Deployment Experience: Rolling Out a New Antispam Solution in a Large Corporation

Barry Leiba
IBM Research
Hawthorne, NY
leiba@watson.ibm.com

Jason Crawford
IBM Research
Hawthorne, NY
ccjason@us.ibm.com

Abstract

Our research group has developed new, state-of-the-art antispam software, described in other papers. We are in the process of deploying that software in a large production corporate infrastructure, as a replacement for the prior antispam solution. This paper describes how we went about the deployment, and our experiences therewith, with an eye toward pointing out the icebergs and the lifeboats, to help make the process go smoothly for others. We also report what we learned, in the process, about user needs and expectations with respect to antispam solutions.

1 Introduction

As antispam software evolves, large organizations may find themselves replacing existing antispam solutions with new ones, for reasons technical, contractual, or both. While we would like to think that this should be easy, and vendors will certainly present their software (and hardware) as drop-in replacements, there are, in fact, many things to be considered and many things that can go wrong along the way.

IBM's[1] Research Division has developed a state-of-the-art antispam solution, called SpamGuru (Segal et al., 2004). SpamGuru represents a significant improvement over the prior system in use at IBM, using novel message classifiers (Rigoutsos and Huynh, 2004; Leiba et al., 2005) and an experimental aggregation system (Segal, 2005) to use the combination to best advantage. With the cooperation of the Research Division IS department and the Office of the CIO, we have begun deploying our system in the corporate infrastructure.

[1] "IBM" is a registered trademark of the IBM Corporation.

As with any such deployment in a large corporation, there are many procedures to follow, assurances to be made, plans to be documented, checks to be checked, and balances to be balanced. In addition, there are several aspects unique to the deployment of antispam software. This paper will describe those aspects, the phased process we used to ensure that it all went smoothly, the difficulties we encountered, and how we resolved them.[2] We will also tell what we learned about what users want from, and expect of, antispam solutions, and how those needs and expectations affect deployment plans.

2 Getting Started

When we first implemented antispam filtering at IBM, we set up a task force to analyze the situation, review the alternatives, and make recommendations to the Office of the CIO. The CIO office acted very quickly and, with the output of the task force barely in hand, we deployed an antispam solution, first in the US and then throughout the corporation. At first, spam was handled by labeling it, by prepending a string to the subject, and allowing user agents (or the users themselves) to handle the suspected spam based on the label.

While that first system evolved, and went from labeling to deleting, a team in IBM's Research Division formed, composed of people who had been working on text classification, machine learning, email, and antispam filtering on other projects. It was from there that we got started on a project to replace the initial system, the "prior" antispam solution, with one developed by our Research team.

Of course, the first part of getting started, for us, was to develop the software. SpamGuru is described in detail

[2] Corporate policy prevents us from revealing details of the prior antispam system (hereinafter referred to as the "prior" or "old" system), and from comparing its performance quantitatively to that of SpamGuru. Fortunately, such details are not germane to the central aim of this paper, which is to describe the experience of replacing an existing anti-spam solution, not to make specific comparisons.

in another paper (Segal et al., 2004), and, while software development is not a part of most deployments, the selection of an antispam solution is a general issue, and a complex one in a world with many options. There is no "best" antispam technique, and the best approaches combine many techniques to make them flexible and robust (Leiba and Borenstein, 2004). This lack of a single best technique leaves a lot to choose from, in both software and hardware (many antispam "appliances" are now available, sold as "plug-and-play" devices to be inserted into the organization's mail stream), as well as hosted services.

For an organization that's deciding to *change* how it handles spam, the primary technical consideration is how effective the new system is, as compared with the old one. Note that this is a very different decision point from what might be used in choosing to implement an antispam system for the first time, as users start with different expectations; indeed, since they have likely been receiving a great deal of spam, nearly anything will seem better. When replacing an existing system, one faces users whose expectations have been set by the prior system, and even small changes will be noticed.

We decided that the best way to compare the systems was one that also allowed us to do comparisons of a number of aspects, including reliability, manageability, and user satisfaction: we would run the two systems in parallel, using a phased rollout.

3 The Phased Approach

We had the good luck of having a Research domain, *watson.ibm.com* (the "Watson domain"), with which we could do some experimenting before implementing the new system on the main corporate domains (*us.ibm.com* and the other "country-code" domains). The Watson domain provided us a production environment that doubled as a testing environment, because disruptions to that domain are not as serious as are disruptions to the country-code domains. Although having a domain like this was not necessary, and in some environments it's not possible, being able to test on such a domain gave us an extra level of comfort that we valued.

As we tested our deployment in the Watson domain, we chose a set of deployment phases that turned out to work well, and which we used again when we later supported the country-code domains:

1. Insert the new SMTP server into the stream, with all processing turned off.
2. Turn on classification in the new software, but relay or discard mail based on the decision of the prior antispam software.
3. Switch the operations in phase 2: continue classifying mail in both systems, but handle the mail according to the *new* software.

Phase 1 made sure that the routing, MX records, and the like were set up correctly, and also ensures that the servers involved can handle the email volume. It was good to run this phase briefly, to eliminate these as potential causes for the problems we might encounter in the next phase. How the different systems are arranged will depend upon the features available in each software package. In our case, because we wrote the SpamGuru system, we had infinite flexibility in how it worked, and could set it up to take the output from the old software, keep logs of comparisons, and then handle the mail according to the old classification or the new (or both), based on configuration options.

In phase 2, we again had to evaluate how we handled the email volume, since we began doing the extra work of email classification. At this phase, we compared the throughput of the old and new systems, and their effectiveness at filtering the spam. While we only ran phase 1 for a week or so, phase 2 continued for a longer period, as we monitored and tuned the new system.

Phase 3 essentially put us into full production, but with a "safety net": if something should go wrong, we still had the old software running and could fall back to it. We also could continue to monitor the effectiveness of both, and compare them, and the results of that helped us know what to expect over time when we moved into the country-code domains.

4 User Experience and Expectations

The SpamGuru deployment in the Watson domain gave us a good chance to do an informal survey of users on their experience with the system, and to observe their behavior and expectations. What we got, through feedback from the pilot users and input from the corporate task-force, was a good sense for some of the needs users have, some of the needs they *think* they have, and what their tolerance is for ups and downs in an antispam system.

Very broadly, these are what we came across (some of these, of course, will surprise no one):

1. Users do not want *any* spam.
2. Users do not want *any* legitimate mail deleted as spam (no false positives).
3. Users vary in how they set the priority between items 1 and 2 – that is, what their tolerance is for false positives.
4. Users are certain that you *are* deleting legitimate mail.
5. Users *want* to send you the spam that gets past the filters.
6. If there are outages or problems, users *will* complain.

4.1 Spam and False Positives

We all know that there is a conflict between items 1 and 2, above, and we strive to balance it. The trouble is that different groups and different individuals we surveyed expressed strong needs for different balances. Some scientists we spoke to, who get a large amount of Internet mail related to their work, have no tolerance for false positives and would rather get more spam than miss any legitimate messages. On the other hand, an administrative group made it clear that their priority was to get *no* offensive mail, and they were quite willing to miss some legitimate mail to achieve that. Corporate policy, too, would affect the balance between eliminating spam and avoiding false positives. And in divisions outside of Research, we knew we would have to worry much more about mail coming from customers, for which false positives would be serious.

In addition to the *needs*, though, is the *perception*. Many users – perhaps most – believe that the antispam filters are catching legitimate mail as spam, and those with the strongest need for no false positives believe that the strongest. One way to deal with that fear is to not *delete* spam, but to deliver it to the user in another manner, outside the inbox.

A result of corporate policy to delete is that users who worry about false positives have no way to check for them or fix them. What's worse for SpamGuru is that it is a learning system that relies on input from users, and it is especially valuable to be told about false positives. To get that feedback, we must give them a way to look at the mail we'd like to be deleting.

SpamGuru has a number of ways to do that. Instead of *completely* deleting a message, we can, for example:

1. tag the message with a header record (or text prepended to the subject) and deliver it, or
2. deliver the message using sub-addressing (changing smith@example.com to, for instance, smith+junk@example.com), or
3. archive the message in a web-accessible repository, so that users may review it as they please.

The first mechanism allows the user to use user-agent filtering to move the message to a non-inbox folder if she so chooses. But both of the first two mechanisms go against corporate policy, and so it's the third that we implemented in our experiment.

4.2 User Voting

We found that users' minds were eased by knowing that they had the option of checking for false positives, and that they seldom actually checked. This is both good and bad: It's nice to be trusted, yet, because SpamGuru is a cooperative system driven by feedback, if users don't check the archive and vote, we don't learn. Now, that's not entirely true, because of item 5 in the list above, "Users *want* to send you the spam that gets past the filters," so we will get some votes in any case.

SpamGuru provides three ways for users to give feedback to classification engines:

1. The archive – the archive has a GUI for voting.
2. An API – we provide an API that MUAs and other applications can call to submit votes.
3. Email – the message can be forwarded to one of two email addresses, to vote "good" or "spam".

For users who do not use the archive, we provided an easily installed pair of Lotus Notes "buttons", which allow a user to select one or more messages and simply press the appropriate button to send us votes on the selected messages. Users who use other MUAs can forward the messages, and we've told them how.

We receive many votes, but few votes are for non-spam. We have tried to encourage users to vote as much of their legitimate mail as they're willing to take the time to, so that our classifiers may keep learning "good" patterns in addition to spam patterns. Indeed, our users probably are doing so, but "as much as they're willing" is pretty close to "none". We are, therefore, relying on the handful of "early adopter" types, who are willing to do the extra work to help the effort, to give us the relatively few non-spam votes that we get. This limits the effectiveness of our learning.

For these early adopters, we set up a volunteer "pilot". Users who sign up for that agree to receive *more* spam during the pilot period, in order to help tune the system.

1. We optionally change the spam threshold for them. Raising the threshold will send more spam their way for voting. Lowering it will cause more apparent "false positives", which they will vote.
2. For all mail where the old and new antispam systems *disagree*, we tag the mail with "[PleaseVote]" on the subject, and we deliver it.
3. Based on the user's individual message volume, we tag a random sampling of spam and a random sampling of non-spam with "[PleaseVote]". This forces votes for some legitimate mail.

Pilot participants are asked to vote thoroughly and carefully on all mail that's tagged with "[PleaseVote]". We have about 150 pilot users, and their votes are helping keep the SpamGuru system performing well. As we move forward, we continue to solicit more participants to volunteer to join this voting corps.

5 Reliability and Availability

To the final point, "If there are outages or problems, users *will* complain," we can add that "Research users will complain *more*." The good side of that is that they will complain *quickly*, as well, and we have actually relied on that to let us know about problems right away.

As it turns out, reliability problems all occurred early, and were soon resolved, and the system has now been stable for a considerable time. When we first set up the system, we used an SMTP server built on Research software called the Internet Messaging Framework (von Känel et al, 1998). We used the IMF server in the Watson domain, but corporate policy required a switch to Sendmail[3] for deployment in the country-code domains, so we re-implemented the links to the SMTP server using the more limited Sendmail *milter* interface.

The biggest drawback of using the milter lay in our personalized handling of messages. SpamGuru derives personalized as well as global knowledge from votes and can provide personalized classifications. If a message is sent to multiple recipients, SpamGuru can evaluate the message using *personalization*, and might give the same message different spam scores for the different users. The IMF server allowed us to split the message, in that case, sending different copies, with different scores, to the different users. The milter interface does not provide that function, and so we are, for now, doing without it. We currently use the lowest personalized spam score as the overall score for the message.

The importance of phase 2 became clear when we found that SpamGuru had some initial throughput problems, causing the mail queues to back up severely. The Watson domain processes some 250,000 messages per day, while the country-code domain in North America handles some 14,000,000. We were only operating in the Watson domain at the time, and we already could not handle the load. Since we were, in phase 2, still obeying the prior antispam system's judgment, users saw no ill effects.

Another feature of our phase 2 performance testing was our decision to run the old and new solutions on nearly identical hardware. This choice allowed us to use standard CPU, disk, network, and memory monitoring tools to compare the performance of both systems. That, in turn, allowed us to anticipate which system would be better able deal with expected short and long term increases in load, so we could better understand long term hardware and maintenance costs of the new solution. It also allowed us to shake out the code and fix some software problems.

6 Effectiveness

As we said earlier, once the new antispam system is stable, its effectiveness at filtering spam is probably the most important thing to evaluate before making the switch. This was another important reason for phase 2 in the plan: since SpamGuru is a learning-based system, though it was bootstrapped with an initial feature database it needed time to learn from the current stream of mail

After an initial learning period, SpamGuru consistently identified and removed a few percent more spam than the prior system. Had we deployed the new system outright, during the learning period in which it was much less effective, we would have had a user rebellion. Contrast that with the initial antispam deployment at IBM, where simply labeling some 2/3 of the spam delighted nearly everyone. User expectations increase quickly, and once expectations are built, they're hard to tear down, and the organization is committed to maintaining a comparable level of service thenceforth.

7 Miscellaneous Notes

There are a good number of other issues that one encounters when deploying any system within a large corporation. We had to interact with several different groups that supported different aspects of the production environment – and, actually, this part went remarkably well, as we connected with people who were eager to see better antispam software working.

Nevertheless, we still faced the issues of obtaining hardware, getting software installed, and setting up access controls. We had to worry about budgets and about who could authorize which changes, and we had many teleconferences to make sure everything was sorted out. Most notably, we had many delays caused by periodic "change freezes", during which progress on the deployment halted while we waited for end-of-quarter accounting, and the like.

Lest that all sound bad: it was not. The change freezes were frustrating, but the rest went quite smoothly thanks to the people who wanted to see it work, and it was *good* to have others check over what we had done and ask the questions that, often, we were too close to the system to ask.

[3] "Sendmail" is a registered trademark of Sendmail, Inc.

8 Summary

In our case, a change was made by choice, from one antispam solution to another. Other organizations may also choose to – or have to – change, perhaps for reasons of effectiveness, perhaps for reasons of cost, or perhaps for reasons of satisfaction with a vendor's product support. For whatever reason the change is made, it must be made with care, noting particularly the following:

1. It is harder to replace an antispam solution than to implement one in the first place. User expectations are a significant issue.
2. Backup and back-out plans are essential.
3. Phased deployment helps a great deal, allowing reliability, throughput, and effectiveness to be measured (and compared).
4. Knowing your user base, and the differences in needs among different subgroups, is important in choosing a new system, in configuring the system, and in maintaining satisfaction during a new deployment.
5. Feedback in the system is useful for training some systems, and, for all systems, for judging the effectiveness and for maximizing user satisfaction. The presence of a feedback mechanism may be a deciding factor in the choice of a system. Alternatively, providing one in-house should be strongly considered.
6. Feedback also satisfies users, in that it gives them a formal way to "complain", and to know that their complaints will be acted on. This effect also reduces the support cost that previously went to dealing with user complaints that were directed inappropriately (because there was no appropriate place to direct them).
7. With the variety of antispam mechanisms available today, choosing something that can implement several of them together is best (see Leiba and Borenstein, 2004).

Acknowledgements

The authors thank the other members of the IBM Research antispam team, the Research Division IS department, the Office the CIO, and the IBM Global Services team that worked on the deployment described herein, with special thanks to Brian Fonseca, Mike Halliday, Annette Higgins, Matt Karius, and Richard Segal.

References

von Känel, J., Givler, J.S., Leiba, B., and Segmuller, W. "Internet Messaging Frameworks", IBM Systems Journal, vol 37 #1, IBM Corporation, 1998.

Leiba, B. and Borenstein, N. "A Multifaceted Approach to Spam Reduction", Conference on Email and AntiSpam 2004, July, 2004.

Segal, R., Crawford, J., Kephart, J., and Leiba, B. "SpamGuru: An Enterprise Anti-Spam Filtering System", Conference on Email and AntiSpam 2004, July, 2004.

Rigoutsos, I. and Huynh, T. "Chung-Kwei: a Pattern-discovery-based System for the Automatic Identification of Unsolicited E-mail Messages (SPAM)", Conference on Email and Anti-Spam 2004, July, 2004.

Segal, R. "Combining Multiple Classifiers", Virus Bulletin, February 2005.

Leiba, B., Ossher, J., Rajan, V.T., Segal, R., and Wegman, M. "SMTP Path Analysis", Conference on Email and AntiSpam 2005, July, 2005.