally blocking systems that have not actually sent us any spam, but may be a network "neighbour" of a spammer. This is enforced both on the Spamd source, and the Route Server.

As a side note, many of these sources are also found in the default OpenBSD `/etc/mail/spamd.conf` configuration file, and are generally well-known to the OpenBSD community.

3.5 Transmission Path from Spamd Source to Clients

When a Spamd Source wishes to add a specific IP address to the distributed whitelist, they run the equivalent of this command:

```
bgpctl network add 192.0.2.55/32 community $AS:42
```

where `192.0.2.55` is the desired IP address, and where `$AS` is the AS number assigned to this Spamd Source.

Once this address is added to the system, the Spamd Source BGP process will see the new route, and tell all of its peers (including the Route Server) about this new route. When the Route Server receives this, it will then also notify all of its peers, including the Client systems. A Client system will receive it, and use the "`match ... set pftable`" filter rule to add the IP address to the appropriate PF table.

A script to update the lists distributed by BGP on the Spamd Source is available in the Appendix.

4 Sample client configuration

Before we show a sample Client system configuration, the authors wish to show a sample of the output of `bgpctl show rib detail` for a single entry, as the information contained is the basis of our filtering. This example is a host route for `192.0.2.55` from `AS 65043`. The "Communities" entry shows that it has been marked with the "`65066:42`" and the "`65043:42`" Communities, which means we can make decisions based on one or both of them.

```
BGP routing table entry for 192.0.2.55/32
    65043
    Nexthop 198.18.0.191 (via ???) from 203.0.113.113 (203.0.113.113)
    Origin IGP, metric 0, localpref 100, weight 0, external
    Last update: 02:10:26 ago
    Communities: 65066:42 65043:42
```

4.1 Sample client pf.conf

This section is used to declare what filters will be applied to the various lists in PF. In this sample configuration, we will add a rule for the WHITE list `bypass`, to the default spamd(8) ruleset. In this sample, the default spamd(8) ruleset is indented for easy identification.

```
        table <spamd-white> persist
        # local bypass file.
        table <nospamd> persist file "/etc/mail/nospamd"

 # new bypass file from BGP.
 table <bgp-spamd-bypass> persist

 # we add this line
 pass in quick log on egress proto tcp from <bgp-spamd-bypass> to any port smtp
 # everything else goes to spamd

         # Exiting spamd(8) configuration
         pass in quick on egress proto tcp from <nospamd> to any port smtp
         pass in quick log on egress proto tcp from <spamd-white> to any port smtp
         pass in quick on egress proto tcp from any to any port smtp \
             rdr-to 127.0.0.1 port spamd
         pass out log on egress proto tcp to any port smtp
```

4.2 Sample client bgpd.conf

This section is used to connect to the Route Server and fetch the lists. After the lists are fetched, a filter is used to add and remove the IP addresses to the specified PF tables.

/etc/bgpd.conf

```
# Begin bgpd.conf

spamdAS="65066"

AS 65001
fib-update no   # Do not change the routing table

group "spamd-bgp" {
        remote-as $spamdAS
        multihop 64
        enforce neighbor-as no

        # rs.bgp-spamd.net
        neighbor 81.209.183.113
        announce none
}

# 'match' is required, to remove entries when routes are withdrawn
match from group spamd-bgp community $spamdAS:42  set pftable "bgp-spamd-bypass"

# EOF
```

4.3 Using spamd.conf to block BLACK list entries

A naive implementation can simply use PF to block BLACK list entries. This has the obvious disadvantage that any host that is blocked, will not know it is being blocked and can simply assume that the destination system is offline. Additionally, there will not be any way for the sender to know it is being blocked on purpose. This information is necessary for several corner cases, where a mail server triggered the BLACK list entry, but is still legitimate. In such a case, telling the sending server that they are on the BLACK list, allows for the administrator to use alternate means of contact to explain why their system should not be blocked.

For these reasons, the authors strongly recommend that Client systems use `spamd.conf`[10] as a means to inform systems on the BLACK list, that they are - in fact - blacklisted.

4.3.1 Blocking of Combined BGP blacklists

Here is a method to simply block all addresses on the BLACK list. This has the advantage of being simple for the Client system administrator, however it require that the Client system administrator determine the reasons why any address was blocked.

Below is a simple script for cron to update BLACK lists. It will print the host IP address of each entry to the `/var/mail/spamd.black` file, then run `spamd-setup` (which will be configured next). Here, we use the fact that the Route Serve marks all distributed BLACK list routes with the Community string `65066:666`. It is designed to be run from `cron` at a frequency faster than the greylist pass time, (for example, every 20 minutes) so that the trapped lists are updated on a regular basis.

```
   /usr/local/sbin/bgp-spamd.black.sh

#!/bin/sh
AS=65066

bgpctl show rib community ${AS}:666 | awk '{print $2}' | \
        sed 's/\/.*$//' > /var/mail/spamd.black

/usr/libexec/spamd-setup

# EOF
```

And a `spamd.conf` configuration file, for `spamd-setup`.

```
   /etc/mail/spamd.conf

# Configuration file for spamd.conf

all:\
        :bgp-spamd:

bgp-spamd:\
         :black:\
         :msg="Your address %A has sent mail to a spamtrap\n\
          within the last 24 hours":\
         :method=file:\
         :file=/var/mail/spamd.black:

# EOF
```

4.3.2 Separation of BGP blacklists

Here is a method to use the Communities attribute to separate the blacklists into their original sources. This method has the advantage of informing the sender exactly which list they were listed on so the sender can contact the originator of the filter list, instead of every mail administrator using these lists. However, the main disadvantage of this is that the Client systems will need to know some internal information about the BGP network, and keep an up-to-date list in their `spamd.conf` and helper scripts.

In this example script, the ASes "65042", "65043" and "65513" are used instead of the Route Server's AS of "65066". This is so we can split out these specific upstream Spamd Source that are providing the information. By specifically enumerating which lists that will be used, this will explicitly ignore any additional ASes that may be providing us with BLACK list hosts. Users will need to adjust their scripts for local preferences.

The following script is appropriate to update `spamd` from the BGP BLACK list entries. It is designed to be run from `cron` at a frequency faster than the greylist pass time, (for example, every 20 minutes) so that the BLACK lists are updated on a regular basis.

/usr/local/sbin/bgp-spamd.black.sh

```
#!/bin/sh

for AS in 65042 65043 65513; do
    bgpctl show rib community ${AS}:666 | awk {'print $2}' | \
        sed 's/\/.*$//' > /var/mail/spamd.${AS}.black
done

/usr/libexec/spamd-setup

# EOF
```

And a spamd.conf configuration file, for spamd-setup(8).

/etc/mail/spamd.conf

```
# Configuration file for spamd.conf

all:\
        :bgp65042:bgp65043:bgp65513:

bgp65042:\
         :black:\
         :msg="Your address %A has sent mail to a foad.obtuse.com spamtrap\n\
          within the last 24 hours":\
         :method=file:\
         :file=/var/mail/spamd.65042.black:
bgp65043:\
         :black:\
         :msg="Your address %A has sent mail to a ualberta.ca spamtrap\n\
          within the last 24 hours":\
         :method=file:\
         :file=/var/mail/spamd.65043.black:
bgp65513:\
         :black:\
         :msg="Your address %A has sent mail to a bsdly.net spamtrap\n\
          within the last 24 hours":\
         :method=file:\
         :file=/var/mail/spamd.65513.black:

# EOF
```

4.4 Non-OpenBSD Clients

While this paper focuses on using OpenBSD for all 3 types of systems, non-OpenBSD clients can also use these lists for their own anti-spam systems. While specific examples will not be discussed here, any user will need to filter based on BGP Communities, and insert/remove those addresses into their preferred system - provided their configuration does not alter the routes of the Client system.

Any reimplementation of this network can be done, as long as the Blacklist Source and Route Server systems are able to add large amounts (150k +) of arbitrary IP host-nets with specific BGP Communities. This is normal and very common when administering BGP networks, and should be possible with nearly all BGP server implementations.

4.5 Possible Risks to Client systems

While the risks of this configuration are minimal, there are still some possible issues.

1. Use of system resources. On a test system ran by one of the authors, the current (as of 2013-02-08) list of 103k WHITE entries, and 50k BLACK list entries, the `bgpd` process uses 43.5M of memory, and the `bgp-spamd-bypass` PF table is using approx 16M of memory. This can be a problem for low memory machines.

2. When the `bgpd` process ends, it will empty the `bgp-spamd-bypass` PF table and no longer update the spamd.conf BLACK list files. This will cause the amount of whitelisted systems to return to only what has been seen locally, and the age of the BLACK list entries will quickly grow stale and invalid. The Authors recommend that Client systems monitor and ensure that the `bgpd` is running.

5 Sample Route Server configuration

Here we describe an example configuration for the Route Server. In it, we connect to two Spamd Source systems, and we also provide access for Client systems. Connections to the Spamd Sources are protected. For the Spamd Source peer "upA", TCP MD5 signatures are used. For connections to Spamd Source peer "downB", we will use IPsec with dynamic keying. The `OpenBGPd` daemon will set up the flows, and uses `isakmpd(8)`[11] to manage the session keys.

bgpd.conf follows on next page

```
myAS="65066"

AS $myAS
router-id 203.0.113.113
fib-update no
nexthop qualify via default
transparent-as yes
socket "/var/www/logs/bgpd.rsock" restricted
socket "/logs/bgpd.rsock" restricted

group blacklist-sources {
        multihop 64
        announce none

        # Neighbor upA - John Q Public - john.public@example.com
        neighbor 198.51.100.198 {
                dump all in "/tmp/upA-all-in-%H%M" 3600
                descr "upA"
                remote-as 65198
                tcp md5sig key deadbeef
        }
        # Neighbor downB - Mike Bolton - mike@bolt.example.net
        neighbor 198.18.0.191 {
                dump all in "/tmp/downB-all-in-%H%M" 3600
                descr "downB"
                remote-as 65191
                ipsec ah ike
        }
}

group RS {
        announce all
        set nexthop no-modify
        enforce neighbor-as no
        announce as-4byte no
        multihop 64
        ttl-security no
        holdtime min 60
        softreconfig in no
        maxprefix 1 restart 5

        neighbor 0.0.0.0/0 {  passive  }
}

deny from any
allow from group blacklist-sources
allow to any

# Ensure that an IP to be blacklisted is only a host entry
deny from group blacklist-sources inet  \
        community neighbor-as:666 prefixlen < 32
deny from group blacklist-sources inet6 \
        community neighbor-as:666 prefixlen < 128

# Set my own community, so clients have an easy way to filter
match from group blacklist-sources \
        community neighbor-as:666 set community $myAS:666
```

6 Security Concerns

This is a completely different BGP network from the global routing table. Any routes added here will not be visible to or modify the global routing table, and will not prevent any IP traffic from flowing. Using the included configurations will not change the routing table of any connected system, and will not attempt to steal or re-route any traffic. Additionally, routes that are delivered to the Client systems are configured in such a way that routes will not be accepted by the kernel, because the desired `nexthop` will not be directly reachable.

These routes are intended to be added to the PF table as a way to assist `spamd` in being more effective - both in catching hosts that should not make their way through greylisting via a short term block list, and by allowing real mailservers through via a `bypass` pf table to avoid greylisting delays. These routes are not intended for use to restrict general connectivity.

6.1 Security Concerns for Spamd Sources

The sources for `spamdb` information used must be trusted to provide only valid information to the participants. It must be agreed ahead of time what the selection criteria for a "real" mailserver is - as well as what the criteria for trapping hosts is.

If a participant in the system is trapping hosts long term, this will affect all participating sites (for example, if one participant summarily decides `google.com` is evil and always traps their hosts, all people using their traplist to blacklist hosts will be affected). Similarly, if a participant in the system does not apply a sufficient degree of care to ensure entries published to the bypass list are "real" smtp servers - you run the risk of more spam leakage coming through.

Additionally, all sources must themselves be kept secure. Someone with nefarious intent who can manipulate one of the participant's BGP servers can easily publish erroneous information to either prevent mail from coming through at all, or allow lots of spam to bypass spamd. (Imagine if a participant were to advertise `0.0.0.0/0` as a member of the bypass or trapped lists.)

In order to protect the integrity of the IP address lists, it is recommended that Spamd sources protect their BGP sessions to the Route Servers with security features such as TCP MD5 signatures or IPSec tunnels.

6.2 Security Concerns for Route Servers

The Route Servers MUST NOT accept a `default` or `0.0.0.0/0` route from Spamd Sources. The Route Server needs to ensure that the entire internet is not either WHITE listed, nor BLACK listed.

The Route Server SHOULD NOT accept non-host routes from Spamd Sources. The authors strongly recommend that each and every host to be BLACK listed or WHITE listed instead be explicitly enumerated. Local implementations may adjust this for their own needs, but the authors recommend that sites be conservative with their lists, to allow actual behaviour dictate which list a host should be on.

The Route Server MUST NOT accept any entries from Client systems. Client systems are unknown and untrusted, and the administrator does not know of the quality of their lists.

Like all publicly facing systems, Route Servers SHOULD install security patches and be generally kept up to date.

6.3 Security Concerns for Clients

Client systems SHOULD NOT modify local routing table based on received routes and SHOULD block only `smtp` traffic based on the received routes. The ability to ping or connect to a website on the same IP address as a BLACK list host is valuable, as well as emailing the administrator of the BLACK list host. Additionally, WHITE entries

should not modify the routing table, as we are only listing host IP addresses, and not a "better" route for these hosts.

7 Future Work

Future work on this topic include collecting statistics about the addresses being distributed, as well as the churn rate. Additionally, work on simplifying inserting the traplist addresses into spamd(8) is desired.

8 Acknowledgements

Many thanks to Peter N.M. Hansteen of BSDly.net, Bob Beck of obtuse.com, and the University of Alberta at ualberta.ca for being sources of spamdb information.
Thanks to Claudio Jeker and Stuart Henderson for assistance with BGP design and behaviour.

9 Availability

This paper, as well as client configuration examples are available at

http://www.bgp-spamd.net/

It is planned that the reference server "rs.bgp-spamd.net" will provide this service for the entirety of calendar year 2013. In December 2013, there will be an announcement about the future status of this project.

Please note that this reference server is an experiment and is subject to modification and closing. The authors will attempt to provide reasonable notice before closing the list, however no guarantees can be made.

An announcement mailing list will be available via the website, for important Client system announcements, as well as the general status of the reference implementation.

Appendix

The following is appropriate to update the BGP source from spamd output. It is designed to be run from cron at a frequency faster than the greylist pass time, (for example, every 10 minutes) so that the trapped lists are updated before machines are passed on other sites.

```
#!/usr/bin/perl
# Copyright (c) 2013 Bob Beck <beck@obtuse.com>  All rights reserved.
#
# Permission to use, copy, modify, and distribute this software for any
# purpose with or without fee is hereby granted, provided that the above
# copyright notice and this permission notice appear in all copies.
#
# THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
# WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
# MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
# ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
# WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
# ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
# OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

# perl to parse spamdb output and deal with bgpd. can take multiple
# spamdb output from multiple mail servers on command line, talks
# to bgpd on localhost to update whitelist.
```

```
# for typical use from a number of spamd collectors collect spamdb output
# in a number of files, and then run this script on the bgpd machine with
# all the files as command line args to the script.

# so typical use from cron is either
#
# ssh machine1 "spamdb" > machine1.spamdb
# ssh machine2 "spamdb" > machine2.spamdb
# bgpspamd.perl machine1.spamdb machine2.spamdb
#
# If bgpd is not running on the spamd machine, or if spamd and bgpd are
# only running on the same machine
#
# spamdb | bgpspamd.perl
#

use integer;

# AS to use - please change from this value.
my $AS = 65535;
# community string for spamd bypass - where trusted whitelist entries get sent.
my $CS = 42;
# community string for spamd traps - where we send traps.
my $TCS = 666;

# These two control how we pick only good whitelist entries to
# recommended for spamd bypass - we want to only send things we are
# relatively certain are "real" mailservers - not leakage from
# spam software that retries. for this to be effective we do have
# to be logging real connecitons to our mailservers and noticing
# them with spamlogd.
#
# Only distribute white entries that are older than 75 days
my $agelimit = 3600 * 24 * 75;
# Only distribute white entries that have made more than 10 smtp connections.
my $maillimit = 10;

my %ips;
my %tips;
my $key;

while (<>) {
    my $now = time();
    if (/^WHITE/) {
        chomp;
        my @line = split(/\|/);
        if (($line[5] < $now - $agelimit) && $line[8] > $maillimit) {
            $ips{"$line[1]"}=1;
        }
    } elsif (/^TRAPPED/) {
        chomp;
        my @line = split(/\|/);
        $tips{"$line[1]"}=1;
    }
}
open (BGP, "bgpctl show rib community $AS:$CS|") || die "can't bgpctl!";
while (<BGP>) {
    if (/^AI/) {
        chomp;
        my @line = split;
        my $ip = $line[1];
```

```
        $ip =~ s/\/.*$//;
        $ips{$ip}-=1;
    }
}
close(BGP);

open (BGP, "bgpctl show rib community $AS:$TCS|") || die "can't bgpctl!";
while (<BGP>) {
    if (/^AI/) {
        chomp;
        my @line = split;
        my $ip = $line[1];
        $ip =~ s/\/.*$//;
        $tips{$ip}-=1;
    }
}

close(BGP);

foreach $key (keys %ips) {
    if ($ips{$key} > 0) {
        system "bgpctl network add $key community $AS:$CS > /dev/null 2>&1\n";
    } elsif ($ips{$key} < 0) {
        system "bgpctl network delete $key community $AS:CS > /dev/null 2>&1\n";
    }
}

foreach $key (keys %tips) {
    if ($tips{$key} > 0) {
        system "bgpctl network add $key community $AS:$TCS > /dev/null 2>&1\n";
    } elsif ($tips{$key} < 0) {
        system "bgpctl network delete $key community $AS:$TCS > /dev/null 2>&1\n";
    }
}
```

Notes

[1] spamd(8), spamd - spam deferral daemon, OpenBSD manual pages
[2] spamdlogd(8), spamlogd - spamd whitelist updating daemon, OpenBSD manual pages
[3] RFC4271, Y. Rekhter, T. Li, and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", January 2006
[4] pf(4), pf - packet filter, OpenBSD manual pages
[5] spamdb(8), spamdb - spamd database tool, OpenBSD manual pages
[6] RFC 822, Postel, J., "SIMPLE MAIL TRANSFER PROTOCOL", August 1982
[7] spamd-setup(8), spamd-setup - parse and load file of spammer addresses, OpenBSD manual pages
[8] bgpd(8), bgpd - Border Gateway Protocol daemon, OpenBSD manual pages
[9] connecting IP address, HELO/EHLO, envelope-from, and envelope-to of the connection
[10] spamd.conf(5), spamd.conf - spamd configuration file, OpenBSD manual pages
[11] isakmpd(8), isakmpd - ISAKMP/Oakley a.k.a. IKEv1 key management daemon, OpenBSD manual pages