The Open Source Solution to SPAM

Covers SpamAssassin Version 3.0

SpamAssassin

O'REILLY®

Alan Schwartz

O'REILLY®

SpamAssassin

Want to work from home and get paid well? Everyday can be payday! Perhaps you'd prefer to PROTECT YOUR PET FROM FLEAS AND TICKS? Tired Of Online Dating? Meet Someone Real! Tired with life? Buy Xanax!

Spam. The scourge of the Internet. It fills our inboxes, burns our bandwidth, brings raw and licentious images into the sanctity of our homes, and is single-handedly responsible for almost totally destroying the utility of electronic mail. How do you fight this rising electronic tide of email sewage? For many, the answer lies in a free and open source tool known as SpamAssassin.

SpamAssassin is perhaps the most widely deployed anti-spam tool on the Internet today. Let Alan Schwartz, experienced mail administrator, show you how to apply this powerful tool.
You'll learn how to:

- Customize SpamAssassin's rules, and even create new ones
- Train SpamAssassin's Bayesian classifier, a statistical engine for detecting spam, to optimize it for the sort of email that you typically receive
- Block specific addresses, hosts, and domains using third-party blacklists like the one maintained by Spamcop.net
- Whitelist known good sources of email, so that messages from clients, coworkers, and friends aren't inadvertently lost
- Configure SpamAssassin to work with newer spam-filtering methods such as Hashcash (*www.hashcash.org*) and Sender Policy Framework (SPF)

Alan also shows how to install SpamAssassin in a variety of different configurations. SpamAssassin integrates with all major mail transport and delivery agents, including the venerable sendmail, procmail, postfix, qmail, and Exim.

International cell calls for a nickel? Forget that. SpamAssassin is free. And even better, SpamAssassin has proven to be effective. Buy this book. Install SpamAssassin. Take back your inbox.

"Detailed, accurate, and informative—recommended for spam-filtering beginners and experts alike."

—Justin Mason, SpamAssassin development team

www.oreilly.com

US $24.95 **CAN $36.95**

ISBN: 978-0-596-00707-2

9 780596 007072 52495

SpamAssassin

Other resources from O'Reilly

Related titles

- DNS and Bind
- TCP/IP Network Administration
- Essential System Administration
- LDAP System Administration
- Essential SNMP
- Network Security Assessment
- Network Security Hacks
- Network Security with OpenSSL
- Managing Security with Snort and IDS Tools

oreilly.com

oreilly.com is more than a complete catalog of O'Reilly books. You'll also find links to news, events, articles, weblogs, sample chapters, and code examples.

oreillynet.com is the essential portal for developers interested in open and emerging technologies, including new platforms, programming languages, and operating systems.

Conferences

O'Reilly brings diverse innovators together to nurture the ideas that spark revolutionary industries. We specialize in documenting the latest tools and systems, translating the innovator's knowledge into useful skills for those in the trenches. Visit *conferences.oreilly.com* for our upcoming events.

Safari Bookshelf (*safari.oreilly.com*) is the premier online reference library for programmers and IT professionals. Conduct searches across more than 1,000 books. Subscribers can zero in on answers to time-critical questions in a matter of seconds. Read the books on your Bookshelf from cover to cover or simply flip to the page you need. Try it today with a free trial.

SpamAssassin

Alan Schwartz

O'REILLY®
Beijing • Cambridge • Farnham • Köln • Paris • Sebastopol • Taipei • Tokyo

SpamAssassin
by Alan Schwartz

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*safari.oreilly.com*). For more information, contact our corporate/institutional sales department: (800) 998-9938 or *corporate@oreilly.com*.

Editor:	Jonathan Gennick
Production Editor:	Darren Kelly
Cover Designer:	Ellie Volckhausen
Interior Designer:	Melanie Wang
Production Services:	Nancy Crumpton

Printing History:

July 2004: First Edition.

This book uses RepKover™, a durable and flexible lay-flat binding.

ISBN: 0-596-00707-8
[M] [11/04]

Table of Contents

Preface

If you use email, it's likely that you've recently been visited by a piece of spam—an unsolicited, unwanted message, sent to you without your permission.* If you manage an email system, it's almost certain that you've had to help your users avoid the deluge of unwanted email.

System administrators pay for spam with their time. The Internet's email system was designed to make it difficult to lose email messages: when a computer can't deliver a message to the intended recipient, it does its best to return that message to the sender. If it can't send the message to the sender, it sends it to the computer's postmaster—because something must be seriously wrong if both the email addresses of the sender and the recipient of a message are invalid.

The well-meaning nature of Internet mail software becomes a positive liability when spammers come into the picture. In a typical bulk mailing, anywhere from a few hundred to tens of thousands of email addresses might be invalid. Under normal circumstances these email messages would bounce back to the sender. But the spammer doesn't want them! To avoid being overwhelmed, spammers often use invalid return addresses. The result: the email messages end up in the mailboxes of the Internet postmasters, who are usually living, breathing system administrators.

System administrators at large sites are now receiving hundreds to thousands of bounced spam messages each day. Unfortunately, each of these messages has to be carefully examined, because mixed in with these messages are the occasional bounced mail messages from misconfigured computers that actually should be fixed.

As the spam problem grows worse and worse, system administrators are increasingly taking themselves off their computers' "postmaster" mailing lists. The result is predictable: they're deluged with less email, but problems that they would normally dis-

* Spam is also a registered trademark of Hormel Foods, which uses the word to describe a canned luncheon meat. In this book, the word "spam" is used exclusively to refer to Internet spam and not the meat.

cover by receiving postmaster email are being missed as well. The Internet as a whole suffers as a result.

Although there are many important ways to reduce spam—including obscuring email addresses, complaining to spammers' service providers, and seeking legal and legislative relief—few remedies are as immediately effective as filtering email messages on the basis of content and format, and few filtering systems are as widely used and well maintained as SpamAssassin™.

This book is for mail system administrators, network administrators, and Internet service providers who are concerned about the growing toll that spam is taking on their systems and their users and are looking for a way to regain some control or reduce the burden on their users.

Scope of This Book

This book is divided into nine chapters and one appendix. The first four chapters deal with core SpamAssassin concepts that are independent of the underlying mail system.

Chapter 1, *Introducing SpamAssassin*
: Explains what SpamAssassin does, and provides a conceptual overview of its organization and features.

Chapter 2, *SpamAssassin Basics*
: Covers the installation, testing, and basic operation of SpamAssassin.

Chapter 3, *SpamAssassin Rules*
: Details the configuration of SpamAssassin, and focuses particularly on SpamAssassin's spam-detection rules. It explains how to increase or decrease the impact of rules, write new rules, and add addresses to blacklists and whitelists.

Chapter 4, *SpamAssassin as a Learning System*
: Reviews the learning features of SpamAssassin: automatic whitelisting and Bayesian filtering. It provides the theory behind these features and discusses how to configure, train, and tune them.

The remaining five chapters detail the integration of SpamAssassin with several popular mail transport agents (MTAs) to provide sitewide spam-checking. They also explain how to set up a SpamAssassin gateway to check all incoming mail before delivery to an internal mail host.

Chapter 5, *Integrating SpamAssassin with sendmail*
: Explains how to integrate SpamAssassin with the sendmail MTA, using the milter interface. As an example of this approach, the installation and configuration of MIMEDefang is described.

Chapter 6, *Integrating SpamAssassin with Postfix*
: Explains how to integrate SpamAssassin with the Postfix MTA, using the *content_filter* interface. As an example of this approach, the installation and configuration of amavisd-new, a daemonized content filter, is described.

Chapter 7, *Integrating SpamAssassin with qmail*
: Explains how to integrate SpamAssassin with the qmail MTA.

Chapter 8, *Integrating SpamAssassin with Exim*
: Explains how to integrate SpamAssassin with the Exim MTA using several different popular approaches including custom transports, exiscan, and sa-exim.

Chapter 9, *Using SpamAssassin as a Proxy*
: Explains how to set up a SpamAssassin POP mail proxy to support users who download their email with POP clients.

The Appendix lists useful resources for more information about SpamAssassin and other antispam approaches.

Versions Covered in This Book

At the time this book went to press, SpamAssassin 2.63 was the latest released version of SpamAssassin and was in wide use. The next-generation release of SpamAssassin, SpamAssassin 3.0, was available for beta-testing and is expected to be released at about the time this book appears in stores. SpamAssassin 3.0 introduces several important new features and changes parts of the Perl API.

Accordingly, this book covers both versions of SpamAssassin. When a topic or setting is specific to one version, I so note it.

Conventions Used in This Book

The following conventions are used in this book:

Italic
: Used for Unix file, directory, user, and group names and for Perl modules, objects, method names, and method options. It is also used for URLs (uniform resource locators) and to emphasize new terms and concepts when they are introduced.

`Constant Width`
: Used for Unix commands, code examples, and system output. It is also used for scripts, process names, and SpamAssassin directives.

`Constant Width Italic`
: Used in examples for variable input (e.g., a filename you must provide).

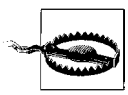
`$`

The Unix Bourne shell or Korn shell prompt.

`#`

The Unix superuser prompt. I use this symbol for examples that should be executed by root.

This icon designates a note, which is an important aside to the nearby text.



This icon designates a warning related to the nearby text.

Using Code Examples

All the code in this book is available for download from *http://www.oreilly.com/catalog/spamassassin*. See the file *readme.txt* in the download for installation instructions.

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books *does* require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation *does* require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, and publisher; for example: "*SpamAssassin*, by Alan Schwartz (O'Reilly)."

If you feel your use of code examples falls outside fair use or the permission given previously, feel free to contact us at *permissions@oreilly.com*.

Comments and Questions

We have tested and verified the information in this book to the best of our ability, but you may find that features have changed (or even that we have made mistakes!). Please let us know about any errors you find, as well as your suggestions for future editions, by writing to:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
(800) 998-9938 (U.S. and Canada)
(707) 827-7000 (international/local)
(707) 829-0104 (fax)

You can also contact O'Reilly by email. To be put on the mailing list or request a catalog, send a message to:

info@oreilly.com

We have a web page for this book, which lists errata, examples, and additional information. You can access this page at:

http://www.oreilly.com/catalog/spamassassin

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about O'Reilly books, conferences, Resource Centers, and the O'Reilly Network, see the O'Reilly web site at:

http://www.oreilly.com/

Acknowledgments

Bob Amen, Justin Mason, and Matt Riffle served as technical reviewers for this book. Any remaining errors, of course, are mine.

I have once again had the pleasure of collaborating with an excellent O'Reilly editor, Jonathan Gennick. The O'Reilly production crew for this book included Darren Kelly, Ellie Volckhausen, and Nancy Crumpton.

This book is dedicated to the developers and user community of SpamAssassin, for their fine work in helping to stem the flood of unwanted email.

Never-ending thanks to M.G. and Ari, who make it all worthwhile.

CHAPTER 1

Introducing SpamAssassin

The SpamAssassin system is software for analyzing email messages, determining how likely they are to be spam, and reporting its conclusions. It is a rule-based system that compares different parts of email messages with a large set of rules. Each rule adds or removes points from a message's spam score. A message with a high enough score is reported to be spam.

SpamAssassin was a trademark of Deersoft, and Deersoft has been acquired by Network Associates. In this book, I won't write SpamAssassin™ each time I mention it because that would be distracting, but you should assume that the trademark symbol is there.

Many spam-checking systems are available. SpamAssassin has become popular for several reasons:

- It uses a large number of different kinds of rules and weights them according to their diagnosticity. Rules that have been demonstrated to be more effective at discriminating spam from non-spam email are given higher weightings.
- It is easy to tune the scores associated with each rule or to add new rules based on regular expressions.
- SpamAssassin can adapt to each system's email environment, learning to recognize which senders are to be trusted and to identify new kinds of spam.
- It can report spam to several different spam clearinghouses and can be configured to create *spam traps*—email addresses that are used only to forward spam to a clearinghouse.
- It is free software, distributed under either the GNU Public License or the Artistic License. Either license allows users to freely modify the software and redistribute their modifications under the same terms.

Example 1-1 shows a message that has been tagged as spam by SpamAssassin. Elements added by SpamAssassin appear in bold.

Example 1-1. A message tagged by SpamAssassin

```
From riverol5380503@jubii.dk Fri Nov  7 18:26:05 2003
Received: from localhost [127.0.0.1] by localhost
        with SpamAssassin (2.60 1.212-2003-09-23-exp);
        Sun, 09 Nov 2003 12:24:22 -0600
From: "brianj" <riverol5380503@jubii.dk>
To: <Undisclosed.Recipients@mailin-2.priv.cc.uic.edu>
Subject: Live your dream life!!                MPNWSTU
Date: Fri, 07 Nov 2003 15:32:41 -0800
Message-Id: <000016646728$00007347$00000042@mail3.mailnara.net>
X-Spam-Status: Yes, hits=12.9 required=5.0 tests=CLICK_BELOW,
        FORGED_MUA_EUDORA,FROM_ENDS_IN_NUMS,MISSING_OUTLOOK_NAME,
        MSGID_OUTLOOK_INVALID,MSGID_SPAM_ZEROES,NORMAL_HTTP_TO_IP,
        SUBJ_HAS_SPACES,SUBJ_HAS_UNIQ_ID autolearn=no version=2.60
X-Spam-Flag: YES
X-Spam-Checker-Version: SpamAssassin 2.60 (1.212-2003-09-23-exp)
X-Spam-Level: **********
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="----------=_3FAE8656.371BED4D"

This is a multi-part message in MIME format.

------------=_3FAE8656.371BED4D
Content-Type: text/plain
Content-Disposition: inline
Content-Transfer-Encoding: 8bit

Spam detection software, running on the system has
identified this incoming email as possible spam.  The original message
has been attached to this so you can view it (if it isn't spam) or block
similar future email.  If you have any questions, see
the administrator of that system for details.

Content preview:  Do you owe large sums of money? Are you stuck with high
  interest ra{tes? We can help! You can do what tens of thousands of
  americans have done, consolidate your high interest bills into one
  easy, low interest, monthly payment. [...]

Content analysis details:   (12.9 points, 5.0 required)

 pts rule name              description
---- ---------------------- --------------------------------------------
 1.0 SUBJ_HAS_SPACES        Subject contains lots of white space
 4.3 MSGID_SPAM_ZEROES      Spam tool Message-Id: (12-zeroes variant)
 0.9 FROM_ENDS_IN_NUMS      From: ends in numbers
 0.2 NORMAL_HTTP_TO_IP      URI: Uses a dotted-decimal IP address in URL
 0.2 SUBJ_HAS_UNIQ_ID       Subject contains a unique ID
 4.3 MSGID_OUTLOOK_INVALID  Message-Id is fake (in Outlook Express format)
 0.1 MISSING_OUTLOOK_NAME   Message looks like Outlook, but isn't
 1.9 FORGED_MUA_EUDORA      Forged mail pretending to be from Eudora
 0.0 CLICK_BELOW            Asks you to click below

The original message was not completely plain text, and may be unsafe to
open with some email clients; in particular, it may contain a virus,
```

Example 1-1. A message tagged by SpamAssassin (continued)

```
or confirm that your address can receive spam.  If you wish to view
it, it may be safer to save it to a file and open it with an editor.


------------=_3FAE8656.371BED4D
Content-Type: message/rfc822; x-spam-type=original
Content-Description: original message before SpamAssassin
Content-Disposition: attachment
Content-Transfer-Encoding: 8bit

Received: (qmail 25515 invoked from network); 7 Nov 2003 18:26:02 -0600
Received: from mailin-2.cc.uic.edu (HELO mailin-2.priv.cc.uic.edu) (128.248.155.213)
  by email0.cc.uic.edu with SMTP; 7 Nov 2003 18:26:02 -0600
Received: from mail3.mailnara.net (c-24-98-136-187.atl.client2.attbi.com [24.98.136.187])
        by mailin-2.priv.cc.uic.edu (8.12.10/8.12.9) with ESMTP id hA80PxJk011669;
        Fri, 7 Nov 2003 18:26:00 -0600
Message-ID: <000016646728$00007347$00000042@mail3.mailnara.net>
To: <Undisclosed.Recipients@mailin-2.priv.cc.uic.edu>
From: "brianj" <riverol5380503@jubii.dk>
Subject: Live your dream life!!                MPNWSTU
Date: Fri, 07 Nov 2003 15:32:41 -0800
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="----------=_1068251164-2528-687"
X-Priority: 3
X-MSMail-Priority: Normal
X-Mailer: QUALCOMM Windows Eudora Version 5.1
X-MimeOLE: Produced By Microsoft MimeOLE V5.00.3018.1300
Content-Length: 2290
Lines: 72


Do you owe large sums of money? Are you stuck with
high interest ra{tes? We can help!

You can do what tens of thousands of americans have
done, consolidate your high interest bills into one
easy, low interest, monthly payment.

By first reducing, and then completely removing your
d+ebts, you will be able to start fresh. Why keep
dealing with the stress, headaches, and wasted money,
when you can consolidate your d+ebt and pay them off
much sooner!

Click below to learn more:

http://61.186.254.9?affiliateid=mailer10



hjeuubnfs

------------=_3FAE8656.371BED4D—
```

The SpamAssassin report is revealing. Despite the fact that this message includes several tricks to fool spam-checkers, such as random characters at the end and breaking up the words "rates" and "debt" with symbols, SpamAssassin identifies several suspicious characteristics and assigns a high spam score.

How SpamAssassin Works

There are several ways that SpamAssassin makes up its mind about a message:

- The message headers can be checked for consistency and adherence to Internet standards (e.g., is the date formatted properly?).
- The headers and body can be checked for phrases or message elements commonly found in spam (e.g., "MAKE MONEY FAST" or instructions on how to be removed from future mailings)—in several languages.
- The headers and body can be looked up in several online databases that track message checksums of verified spam messages.
- The sending system's IP address can be looked up in several online lists of sites that have been used by spammers or are otherwise suspicious.
- Specific addresses, hosts, or domains can be blacklisted or whitelisted. A whitelist can be automatically constructed based on the sender's past history of messages.
- SpamAssassin can be trained to recognize the types of spam that you receive by learning from a set of messages that you consider spam and a set that you consider non-spam. (SpamAssassin and the spam-filtering community often refer to non-spam messages as *ham*.)
- The sending system's IP address can be compared to the sender's domain name using the Sender Policy Framework (SPF) protocol (*http://spf.pobox.com*) to determine if that system is permitted to send messages from users at that domain. This feature requires SpamAssassin 3.0.
- SpamAssassin can privilege senders who are willing to expend some extra computational power in the form of Hashcash (*http://www.hashcash.org*). Spammers cannot do these computations and still send out huge amounts of mail rapidly. This feature requires SpamAssassin 3.0.

Organization of SpamAssassin

At heart, SpamAssassin is a set of modules written in the Perl programming language, along with a Perl script that accepts a message on standard input and checks it using the modules. For higher-performance applications, SpamAssassin also includes a daemonized version of the spam-checker and a client program in C that can accept a message on standard input and check it with the daemon.

Other Antispam Approaches

SpamAssassin combines message format validation, content-filtering, and the ability to consult network-based blacklists. Filtering systems require little user intervention and introduce little delay into the process of sending and receiving email. There are other approaches to preventing spam, each of which comes with its own advantages and disadvantages (and many of which can be used in addition to, as well as in place of, SpamAssassin).

In a *challenge/response* system, the system holds all messages from unknown senders and sends them a reply message with a unique code or set of instructions (the *challenge*). The senders must reply to the challenge in some fashion that verifies their email addresses and (generally speaking) proves that they are human beings, rather than an automated bulk mail program (the *response*). After a successful response, the system allows messages from the sender to be accepted, rather than holding them.

In *greylisting* systems, the mail server initially returns a temporary SMTP (Simple Mail Transfer Protocol) failure code to messages from new senders or sending systems. If the sending system attempts to resend the message after a reasonable time period, the mail server accepts the message and subsequent messages from the sending host. Because spammers are likely either to treat the temporary failure as a permanent failure, or to attempt to deliver messages continually during the greylisting time period, their messages are not received.

In time-limited address systems, users generate unique variations of their email address to include in different web forms, email messages, newsgroup postings, etc. Addresses may be valid only for a limited time or may be valid until revoked by the user. In these systems, if a user receives spam at one of his addresses, he can usually identify the company that spammed him (or sold his address to a spammer), and he can quickly invalidate the address to prevent further spam.

In micropayment systems, senders must pay a small fee for each message they send, making large-scale spam runs costly. In some of these systems, the micropayment is refunded when the recipient determines that the message is in fact non-spam. (SpamAssassin 3.0 supports a variation of micropayments in the form of Hashcash, in which the payment is made in processing time rather than money.)

Most of SpamAssassin's behavior is controlled through a systemwide configuration file and a set of per-user configuration files. The per-user configuration can also be stored in an SQL database.

For a great deal more about Perl, check out *Learning Perl*, by Randal L. Schwartz and Tom Phoenix, or *Programming Perl*, by Larry Wall, Tom Christiansen, and Jon Orwant, both from O'Reilly.

Mailers and SpamAssassin

Although it's possible to run SpamAssassin manually on a single message, SpamAssassin becomes really useful when all incoming messages are scanned automatically. There are several ways that this can be done.

Figure 1-1 shows a typical mail transmission. The sending system connects to the recipient's mail transport agent (MTA) and transmits the message. If the message is destined for a user on the MTA's system, the MTA hands the message off to the local mail delivery agent (MDA), which is responsible for storing the message in a user's mailbox. Users may log into the system and read their mail directly from their mailboxes (as is typical on multiuser Unix systems), or, if the system runs the appropriate servers, users may download their mail using a mail client that supports the POP (Post Office Protocol) or IMAP (Internet Message Access Protocol) protocols.

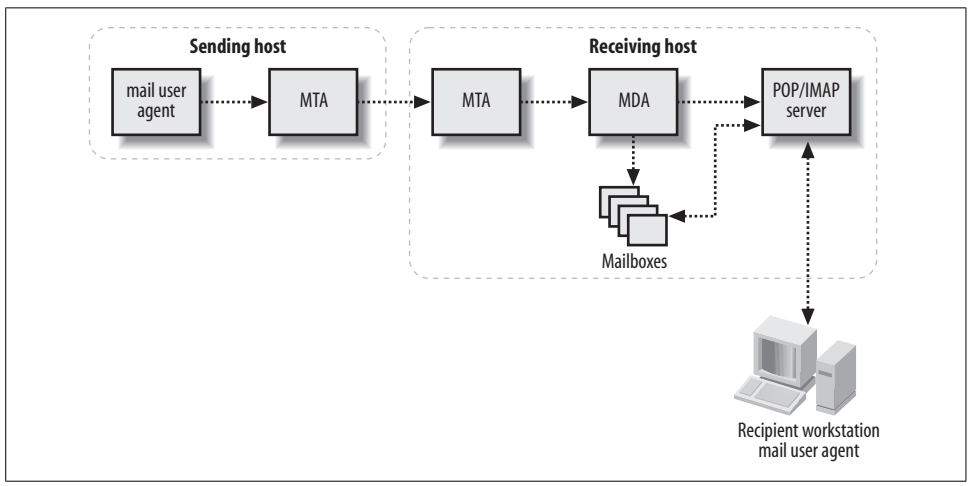


Figure 1-1. A typical mail transmission

SpamAssassin can be run in three fundamental places: at the MTA, at the MDA, and as a POP proxy. Each has advantages and disadvantages.

Scanning at the MTA

Some MTAs provide a way for incoming messages to be passed through a filter during the SMTP transaction; others can pass messages through a filter after the SMTP transaction is complete. Spam-checking is one kind of filtering that can be usefully performed at the MTA; virus-checking is another. In many cases, sophisticated filtering daemons have been developed for specific MTAs, and these daemons are capable of calling SpamAssassin to perform spam checks.

Because all email destined for users on the system must pass through the MTA, it is a natural place for centralized spam-checking. If you run a gateway MTA that delivers mail to several internal systems, you can perform spam-checking at the gateway MTA to limit the amount of spam that any internal server will receive.

In addition to tagging messages that appear to be spam, MTA-based filters can often take other actions, such as blocking a message (either refusing to complete the SMTP transaction or discarding it after the SMTP transaction has taken place) or redirecting it to quarantine area. If the MTA is already running a filtering system to do virus-checking, spam-checking can usually be performed by the same filter and share some of the overhead associated with filtering.

A disadvantage of scanning at the MTA alone is that the MTA filtering system may not be able to access per-user preferences for scanning if the filter does not have access to the recipient information, if the recipient is at another host, or if the message is destined for multiple users on the same system.

Scanning at the MDA

On many Unix systems, the mail delivery agent is procmail, which can submit messages to SpamAssassin and act on the results. This is the most typical way that SpamAssassin is installed alone, as it does not require any MTA-specific filter interfaces.

This configuration maximizes flexibility. Systemwide SpamAssassin rules can be applied to all incoming messages, and users can supplement or modify them with their own per-user SpamAssassin configuration, because, by definition, the MDA always knows the recipient to which it is delivering the message. Users who are proficient in writing procmail recipes gain complete control over the disposition of messages marked as likely spam; procmail can be instructed to discard them, file them in a separate mailbox, modify message headers, or take many other actions.

The downside of this configuration is that spam-checking is applied only after a message has been received by the system and has consumed some system resources. Another disadvantage is that spam-checking must be set up on every system that has local recipients, rather than at a single centralized MTA gateway.

Scanning with a POP Proxy

POP mail users who want the benefits of SpamAssassin on mail servers that don't provide it can use a proxy to perform spam-checking. The proxy runs on the client computer and integrates with the POP mail reader to scan messages as they are downloaded via POP.

The best known POP proxy for SpamAssassin on Windows systems is SAproxy by Stata Labs. SAproxy Pro is a commercial product, but the source code is freely

available under the same terms as SpamAssassin itself for administrators who wish to compile it and provide it to their users.

Proxies are the most decentralized approach to spam-checking and require the mail server to be liberal in accepting messages so that each user's proxy can apply their own standards. This may increase the storage load on the mail server. On the other hand, proxies completely remove the computational load from the mail server, as all spam-checking is performed by the client.

Scanning at Multiple Places

It's entirely possible to run SpamAssassin at two or even all three of the places discussed in the previous sections. An MTA-based filter could use SpamAssassin with conservative settings to refuse messages that are highly suspicious. An MDA filter on the same system could apply a more liberal (and per-user) definition of spam in order to tag messages for users who read their mail on the server itself. Finally, POP users could apply their own spam-checking by running SAproxy on their client machines.

The Politics of Scanning

If you're an ISP that provides email service, many of your users will want—perhaps even demand—spam-tagging or spam-filtering of their incoming email. Other users, however, may not want their email tagged or filtered, either because they don't get much spam, don't perceive the spam they receive to be a problem, or are concerned about the possibility of a real message being mistakenly tagged as spam.

Before you implement systemwide or sitewide spam-checking, consider carefully the needs of your users and your responsibilities toward them. At minimum, you must inform users (and would-be users) of any unconditional spam-checking you perform on their email. Better yet is to provide spam-tagging only for those users who opt to turn it on. Best of all is to enable each user to configure their own settings and threshold for how spam is recognized. This is doubly important if you not only tag messages for users but actually filter or block spam for them.

SpamAssassin is an excellent tool for distinguishing spam and non-spam email, but only if you've determined that your users want you to distinguish the two.

CHAPTER 2

SpamAssassin Basics

This chapter explains how to get and install SpamAssassin and its components, perform basic configuration, test the system, and start using it for spam-checking. It covers the basics of using SpamAssassin from the shell or from procmail, and discusses the setup of the daemonized version of the spam-checker. The configuration examples in this chapter provide only the basic functionality. The following chapters cover rule-tweaking, white- and blacklisting, and learning.

Prerequisites

SpamAssassin is written for a Unix or Unix-like environment that includes Perl Version 5, preferably 5.6.1 or later. Perl is now standard on most Unix systems, but if you don't have it, the source code for Perl can be downloaded at *http://www.cpan.org*.

SpamAssassin requires several Perl modules to be installed. If you install SpamAssassin using CPAN (the Comprehensive Perl Archive Network), as described in the next section, these modules will be automatically downloaded and installed as well. If you install SpamAssassin manually, you'll need to be sure that you also have up-to-date versions of the Perl modules *ExtUtils::MakeMaker*, *File::Spec*, *Pod::Usage*, *HTML::Parser*, *Sys::Syslog*, *DB_File*, *Digest::SHA1*, and *Net::DNS*. You may also want *Net::Ident* and *IO::Socket::SSL* if you plan to use the daemonized checker (`spamd`) and its client (`spamc`) and you will allow remote clients to access your daemon.

SpamAssassin can consult several spam checksum clearinghouses. A spam clearinghouse is a server (or a distributed network of servers) that gathers spam messages reported by thousands of users around the world and provides a mechanism for a client to check a new message to see if it matches a message in the clearinghouse. These clearinghouses are known as *checksum*-based clearinghouses because rather than transmit and store complete email messages, they work with cryptographic checksums of messages. A cryptographic checksum is a much smaller data string (typically no more than 256 bits) that is, for all practical purposes, unique to the message from which it is computed.