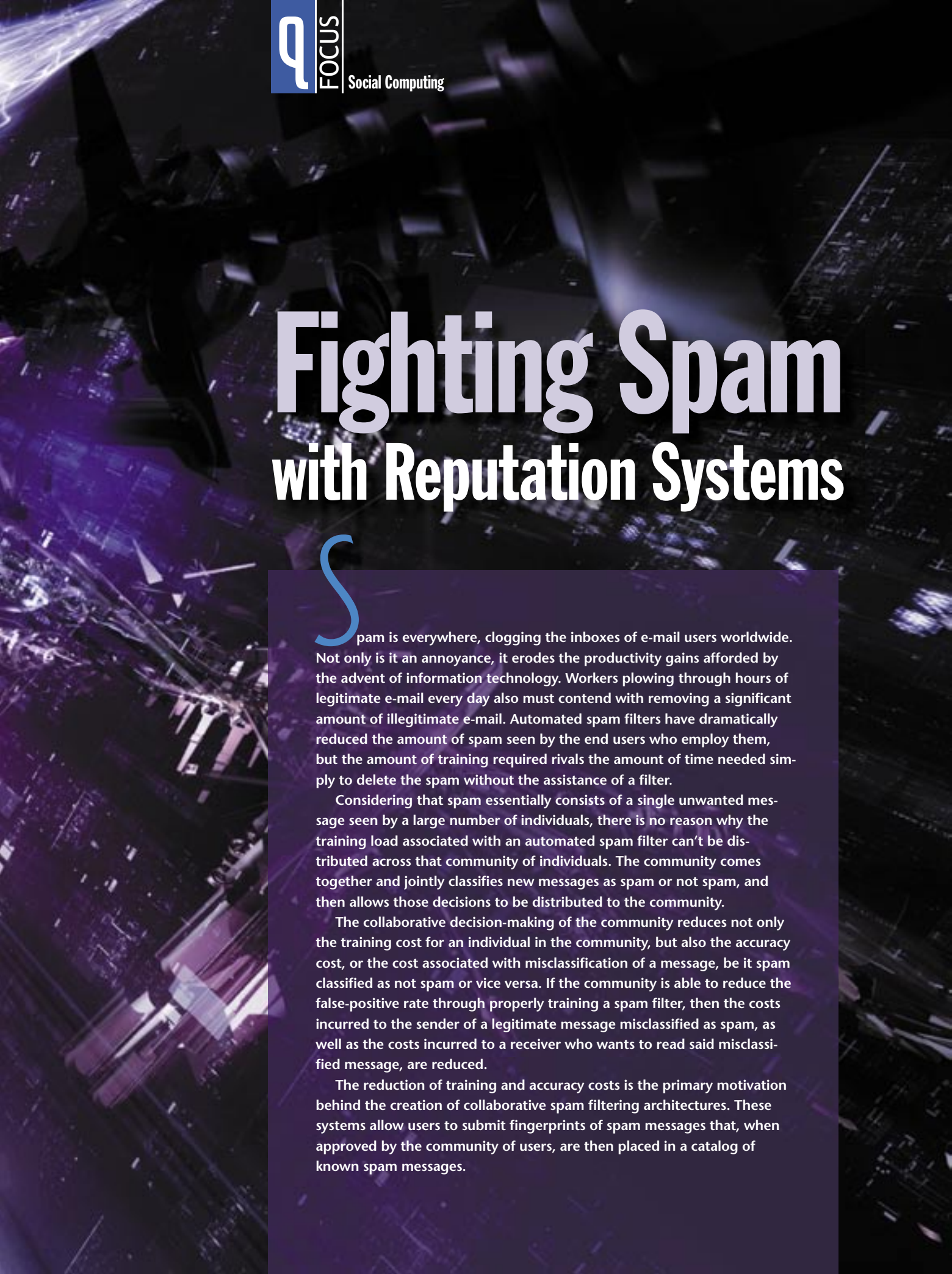
VIPUL VED PRAKASH and ADAM O'DONNELL, CLOUDMARK



Fighting Spam with Reputation Systems

Spam is everywhere, clogging the inboxes of e-mail users worldwide. Not only is it an annoyance, it erodes the productivity gains afforded by the advent of information technology. Workers plowing through hours of legitimate e-mail every day also must contend with removing a significant amount of illegitimate e-mail. Automated spam filters have dramatically reduced the amount of spam seen by the end users who employ them, but the amount of training required rivals the amount of time needed simply to delete the spam without the assistance of a filter.

Considering that spam essentially consists of a single unwanted message seen by a large number of individuals, there is no reason why the training load associated with an automated spam filter can't be distributed across that community of individuals. The community comes together and jointly classifies new messages as spam or not spam, and then allows those decisions to be distributed to the community.

The collaborative decision-making of the community reduces not only the training cost for an individual in the community, but also the accuracy cost, or the cost associated with misclassification of a message, be it spam classified as not spam or vice versa. If the community is able to reduce the false-positive rate through properly training a spam filter, then the costs incurred to the sender of a legitimate message misclassified as spam, as well as the costs incurred to a receiver who wants to read said misclassified message, are reduced.

The reduction of training and accuracy costs is the primary motivation behind the creation of collaborative spam filtering architectures. These systems allow users to submit fingerprints of spam messages that, when approved by the community of users, are then placed in a catalog of known spam messages.

Fighting Spam with Reputation Systems

THE FUNDAMENTALS OF THE CLOUDMARK NETWORK CLASSIFIER

The primary author's involvement with spam began in 1996, when it started trickling in through his 300-baud modem as a result of Usenet posts. At the time, he was researching anonymous remailers[1] and onion routing,[2] and wanted to consider anti-spam in a broader context that included messaging systems where sender was anonymous and network topology was unknown. The fundamental idea that met these design criteria was to allow the first few recipients of the spam message to inform the rest so they could filter out the spam before reading it.

The first prototype of this system, dubbed Vipul's Razor, was released as an open source project in 1998.[3] In 2001, along with a major update (Razor2), we cofounded a company called Cloudmark to work on the technology in a dedicated setting. Today, the collaborative classifier that underlies Razor2 and all of Cloudmark's products is known as the CNC (Cloudmark Network Classifier), a reputation metric analyzer that ensures the integrity of user-submitted feedback by modeling historical consensus and disagreement in the recipient community. The goal of the CNC is to ascertain the disposition of report (spam/legit) based on the first few reports, so only a few reporters have to train the classifier for a new spam attack, thus significantly reducing the overall training cost.

RELATED ANTI-SPAM METHODS

The alarming increase in magnitude of spam in recent years has fostered considerable research and development in the field of anti-spam. Many novel methods have been discovered, evaluated, and deployed. We will briefly discuss some of the more popular approaches.

A simplistic, yet popular classifier known as *address whitelisting* permits delivery of mail only from people known to the recipient. The premise is that exclusion of spammers from the "allowed sender" list keeps spam out of the mailbox. This classifier is easy to implement and works well for recipients who communicate with a well-defined and static set of correspondents. Its training performance is suboptimal in the general case, however, where recipients must continually train their own versions of the classifier to extend their correspondence networks. The accuracy cost of existing implementations is also high, since address-based classification is susceptible to address forgery, and a well-defined allow list hampers first-contact communications. Many of the shortcomings of address whitelisting can be alleviated with sender authentication schemes and reputation-based training, as we will explore later.

Statistical text classification systems such as NB (Naive Bayesian) classifiers[4] use the incoming mail's semantic similarity to the corpus of the recipient's prior communications as the basis for classification. NB classifiers tokenize mail content into words and phrases (or other linguistic units) and learn the probabilities of the appearance of various words and phrases in spam and legitimate corpora supplied by the recipient. The learned set of linguistic units and their corresponding probabilities constitute the *hypothesis*, which is used to classify incoming mail. Although the recipient must train statistical text classifiers incrementally, the training events are rare compared with the frequency of incoming mail. Most implementations come with a built-in hypothesis that serves as a starting point, which helps offset the training cost. Once trained, statistical text classifiers are quite accurate at identifying legitimate communications, and reasonably good at identifying spam. They are known to perform best in single-user environments where the training corpus accurately reflects the preferences of the user. Most real-world deployments of statistical text classification are augmented with orthogonal classifiers to derive acceptable spam-detection performance.

THE CNC ARCHITECTURE

The Cloudmark Network Classifier is a community-based filter training system. It does not rely upon any one semantic analysis scheme but upon a large set of orthogonal signature schemes that are trained by the community. The CNC consists of four important architectural components: agents, nomination servers, catalog servers, and a reputation system known as TeS (trust evaluation system). The agent software suite consists of a variety of packages used by e-mail recipients to report feedback, such as "Message is spam" and "Message is not spam," to the nomination servers. The feedback takes the form of a set of small fingerprints that are generated from the message and are on the order of 14 to 20 bytes. Fingerprinting a message rather than transmitting the entire content of the message protects the privacy of the e-mail recipient and dramatically reduces the cost associated with transmitting, storing, and processing feedback. We discuss the

general problem of spam fingerprinting in a later section.

The feedback submitted by the recipients is passed to the nomination servers, which collect all fingerprints that are nominated by the recipients as either possible new spam fingerprints or false positives. If all users were equal, and a single user could nominate a fingerprint as spam or not spam, then the signature would be redistributed to the community. The CNC, however, requires that submitted feedback be corroborated by multiple trusted members of the community. The logic that determines the community's faith in the validity of a fingerprint is embodied in TeS. It is the job of TeS alone to select which new reports are valid or invalid.

Once TeS determines that a fingerprint is "spammy," then the fingerprint is added to the catalog server. It is the last point in the process for the addition of new spam fingerprints and the first point of communication for a user who receives a new message. All messages received by a user are fingerprinted and the fingerprints are queried against the catalog server. If the queried fingerprint exists in the catalog server, the agent filters the message as spam. If the fingerprint is not in the catalog server and the recipient feels that the message is spam, then the recipient submits a fingerprint to the nomination server and the process begins again. A diagram of the process flow is presented in figure 1.

FINGERPRINTING ALGORITHMS

As stated, the CNC relies upon the use of fingerprinting algorithms for identifying messages classified as spam by the community. All of the fingerprinting algorithms employed by the CNC take the same general form: a many-to-one mapping between messages and the field of 14- to 20-byte numbers. A good fingerprinting algorithm would map many messages that are similar—namely mutations of one another—to the same fingerprint while not mapping any additional messages to the fingerprint.

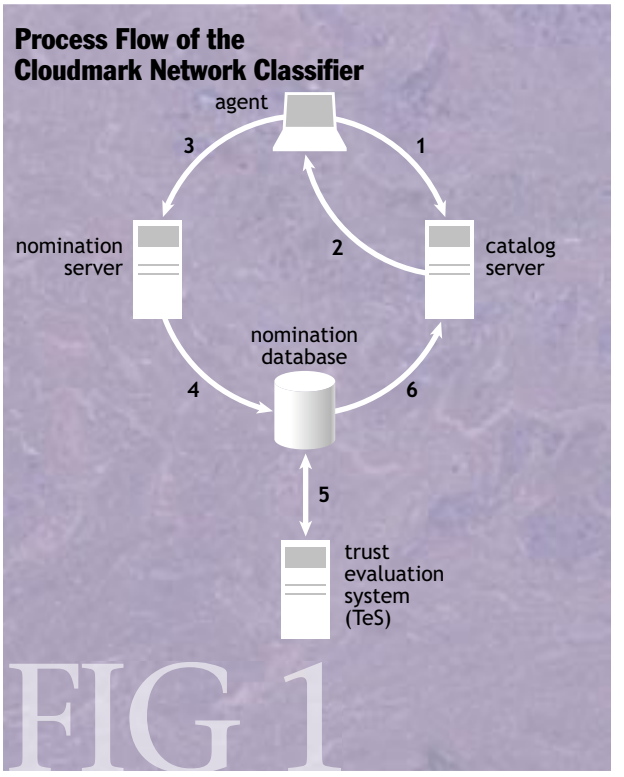
We have formalized these two properties of fingerprinting algorithms by creating two metrics: multiplicity and cross-class collision. The former describes how well a single signature classifies mutations of a single spam species. The latter measures the potential rate at which the signature could cause false positives in the system. While the creation of fingerprinting schemes is a creative process, these metrics work as a general framework to evaluate the efficacy of new signature schemes. These metrics were developed in-house mostly because there is very little literature on signature-based spam filters.

Spammers, as we know, will mutate a single message to escape naive signature schemes. A *mutation set* is defined as all the messages agreed upon to be derived from a single-source message, or that share a common message ancestor. We assume that mutation sets do not overlap.

Perfectly classifying spam into mutation classes is, for all practical purposes, an impossible task. For the evaluation of new fingerprinting algorithms, however, we can achieve a reasonable approximation by classifying corpora manually.

At Cloudmark, we define three classes of fingerprinting algorithms: a *unique* algorithm, which generates a unique hash value for every unique message, regardless of how small a mutation may exist between any two messages; a theoretical *oracle* algorithm, in which all spam messages in a single mutation class generate the same fingerprint; and a *target* algorithm, which is the fingerprinting algorithm we develop. Optimally, the target algorithm will operate exactly as the oracle and generate a single fingerprint for all messages in the same mutation class.

On the back end, two factors drive the accuracy and false-positive rate seen by the end user. We know that any new spam that comes in is one of three things: something we have seen before, a mutation of an old spam campaign that we have seen before, or a completely new campaign. If it is a campaign we have seen before, it should be in the catalog server. If it is a new campaign, the community will decide whether the bulk e-mail is spam, and the



Fighting Spam with Reputation Systems

fingerprint will eventually move to the catalog server.

Mutations of old campaigns are something that we should be able to improve upon, however. To combat mutations of old campaigns slipping past our system to the users, we must employ fingerprinting algorithms that possess high multiplicity. A cryptographic hashing algorithm, for example, would not be an appropriate fingerprint because of its sensitivity to mutations, but it would fare well when it comes to generating fingerprints that would unintentionally cover legitimate messages.

If our agents receive a large number of messages that do not have any mutations, the multiplicity numbers will look artificially low. We therefore use an additional metric, known as *unbiased multiplicity,* for the evaluation of fingerprinting algorithms during the design stage. This metric quantifies how close an experimental fingerprinting algorithm gets to generating only a single metric per mutation class.

A fingerprint with high multiplicity is capable of covering multiple mutations of the same spam from a single campaign. From the standpoint of the community, a high-multiplicity fingerprint means that a single spam campaign consisting of multiple mutations will be eliminated far earlier than if multiple fingerprints are required.

It is possible to generate a fingerprint with extremely high multiplicity, wherein whole volumes of messages are covered by the same fingerprint. The danger, however, is that a high-multiplicity fingerprint would also cover messages that are not spam or cause collisions with messages contained in the legitimate class (i.e., false positives). Since TeS independently protects against false positives by contesting fingerprints that cover both spam and legitimate messages, a high cross-collision rate renders colliding fingerprints ineffective in stopping spam.

In summary, a perfect fingerprinting algorithm will generate zero cross-class collisions (no false positives) and will have an unbiased multiplicity of one (all variations of a spam message identified). We employ six different fingerprinting algorithms, each of which has a method of operation that is mutually orthogonal. The generation of additional fingerprinting algorithms is left as an exercise to the reader.

ACCURACY IS KING

From the standpoint of a user, the most important metrics are the accuracy and the false-positive rates that the system generates. We define accuracy as the percentage of spam messages that are sent to a user's mailbox that are correctly classified as spam without user intervention. The false-positive rate is the number of nonspam messages whose fingerprints generated by f_e (our fingerprinting algorithm) have been promoted to the catalog server.

$$\text{accuracy} = \frac{|S_{\text{catalog}} \cap S|}{|S|}$$

$$\text{false positive} = \frac{|f_e(S') \cap F|}{|f_e(S')|}$$

THE PROS AND CONS OF SENDER REPUTATION

Several initiatives are currently being pursued that attempt to compute the reputations of senders rather than of e-mail. Both SPF (Sender Policy Framework)[5] and DKIM (DomainKeys Identified Mail)[6] try to identify senders by the sets of e-mail servers they use to send out mail. SPF is the more widely deployed of the two, but DKIM is gaining ground. The SPF scheme allows senders to publish lists of servers they use for sending mail through a DNS record. For example, examplesender.com can publish that it sends mail from mx1.examplesender.com or from mx2.examplesender.com. Then, before accepting mail, the recipient mail server can ensure that a sender claiming to be examplesender.com is actually coming from one of the mail servers in examplesender.com's DNS records.

DKIM, meanwhile, signs all outgoing messages with an asymmetric key whose public counterpart is published through the sender's DNS. SPF and DKIM essentially make it very hard to forge the identity of the sender, making spam filtering based upon sender information more feasible.

Once the sender's identity is established with SPF or DKIM, reputation systems can be used to measure "the goodness of senders" over time. Quite a few sender reputation projects are under way[7] that aim to track and compute sender reputation and use it to filter mail from bad senders. Sender reputation would also enable more robust versions of the address whitelisting classifier described earlier.

How accurate would such classifiers be? While we believe sender authentication is a useful and healthy augmentation to e-mail, our perspective on the usefulness of sender authentication in the context of anti-spam differs from the prevalent optimism of the industry. The problem with sender authentication schemes is that they

do not identify individual senders—they only associate identities with a collection of senders behind a host. To be more exact, sender authentication schemes identify the software and network infrastructure used by a sender to send out e-mail. There are two problems with this: a sender's reputation is affected by the behavior of all senders with whom the sender shares network resources; and the sender's reputation is affected by malicious code that hides on the sender network to send out spam or malicious code via e-mail. The first is a problem of granularity, and the second is a problem of impersonation.

We contend that host-level sender authentication and reputation will be good at identifying immaculately good senders (or sender collections), such as small organizations and phishing-afflicted institutions that are able to enforce a conscientious sender policy and safeguard their networks from zombies. Sender authentication will also be effective at identifying the absolutely bad senders—networks set up with the sole purpose of sending spam. Networks with large numbers of users behind them and those in which security will be breached by spam-spewing zombies, however, will end up with sullied reputations.

If we were to design a fingerprinting algorithm that simply used the authenticated sender host as the fingerprint, the algorithm would have a high multiplicity and a high cross-collision rate. As previously described, a high cross-collision rate results in the classifier's inability to use the contested fingerprints (sender hosts, in this case) to make filtration decisions. Thus, a sender host-based classifier would have to rely on out-of-band methods to classify a large amount of e-mail.

DKIM does allow for weak authentication of individual senders. The reliability of individual sender authentication is a function of a domain's ability to authenticate senders within the domain. Although internal authentication of senders through methods such as SMTP-AUTH is not widely deployed today, it's a very tractable solution.

Pushing authentication and, eventually, reputation to individual senders will alleviate the problem of granularity and will make for better classifiers. It might also be possible to solve the problem of impersonation by modeling sender patterns that differentiate zombie activity from that of the legitimate user.

The collaborative filtering system we describe does not need to establish trust for a global pool of mail senders. The system we describe needs to establish a weaker form of trust, which states that e-mail recipients will correctly or incorrectly classify spam regardless of who they are among a pool of users whose number is relatively small compared with the number of e-mail users worldwide.

CONCLUSION

We described the architecture and operation of the CNC and illustrated the emergent properties of the reputation system underlying the classifier. We also presented a framework for evaluating the efficacy of spam fingerprinting algorithms. Finally, we contrasted the CNC approach with other popular methods for classifying spam. The descriptions have been simplified to highlight the central themes. We hope that we have managed to convey the importance of reputation-based methods in the fight against spam. Q

REFERENCES

1. Chaum, D. 1981. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM* (February).
2. Dingledine, R., Mathewson, N., and Syverson, P. 2004. Tor: The second-generation onion router. *Proceedings of the 13th Usenix Security Symposium*.
3. Vipul's Razor; http://razor.sf.net/.
4. A plan for spam. 2002; http://www.paulgraham.com/spam.html.
5. Sender Policy Framework; http://spf.pobox.com/.
6. DomainKeys Identified Mail; http://mipassoc.org/dkim/.
7. Haskins, R. The rise of reputations in the fight against spam; http://linuxworld.sys-con.com/read/48128.htm.

LOVE IT, HATE IT? LET US KNOW

feedback@acmqueue.com or www.acmqueue.com/forums

VIPUL VED PRAKASH is the founder and chief scientist of Cloudmark, an anti-spam technology company. He is best known for creating Vipul's Razor and the Cloudmark Network Classifier. He is a prolific open source developer and has written numerous extensions to the Perl programming language for networking, cryptography, and object technology. He is a frequent speaker at industry conferences and academic events on the issues of computing and spam. In 2003, MIT's *Technology Review* named him as one of the Top 100 Young Innovators in the World.

ADAM O'DONNELL is a senior research scientist at Cloudmark. He recently completed his Ph.D. as an NSF Graduate Research Fellow in Drexel University's department of electrical and computer engineering. He was the technical editor and contributor to *Building Open Source Network Security Tools* by Mike Schiffman (2002, Wiley), a contributing author on *Hacker's Challenge* (2001, McGraw-Hill), and co-author of *Hacker's Challenge 2* (2002, McGraw-Hill).