

## Real Web 2.0: Battling Web spam, Part 2

### Use the power of community against spam

Skill Level: Intermediate

[Uche Ogbuji \(uche@ogbuji.net\)](mailto:uche@ogbuji.net)

Partner

Zepheira

09 Dec 2008

This two-part installment provides a thorough guide to anti-spam techniques. This second article discusses content analysis, the problem with spam in linkbacks, and how to share in the anti-spam effort with a community of other Web site managers through blacklists and anti-spam services.

There is a great deal to the art and science of fighting spam on the Web. In the [previous article](#) in this column I looked at ways of using workflow to raise hurdles for spammer robots, ideally without undue inconvenience to legitimate users. In this article I discuss problems of spam that come from inter-relationships between content sites and how the very community of spam victims can cooperate in the battle against spammers.

### Community policing

The most effective way to combat Web spam is through community action. Spam is a mass attack, and it cries out for a mass response. If a community can work together to uncover behavioral and content patterns of the spammers, it can share these patterns to reduce the effectiveness of robots. The community is especially useful in combating other variations on spam to which the workflow approaches discussed in the previous article are not applicable. First, I'll take a moment to describe linkback spam.

#### Linkback spam

Weblogs and other Web articles are about sharing insights and discoveries. Sometimes, inspired by an entry, people leave comments directly on the Weblog. Sometimes they reference the entry in their own entries or articles, and a linkback is a technical formalism for notifying another site when you've linked to it. It's a network signal, a "ping" that asserts a relationship between one entry and another and helps readers find related content. But, linkbacks also open up opportunities for spammers to abuse Weblogs, sending what are called "spings" (short for "spam pings"). There are three general types of linkback, and these all have their spam-related concerns.

**Refback.** When a Web browser user clicks a link from one page to another, the request to the second site includes an HTTP header called the referrer, with the URL of the first page. This process is called a "refback". The second site might track refbacks, might post listings of them, and might even follow the refback to the origin site and extract information such as the title, meta information, link text, and other text surrounding the link. Spammers realize this as one way to embed one of their client links on a legitimate site to boost its search engine profile. They send requests to target sites with the client link in the referrer field. This is called "referrer spam". The general defense against this is blacklists of refback sites.

**TrackBack.** Refbacks depend on a user following a link, but some Webloggers wanted a more deterministic way to make a linkback notification, and one of the major Weblog software vendors, Six Apart, obliged. TrackBack is a specification that includes "discovery" (advertising where and how people can send pings) and a ping process using a simple HTTP request. Advertisement and discovery for TrackBack are in the form of Resource Description Framework (RDF) information embedded in comments on the page. An author referring to an entry would send a ping to the advertised end-point. TrackBack has become very popular among Webloggers, and spammers follow Webloggers where they go. Spammers will crawl Weblogs for TrackBack listeners and send fake notifications for their client sites. TrackBack spam is such an overwhelming problem that many Weblogs have reluctantly disabled TrackBacks, just as some have disabled comment forms on Weblog entries. The counter measures to TrackBack spam are similar to the content-analysis and blacklist counter measures against comment spam covered later on in this article.

**Pingback.** A Pingback is much like a TrackBack except that the notification messages are in the form of XML-RPC requests rather than simple HTTP requests. The discovery mechanism is a sounder implementation, using HTML links or HTTP headers rather than embedded RDF. Finally, the recipient of a Pingback follows the ping back to the original site to be sure there is actually a link from there, and this step alone weeds out a lot of spammers. A TrackBack could be implemented to perform such a check as well, but it's more a fundamental part of the pingback specification. Other than this check, anti-spam measures for Pingbacks are about the same as for TrackBacks.

## Content analysis

Whether spam comes from robots or mechanical turk attacks, it has one ultimate weakness. The primary goal of the spammers is to increase the search engine ranking for their client sites, in a process called "black hat search engine optimization (SEO)" or "Spamdexing". There's usually no point to the whole exercise unless they get their links in, which means that communities can share intelligence on spam client links with the help of statistics and reported links and link patterns. The minor exceptions to this are cases where the spammer really is trying to advertise directly to readers on that site and cases of pure malice or mischief. This is just another reason why multiple approaches and philosophies of spam fighting are so important.

## Statistical analysis

The basic idea in content analysis can be considered independently of a community approach. It involves automated examination of the content to see whether it is close enough to any prior content that has been flagged as spam, through statistical analysis of text. The most common statistical approach is called bayesian inference and is explained in detail in another IBM developerWorks article (see [Resources](#)). Whenever a person flags or confirms a posting as spam or good content (sometimes called "ham"), the statistical tables are updated. Bayesian inference has also gained prominence in e-mail spam-fighting, and in non-spam-related areas such as customer affinity engines (or, commercial recommendation engines).

## Enter the community

Content analysis is even more effective when you can perform it across multiple target sites. A spammer typically works with a long list of such sites and sends robots to spam them one after the other. If the post can be detected as spam on one site, it can prepare the other sites to identify it as such, similar to the hot lists of spam bot registration e-mail and IP addresses described above. In the case of content analysis, there have been several commercial (and semi-commercial) services to complement co-operative initiatives.

Akismet is one of the better known collaborative content analysis systems, a commercial service provided by a company associated with the WordPress Weblog platform. Users submit comments to the services and receive a flag to indicate whether it has been detected as spam. Akismet is an interesting illustration of the complexity of the issue, though. It's a closed service, which means users and observers don't really have a way to review the spam detection process. One reason for this is to keep spammers in the dark so they can't game the system as easily, but an unfortunate side effect is that others cannot help improve the service. Worse, suspicion emerges when legitimate comments are flagged as spam. There have been accusations that Akismet wrongfully targets people for blacklisting, including those who criticize the company or its founders. Other such services include Mollom, Defensio, and Project HoneyPot Spam Domains List (PHSDL).

## Vetting registrations and sender details

Many Weblogs allow anyone to post a comment, but sites such as forums often require registration prior to posting anything. The registration process is another stage in which you can try to block spammers. Robots generally reuse e-mail addresses for registrations across target sites. Even the smarter ones that auto-generate addresses leave their traces in the patterns of the generated addresses. Sites and services to help administrators catch spammer registrations include "Stop Forum Spam" and "The phpBB Anti Spam Project" (see [Resources](#)).

## Getting closer to the source

In the great war against Web spam, the fight rages across every aspect of the writable content space. So far in this article I've discussed approaches that focus on the sharp point of the spammer's attack—the target site. There are also broader patterns across almost all spammer activities, and some anti-spam techniques try to take advantage of these patterns.

### Honeypots

Most Web spammers use open HTTP proxy relays for their activities (just as many e-mail spammers use open SMTP relays). This way they mask their actual IP addresses and maybe avoid detection by blacklists and such. Some blacklists actually target all open HTTP relays, reasoning that if they can find these, so can spammers. Another approach used by the community is deploying "honeypots". One form of honeypot pretends to be a dumb open HTTP proxy relay, but when a spammer uses it, the activity is logged and used to update blacklists. In addition, honeypot operators try to identify changing behavioral patterns of spammers.

### Blocking all proxies

In addition to using HTTP proxy blacklists, you might choose to automatically block all proxied requests. HTTP proxies update certain headers for which you can check. Listing 1 is an Apache configuration recipe that accomplishes this (see [Resources](#)).

#### Listing 1. Apache config recipe for blocking proxy servers

```
# Apache config recipe for blocking proxy servers
# http://perishablepress.com/press/2008/04/20/how-to-block-proxy-servers-via-htaccess/
RewriteEngine on
RewriteCond %{HTTP:VIA} !^$ [OR]
RewriteCond %{HTTP:FORWARDED} !^$ [OR]
RewriteCond %{HTTP:USERAGENT_VIA} !^$ [OR]
RewriteCond %{HTTP:X_FORWARDED_FOR} !^$ [OR]
RewriteCond %{HTTP:PROXY_CONNECTION} !^$ [OR]
RewriteCond %{HTTP:XPROXY_CONNECTION} !^$ [OR]
RewriteCond %{HTTP:HTTP_PC_REMOTE_ADDR} !^$ [OR]
RewriteCond %{HTTP:HTTP_CLIENT_IP} !^$
RewriteRule ^(.*)$ - [F]
```

## Community commitment to cost-based spam prevention

Imagine how bad the physical junk mail problem would be if there was no cost to send through the post. In my own mail I'd say I probably get about 40% junk mail. On the other hand, I'd say that 90% or more of my e-mail is junk. One approach towards reducing electronic junk, including spam in e-mail and on the Web, is to assign a small cost to the act of sending mail, or posting information on a Web site. The idea is to have a tiny and manageable burden on legitimate users, but to make it prohibitive for spammers to make millions of attacks in order to get a few thousand spam messages or postings through. Some people have discussed using actual monetary cost for this purpose, but that's an approach fraught with problems. The necessary management of micro-payments and credits would be complex and probably expensive. Also, monetary values are dramatically different across geography and social stratum, so that what might seem an infinitesimal payment for someone in Western Europe or North America is a burden for someone in sub-saharan Africa. The Internet is considered a powerful tool for bridging the digital divide, and it would be very unfortunate to compromise this benefit.

A more practical approach to cost-based spam is to force someone sending e-mail or posting a message to expend a non-trivial amount of CPU cycles in order to do so. That way, whether they are sitting at an Internet cafe in Lagos, or on their home laptop in San Francisco, it shouldn't be a serious burden for them to post a legitimate entry, but it would be prohibitive for the spammer trying to attack a thousand Web sites with a hundred postings each. Such approaches are called "proof-of-work" schemes, and the most common of these is "hashcash".

### Hashcash

Hashcash is a neat approach to cost-based spam reduction. It requires a user to compute a significant SHA-1 hash before posting content, for example by including a challenge in the JavaScript on the posting page. This ensures that the requester must expend a certain amount of CPU cycles in order to post, but it's quite inexpensive for the target site to check. It's basically a denial-of-service counter measure. The main problem with hashcash is that many spammers use "botnets", or hacked computers that have been reprogrammed to obey the spammer's commands. This means that the bad guys don't care so much about CPU usage since it's not their CPU that's in use. For this reason, hashcash makes most sense in combination with other anti-spam techniques.

### Wrap up

If by now your head is spinning with all the tricks, terminology, and details relevant to

the fight against Web spam, you get a sense of the seriousness and difficulty of the issue. But it's an area no modern Web developer can afford to ignore because the real Web 2.0 means harnessing the power of social groups to improve information systems. This will invariably attract nasty people as well as good people, and you'll soon learn that nasty people have rich incentives to be that way, and are thus very clever in their machinations. Almost any time you roll out a clever new Web 2.0 feature, you might be opening a door for a spammer. This tour through the convoluted world of spam and anti-spam should give you a sense of the problems with community on the Web, and some of the ways that same community can come together to find solutions.

# Resources

## Learn

- Read the first article in this series, "[Battling Web spam, part 1](#)", by Uche Ogbuji.
- [Linkbacks](#) (Refbacks, Trackbacks, and Pingbacks) are mechanisms for connecting Weblogs, articles, and other postings with interlocking discussion, often exploited by spammers.
- [Spamdexing](#) is a term that covers various approaches of spammers to increase search engine prevalence, including abuse of other sites.
- Check out this [recipe for blocking proxy servers in Apache](#).
- Learn more about the most popular statistical analysis method for spam in "[Implement Bayesian inference using PHP, Part 1](#)," by Paul Meagher (developerWorks March, 2004).
- Read "[Charming Python: Beat spam using hashcash](#)," by David Mertz. Learn about the hashcash technique for minimizing spam on wikis and such in addition to e-mail.
- Expand your site development skills with articles and tutorials that specialize in Web technologies in the developerWorks [Web development zone](#), including previous installments of this column.
- Stay current with [developerWorks technical events and webcasts](#).

## Get products and technologies

- Try out some forum registration blacklist and ban list services and lists
  - [The phpBB Anti Spam Project](#)
  - ["Stop Forum Spam"](#)
  - ["Fight Comment Spam, Ban IP's "](#) (includes a downloadable blacklist)

## Discuss

- Participate in [developerWorks blogs](#) and get involved in the developerWorks community.

## About the author

Uche Ogbuji

Uche Ogbuji is [Partner](#) at [Zepheira, LLC](#), a solutions firm specializing in the next generation of Web technologies. Mr. Ogbuji is lead developer of [4Suite](#), an open

source platform for XML, RDF, and knowledge-management applications, and its successor [Akara](#). He is also lead on the [Jacqard](#) agile methodology for team Web development, and the [Versa](#) RDF query language. He is a Computer Engineer and writer born in Nigeria, living and working in Boulder, Colorado, USA. You can find more about Mr. Ogbuji at his Weblog [Copia](#).