

# Sendmail's New GreetPause Feature

Hal Pomeranz

In most cases, Spammers are motivated to send their unsolicited emails as rapidly as possible. *Slamming* is a technique where the spammer simply fires all of the SMTP commands necessary to transmit an email message to another mail server without waiting for the normal SMTP responses from the remote machine. In most cases, the remote mail server will end up accepting the message despite the fact that the slammer is actually disobeying the SMTP behavior mandated by various Internet RFCs.

However, no well-behaved mail server should ever inject email messages in this fashion, so it would be nice if there were some way to distinguish between the malicious slamming activity and more normal SMTP traffic. The Sendmail v8.13 release train introduced the GreetPause option for doing exactly that. When GreetPause is enabled, Sendmail simply waits a specified amount of time on each new connection before emitting the standard SMTP greeting message. If the remote server starts sending SMTP commands before your mail server sends out the SMTP greeting string, then the remote side is trying to slam you. When this happens, Sendmail simply rejects all of the incoming SMTP commands and drops the message.

GreetPause is one of those extremely useful spam-fighting techniques that precisely identifies a specific type of unwanted email traffic while having minimal impact on regular SMTP sessions. As such, it's a good idea to implement this feature on all of your Internet-facing mail servers to help cut down on the amount of spam you have to deal with.

## Implementation

Turning on the GreetPause option is simple. Just add a macro like the following to your existing Sendmail macro configuration file:

```
FEATURE(`greet_pause', `1000')
```

The second argument is the number of milliseconds Sendmail should wait before sending the SMTP greeting string to the remote mail server--so here we're pausing for one second. Obviously, you don't want to set this value too high because you are introducing a delay on each new SMTP connection. Something in the range of 1-5 seconds (1000-5000 milliseconds) seems appropriate.

In fact, at the same time they implemented the GreetPause option, the Sendmail developers also implemented some hooks in the standard Sendmail access DB, which allow you to specify different GreetPause delays for specific IP addresses or network ranges. This feature is most commonly used to turn off the GreetPause delay for trusted mail servers (like your internal mail relays) that may be sending mail through machines that have the GreetPause option turned on.

The trick is that you must declare `FEATURE(`greet_pause', ...)` in your macro configuration file *after* your normal access DB declaration in order for these hooks to work:

```
FEATURE(`access_db', `hash -o -T<TMPF> /etc/mail/access')
FEATURE(`delay_checks', `friend')
...other configuration lines can go here...
FEATURE(`greet_pause', `1000')
```

## Sendmail's New GreetPause Feature

### Hal Pomeranz

As of Sendmail v8.13.1 if you get the declarations in the wrong order, you see an error message when you try to compile your macro definitions into a sendmail.cf file:

```
$ m4 config.mc >sendmail.cf
```

```
*** WARNING: FEATURE(`greet_pause') before
FEATURE(`access_db')-- greet_pause will not use access_db!
```

By the way, note the addition of the "-T<TMPF>" option in the access\_db declaration in the example above. This option was added (as of Sendmail v8.12) so that Sendmail will emit a temporary (4xx type) error message to the remote sender if the local access DB is unavailable for some reason. This can be very useful if you're accessing the database from a remote LDAP server that is unavailable for some reason. Also we're using the delay\_checks feature so that we can allow unfiltered email to reach some of our internal recipient addresses (like our "abuse@" addresses). delay\_checks was actually introduced in Sendmail v8.10, but the access DB syntax for this feature was actually changed as of Sendmail v8.12.

Here are a couple of examples showing the GreetPause hooks in the access DB along with the new delay\_checks syntax:

```
GreetPause:192.168 0
Spam:abuse@sysiphus.com FRIEND
```

The first line says use a GreetPause value of zero milliseconds for all connections originating from the 192.168.0.0/16 network. Assuming this is our internal network address range, it basically means we won't be holding up any outgoing email messages with useless GreetPause delays. The second line just shows the new preferred syntax that uses the delay\_checks feature to allow unfiltered email feeds to a specific email address. The older syntax ("To:abuse@sysiphus.com SPAMFRIEND") is still supported for backwards-compatibility reasons, but has been deprecated and may go away in the future.

### Testing the Configuration

Having implemented this new feature, it would be nice if we had a reliable way of testing that it was working properly. Testing is actually pretty straightforward if you have a copy of the extremely handy netcat utility lying around--most Open Source OSes come with netcat pre-installed, and the source is also readily available from the Internet.

The first step is to create a text file containing the SMTP commands typically used to send an email message to a remote mail server. Here's an example that you can use for your testing:

```
helo localhost.localdomain
mail from: hal@deer-run.com
rcpt to: hal@sysiphus.com
data
From: hal@deer-run.com
To: hal@sysiphus.com
Subject: testing
1 2 3
```

## Sendmail's New GreetPause Feature

Hal Pomeranz

```
.  
quit
```

Please change the email addresses in the sample file above so I don't get hit with a lot of backscatter spam. I thank you.

Once you've created this file on the local machine, you should be able to use it to test the GreetPause feature. Here's sample output on a machine that *doesn't* have GreetPause enabled:

```
$ cat smtp-session | nc localhost 25
```

```
220 intl.sysiphus.com ESMTP Sendmail 8.13.4/8.13.4...  
250 intl.sysiphus.com Hello [127.0.0.1], pleased to...  
250 2.1.0 hal@deer-run.com... Sender ok  
250 2.1.5 hal@sysiphus.com... Recipient ok  
354 Enter mail, end with "." on a line by itself  
250 2.0.0 k059BPhT004045 Message accepted for delivery  
221 2.0.0 intl.sysiphus.com closing connection
```

See the "Message accepted for delivery" line? Looks like our message went through just fine. Now let's try that against a machine that has the GreetPause option enabled:

```
$ cat smtp-session | nc ext1.sysiphus.com 25
```

```
554 ext1.sysiphus.com ESMTP not accepting messages  
250 ext1.sysiphus.com Hello [10.1.1.1], pleased to...  
550 5.0.0 Command rejected  
550 5.0.0 Command rejected  
550 5.0.0 Command rejected  
500 5.5.1 Command unrecognized: "From: hal@deer-run..."  
500 5.5.1 Command unrecognized: "To: hal@sysiphus.com"  
500 5.5.1 Command unrecognized: "Subject: testing"  
500 5.5.1 Command unrecognized: ""  
500 5.5.1 Command unrecognized: "1 2 3"  
500 5.5.1 Command unrecognized: "."  
221 2.0.0 ext1.sysiphus.com closing connection
```

Here we see the remote machine rejecting our SMTP commands because we didn't wait for the GreetPause timeout.

So this technique gives us a reasonable mechanism for at least verifying the GreetPause feature is working. However, remember to test from an IP address that doesn't have an exception defined for it in your access DB. If we had done the above test from a machine in the 192.168.0.0/16 network (per our previous access DB example), then it would have appeared to us as if GreetPause wasn't working when in fact all that was going on was that we had specifically disabled the GreetPause delay for hosts in this network range.

# **Sendmail's New GreetPause Feature**

**Hal Pomeranz**

## **Final Thoughts**

GreetPause is an extremely simple technique, yet extremely useful for stopping one particular part of your incoming spam volume. Of course, the folks who are writing bulk-emailing software are reasonably clever people. As more sites implement the GreetPause option, I assume that future spamming software will become more intelligent and simply wait for the SMTP greeting message before slamming in the SMTP commands required to transmit the email message. At that point I guess the good guys will have to retaliate by introducing timeout delays at each stage of the SMTP communication. This, of course, will slow down legitimate email transmission to some degree. And people wonder why I hate spammers so much.