ROBERT HASKINS

ISPadmin

Robert Haskins has been a UNIX system administrator since graduating from the University of Maine with a B.A. in computer science. Robert is employed by Renesys Corporation, a leader in real-time Internet connectivity monitoring and reporting. He is lead author of *Slamming Spam: A Guide for System Administrators*.

rhaskins@usenix.org

THIS ARTICLE IS BASED ON THE NEW book *Slamming Spam: A Guide for System Administrators* (ISBN 0-13-146716-6) by Robert Haskins and Dale Nielsen. This material is taken from Chapter 12 and is identical to the Camram section in the book, except that Figures 4 through 8 have been deleted for space reasons.

Camram is a "sender verification" system, similar to challenge/response systems TMDA and ASK. It has a very nice Web-based interface to CRM114 in addition to its native sender verification functionality. The idea is any message that is not from a sender who computes a certain algorithm (using a Hashcash) is processed through CRM114. Any message that doesn't have the computation result in the headers must be analyzed by CRM114.

While sender verification is controversial within the anti-spam community, these types of systems are useful to some people. Camram might be used in any installation that desired a graphical, Web-based interface to CRM114. It also could be used at a site where additional protection beyond traditional header/content analysis (such as SpamAssassin or bogofilter) was desired. If enough email originators use sender compute headers, impact on recipient Camram email infrastructure would be reduced, due to the fact that those messages with sender compute headers bypass the more resource-intensive CRM114 checks.

For more information on Camram, see *http://www.camram.org*.

Camram

The reason for Camram's original implementation was as a reference implementation for a sender compute system, namely Hashcash. Although this is still a large part of the goal, Camram has tight integration with the CRM114 spam classifier. It also contains a graphical user interface to manage itself and the CRM114 application as well. Camram is worth implementing just for the ease of use it provides in managing CRM114.

Camram can be set up as an invisible proxy between your existing MTA and email systems that want to send your users email. This eliminates the need to

run Camram on your existing (perhaps overly loaded) email systems. Camram refers to this setup as the interception method. You should be aware that Camram is still a work-in-progress. Some of the functionality doesn't work precisely as expected, but it should be suitable for most situations. Be sure to check the Camram Web site often for code updates.

INBOUND MESSAGES

You can deploy Camram in two different ways in your inbound email infrastructure. The first way is by using procmail to redirect incoming messages, in a setup where Camram is run on the same machine as the end user mailboxes. This is the setup we cover here.

The second method that can be used is interception. This method "intercepts" the SMTP port 25 connection and redirects it to the Camram server, which processes the message and sends it to the mailbox. The interception method is used in a situation where your organization's email system is distributed into machines that perform the email relay function and servers that house mailboxes. Another case is when your primary server is Exchange/Domino, where you cannot run Camram directly on the mail server. Implementing an anti-spam solution such as Camram on a separate system helps to distribute the load on machines outside of your regular mail machines.

In either case, the actual processing of messages is the same, regardless of whether the procmail or interception methods are used. Figure 1 shows the flow of messages through the Camram system.

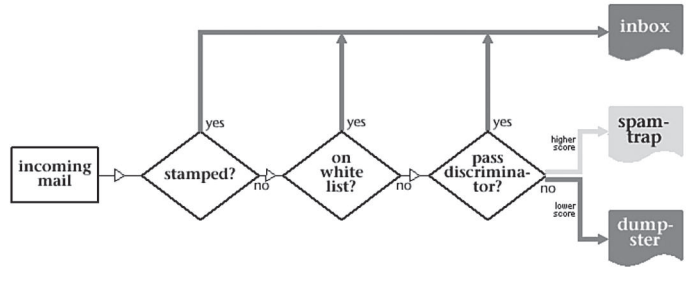

FIGURE 1.

Camram inbound message flow. (From http://www.camram.org; courtesy of Keith Dawson, dawson@worlds.std.com; used with permission.)

OUTBOUND MESSAGES

Messages leaving the Camram system must be stamped to show that they have been processed through the Hashcash computational system (see Figure 2). This is done as a proxy, using the EmailRelay software. The message is reinjected into the MTA on port 30025.

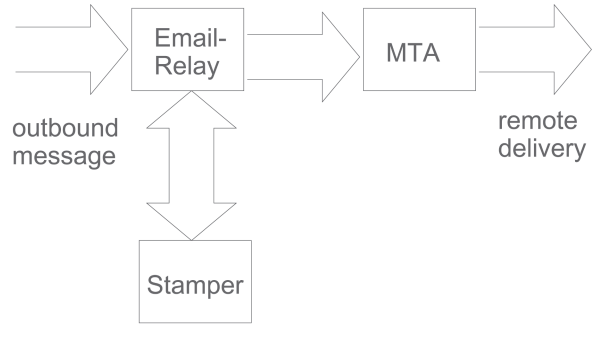

FIGURE 2.

Camram outbound message flow.

Installation

Camram can be downloaded from http://www.camram.org/download.html. We cover Camram version 0.3.25 here. The build script downloads all of the needed components, including:

- TRE—Regular Expression matching library required by CRM114
- CRM114—The Controlled Regular Expression Mutilator covered in Chapter 8, "Bayesian Filtering"
- Hashcash—Implements the sender compute algorithms required by Camram
- EmailRelay—MTA used by Camram to implement its message stamper functionality
- normalizemime—Used by CRM114 to convert MIME-encoded text

These are external packages that Camram requires for operation. Camram will download and install Python if it is not available on the system or if it is not at the correct version level when you run the buildit.sh script (shown next). After downloading, become root, extract the files, add the Camram group and user, and run the build script like this (the downloaded installation is assumed to be /usr/local/src/raging_dormouse-0.3.25.tar.gz):

```
bash$ sudo su
# mkdir /usr/local/src/camram-0.3.25
# cd /usr/local/src/camram-0.3.25
# groupadd camram
# useradd -g camram -m -d /usr/local/camram camram
# tar xzvf ../raging_dormouse-0.3.25.tar.gz
# mv raging_dormouse-0.3.25/* .
# bash buildit.sh
```

You may need to restart the download script if a download error takes place. The raging_dormouse release will exit the build process if there is a checksum error in one of the components. The build script will make sure that the appropriate third-party applications have been downloaded before continuing on.

After the initial setup script has been run, several additional steps need to take place. These actions include:

- Setting up the Camram GUI for use under Apache
- Setting up the MTA (Sendmail) to work with Camram
- Configuring a Procmail recipe for use with Camram

APACHE INSTALLATION

Next, install the Camram hooks for the Apache Web server. The installer attempts to copy the configuration to the Apache configuration directory on some Linux distributions, namely /etc/httpd/conf. If this is not how Apache is set up on your system (for example, Debian), then copy the configuration file manually to the Apache configuration directory and restart Apache like this:

```
# cp -p /usr/local/camram/ancillary/camram.conf/etc/apache/
  camram.conf
# /etc/init.d/apache restart
```

SENDMAIL (MTA) INTEGRATION

Integrating Camram with Sendmail requires setting up Sendmail to listen on three IP addresses and ports: we use 127.0.0.1 port 25, 127.0.0.1 port 30025, and the publicly available inbound interface. Any available IP and port combination can be used, but these are what Camram recommends, so they are the ones we use.

If you set up Sendmail per our examples in other parts of this book, sendmail.mc is located in /usr/local/src/sendmail-8.12.11/cf/cf/. If your current configuration is sendmail.cf, then edit your sendmail.mc file and add the following three lines, replacing 192.168.16.9 with the public IP address of your Camram machine that accepts email from the Internet:

```
DAEMON_OPTIONS('Port=smtp,Addr=192.168.16.9, Name=MTA')dnl
DAEMON_OPTIONS('Port=smtp,Addr=127.0.0.1, Name=MTA')dnl
DAEMON_OPTIONS('Port=30025,Addr=127.0.0.1, Name=MTA')dnl
```

These lines tell Sendmail to listen to port 25 on its public IP address and localhost address (127.0.0.1), as well as 30025 on localhost for reinjecting messages into the MTA. Then rebuild sendmail.cf, install it (saving the old one), and restart Sendmail:

```
bash$ sudo su
# cd /usr/local/src/sendmail-8.12.11/cf/cf/
# make sendmail.cf
# cp /etc/mail/sendmail.cf /etc/mail/sendmail.cf.old
# cp sendmail.cf /etc/mail/sendmail.cf
# /etc/init.d/sendmail restart
```

Camram is now integrated into your Sendmail installation for all users on the system.

PROCMAIL INTEGRATION

The code below illustrates a procmail recipe showing Camram integration. This can be specified on a per-user basis by placing the recipe in each user's .procmailrc file or in a system-wide /etc/procmailrc file.

```
MAILDIR=$HOME/Maildir
DEFAULT=$MAILDIR/
ORGMAIL=$MAILDIR/
# Directory for storing procmail configuration and log files
PMDIR=/var/log/procmail
# Put ## before LOGFILE if you want no logging (not recommended)
LOGFILE=$PMDIR/log
# Set to yes when debugging
VERBOSE=no
# Remove ## when debugging; set to no if you want minimal logging
## LOGABSTRACT=all
# Replace $HOME/Msgs with your message directory
# Mutt and elm use $HOME/Mail
# Pine uses $HOME/mail
# Netscape Messenger uses $HOME/nsmail
# Some NNTP clients, such as slrn & nn, use $HOME/News
# Mailboxes in maildir format are often put in $HOME/Maildir
#MAILDIR=/var/spool/spamtrap      # Make sure this directory exists!
##INCLUDERC=$PMDIR/testing.rc
##INCLUDERC=$PMDIR/lists.rc
:0fw
| /usr/local/camram/bin/procmail_filter
:0
* < 2
/dev/null
```

If you are not using Maildir-formatted mailboxes, you should change the lines that read

```
DEFAULT=$MAILDIR/
ORGMAIL=$MAILDIR/
```

to be

```
DEFAULT=
ORGMAIL=
```

Camram Configuration

Besides the procmail recipe, Camram has three files that can be changed to adjust its behavior:

- /usr/local/camram/ancillary/global_configuration—Default values; we do not make any changes to this file
- /var/spool/camram/configuration—Where most site-specific changes are made to adjust Camram's functions
- /usr/local/camram/ancillary/camram.local—The email relay script used to control the parameters when sending messages from Camram

We also cover how to set up appropriate cron jobs and Camram users at the end of this section.

/VAR/SPOOL/CAMRAM/CONFIGURATION

The valid parameters in the configuration file are the same ones that are valid in the global_configuration file. The configuration file is broken down into the following sections:

Core
Spam analysis
Spam storage
Filter configuration
User email addresses

All of the changes we list next are confined to the Core section. Besides the ones we cover here, some of the parameters you should consider adjusting include any keyword involving a path or any of the CRM114-scoring thresholds. A default file with just the section headers (listed previously) is created at Camram install time. You might want to make a copy of this file before making changes to it. At a minimum, the following parameters should be defined under the Core section in order to change from their default values:

```
authorized_users = comma-list:root,esj,dale
```

This should be a comma-separated list of privileged users who can manage the server via the GUI.

```
challenge_URL_base=string:http://mydomain.com/camram/pdgen.cgi
```

This is the parameter indicating the URL address for the challenge Web page. Change "mydomain.com" to be the address of your Web server.

```
correction_URL =string:http://mydomain.com/camram/correct.cgi
```

This is the URL where users enter corrections for messages misclassified as spam or ham.

```
reinjection_SMTP_port = string:30025
```

This is the port where Camram sends messages back to the MTA. If you used our example, leave this at 30025.

```
central_administration = boolean:0
```

This controls whether end users have access to the CRM114 retraining (0) or only the administrator has access to retraining (1). We recommend setting this to 0 so that end users can train their own filters.

```
password_key=string:notswordfish
```

This is the key used for the private password mechanism. Be sure to change it!

```
log_level=integer:1
```

The default logging level is 1. This value can be anything from 0 to 9, where 0 is no output and 9 is very verbose. Unless you are troubleshooting a problem, 1 should be acceptable. Messages are logged in /var/log/messages.

/USR/LOCAL/CAMRAM/ANCILLARY/CAMRAM.LOCAL

The camram.local file is the script that starts up the email forwarder program, EmailRelay. A few changes need to be made in this file, but before going through those, be sure to make a copy of the file as it was initially distributed:

```
# cd /usr/local/camram/ancillary
# cp -p camram.local camram.local.orig
```

This makes a backup copy of camram.local as camram.local.orig. This script is automatically read each time Camram is run, so there is no need to perform any steps to make changes to this file active. You should consider making the following changes to the parameters in this file:

```
camram_architecture=procmail
```

If you are running Camram on the same machine as the email boxes (as we are in our example), this should be procmail; otherwise, it should be set to intercept.

```
stamper_interface=ip address
```

This is the IP address of the interface that stamped messages should be accepted on. ip address should be set to the internal IP address, which accepts email from users on your local network. Do not set this to any externally available IP address or you could stamp messages for spammers!

```
filter_interface=ip address
```

This defines the interface of the server where email from the Internet originates. ip address should be set to an externally accessible interface that the MX record for your domain is set to, or a host that accepts mail for your domain.

```
local_smart_host=ip address
```

This is the machine that knows how to route email from your server/domain or if your Camram machine is behind a firewall. If your Camram machine is the smart host gateway, then set this to 127.0.0.1.

After changes are made to this file, the script must be invoked.

Executing the script /etc/init.d/camram.local start will start the script on many Linux systems.

CRON JOBS

There are three cron jobs that need to be set up to perform various tasks. Camram distributes suggested cron entries for each.

/usr/local/camram/ancillary/sweepup.py

This script deletes messages in the Camram dumpster. Be sure to run this often enough so that directory lookups don't get too slow when too many files are present.

/usr/local/camram/ancillary/clean_mail_queue.py

This script forwards feedback to the end user and should be run very often. Camram's example cron job runs every minute.

/usr/local/camram/ancillary/mbox2spamtrap.py

This script automatically scans the missed_spam_box folder for messages incorrectly classified by CRM114 and retrains it accordingly.

SETTING UP CAMRAM USERS

Each user who is going to have email (we assume all users) will need to have the appropriate directory and files set up on the system. This is accomplished by running the following script for every user on the system:

```
# /usr/local/camram/ancillary/clean_configuration.py -u username
```

You need to change username to be the user you want to set up on the system. If you are running Camram in intercept mode (without delivery to mailboxes on the machine Camram is running on), you need to create those accounts with the following command:

```
# /usr/local/camram/ancillary/new_account.py -u username
```

If you are running in procmail mode (like we are in our example), the accounts already exist as "real" UNIX users, so this step must be skipped.

After setting up your users, you must run the edit_config_cgi script to create user accounts and set up the database by going to the URL http://mydomain .com/camram/edit_config.cgi. Change "mydomain.com" to be the name of the Camram server you set previously. See the next section for using this screen.

Using Camram

Camram classifies messages into three possible categories:

- Red—Definitely spam; delivered to junk email folder
- Yellow—Possibly spam, possibly non-spam; delivered to spamtrap (possibly spam) folder
- Green—Definitely not spam; delivered to inbox

This results in an accurate classification system of messages, and it requires users to look in their spamtrap folder. Camram has a Web interface for accessing many functions. If you have followed our previous examples, the following screens are available at the listed URLs.

http://mydomain.com/camram/edit_config.cgi

The edit_config screen allows the administrator to edit the default settings for each user and should be run after adding each user, to perform initial setup.

http://mydomain.com/camram/correct.cgi

This screen allows the user to correct CRM114 misclassifications via a Web browser.

http://mydomain.com/camram/recover.cgi

The recover screen allows user access to the junk email folder (Camram calls it the dumpster) from a Web browser.

PREFERENCES

The parameters in the edit_prefs.cgi screen are stored in each user's home directory, in ~user/.camram/configuration. However, the parameters should be changed only by the preferences Web interface and not directly via editing the file, as changes will likely get overwritten. The parameters listed here are the same ones that are defined by the global_configuration file shown previously.

The defaults here are reasonable. The field labeled my_email_addresses should be updated with all aliases for each user. These addresses represent the addresses for which Camram will accept Hashcash stamps for this account.

SPAMTRAP

The correct.cgi screen allows each user to manage their spamtrap (yellow messages), which is the mail folder containing messages that Camram was unsure about when it ran the classifier on them. Simply check the checkbox on the left side for each misclassified message and click the Process button, and your messages will be sent to your inbox and the Bayesian classifier will be retrained.

RECOVER

The recover.cgi screen lists all messages in the dumpster and inbox. This screen allows you to pull false positive messages out of the dumpster and into the spamtrap for reclassification. Copies of all messages processed and classified as spam or not spam will end up in the dumpster. Do not be alarmed by the presence of non-spam emails. It is unfortunate that the dumpster contains more than just rejected spam messages because you can't just browse it quickly to identify false positives.