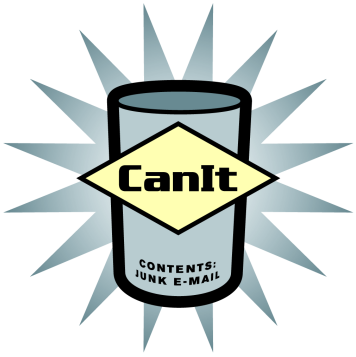

WHITE PAPER

RPTN

A Mechanism for Sharing Bayes Votes

May 2005

Document Control: $Id: rptn.tex,v 1.2 2005/05/17 17:08:26 dfs Exp $

Abstract

This white paper describes the Roaring Penguin Training Network (RPTN), a mechanism for sharing Bayes votes among multiple CanIt installations.

1 Introduction

A recent paper at the 2004 USENIX Large Installation System Administration conference by Jeremy Blosser and David Josephsen presented evidence that Bayesian filtering could be quite effective using a shared Bayes database, even among hundreds or thousands of different users. This lead to the Roaring Penguin Training Network (RPTN), a mechanism for sharing Bayes votes among different CanIt customers.

The theory of RPTN is as follows:

1. Each CanIt installation whose administrator volunteers to submit data to RPTN keeps track of messages that are hand-trained.
2. Periodically, the CanIt installation contacts the central RPTN server and authenticates itself for the purpose of RPTN data upload.
3. The central RPTN server grants permission to the CanIt installation to upload a certain number of spam and non-spam signatures.
4. The CanIt installation randomly samples the hand-trained mail and sends up to the requested number of signatures to the RPTN server. This is called an *RPTN Report*.
5. Periodically, the RPTN server runs quality-assurance checks over the submitted signature sets, and aggregates those that pass the QA stage.
6. The aggregated data is made available to CanIt installations. Periodically, a CanIt installation can download the aggregated Bayes statistics. Administrators (in the case of CanIt-PRO, stream owners) can choose to inherit from the RPTN Bayes data. In this case, the RPTN Bayes statistics are added to the individual CanIt installation (or stream) statistics, and the combined statistics are used for Bayesian analysis.

2 Submitting to RPTN

By default, CanIt will *not* submit information to RPTN. There may be some privacy concerns, because tokens from e-mail messages are submitted to Roaring Penguin's server, and they may appear in the aggregated statistics.

For example, consider the following e-mail message:

```
Subject: Meeting tomorrow

Hi, everyone.

We'll meet tomorrow in the boardroom to discuss
RPTN.  Please bring the notes from the design
meeting last week; I'll supply coffee and muffins.

Regards,

David.
```

The message would result in the following signature being submitted to RPTN:

```
David:1;I'll:1;I'll+supply:1;Please:1;Please+bring:1;RPTN:1;
RPTN+Please:1;Regards:1;Regards+David:1;We'll:1;We'll+meet:1;
boardroom:1;boardroom+discuss:1;bring:1;bring+notes:1;coffee:1;
coffee+muffins:1;design:1;design+meeting:1;discuss:1;discuss+RPTN:1;
everyone:1;everyone+We'll:1;last:1;last+week:1;meet:1;meet+tomorrow:1;
meeting:1;meeting+last:1;muffins:1;muffins+Regards:1;notes:1;
notes+design:1;s*Meeting:1;s*Meeting+tomorrow:1;s*tomorrow:1;supply:1;
supply+coffee:1;tomorrow:1;tomorrow+boardroom:1;week:1;week+I'll:1
```

As you can see, the words and word-pairs from the original message are visible. While it is a little difficult to reconstruct the original message, the information imparted by a signature may be a privacy concern. If you feel that submitting signatures to RPTN may be a privacy risk, *please do not enable RPTN submission.*

Naturally, the more people submitting to RPTN, the better, and if you do not submit data, then your "votes" don't count. It's up to each CanIt administrator to weigh the privacy risks before deciding to enable RPTN submissions. Roaring Penguin will guard the signatures very carefully, but our personnel will have access to the cleartext signatures. Note that if you have Gnu Privacy Guard installed, then RPTN reports are encrypted using our RPTN public key before being e-mailed. This makes it practically impossible for an eavesdropper to obtain the signatures.

The privacy risk from the aggregated statistics is much lower, because the aggregated statistics consist of tokens from thousands of signatures from many different submitters, and there is no indication of which installation submitted which tokens.

3 Authentication

Before a CanIt installation is allowed to submit data to RPTN, it must authenticate itself. The installation connects via HTTPS to the RPTN server and logs in using your Roaring Penguin download username and password. If authentication is successful, the RPTN server returns four values to the CanIt installation:

1. N_s, the maximum number of "spam" votes the server will accept in the next RPTN submission.
2. N_n, the maximum number of "non-spam" votes the server will accept in the next RPTN submission.
3. C, a 160-bit random cookie that identifies the RPTN submission.
4. S, a 160-bit secret used to authenticate the RPTN submission.

4 Submission

RPTN reports are submitted back to RPTN via e-mail. E-mail was chosen for the following reasons:

- CanIt installations must already be able to send out e-mail, so submitting via e-mail does not require any firewall changes.
- E-mail is robust in the face of server failures; if the RPTN server is temporarily unavailable, the RPTN reports will queue until it comes back up.
- GPG is a free, robust and well-known mechanism for encrypting e-mail, and shipping our public key with CanIt makes it simple to securely encrypt RPTN reports.

A CanIt installation constructs an RPTN report as follows:

1. It places the cookie C in the Subject: header

2. It gathers all the signatures for submission and compresses them with `bzip2` to yield data *D*.

3. If GNU Privacy Guard is installed, it encrypts *D* with our RPTN public key to yield ciphertext *E*.

4. It calculates the SHA1 hash of *S* (the 160-bit secret) concatenated with *E* (or *D* if GPG is not installed.) This result is placed in an X-Message-Authenticator: header. This authenticator is designed to stop people from tampering with the SMTP report en route to the RPTN server.

5. The data *E* (or *D* if GPG is not installed) is Base-64 encoded and placed in the message body.

The RPTN report is then e-mailed to a special address that accepts RPTN report submissions.

Before accepting an RPTN report, the RPTN server takes the following actions:

1. It looks up the cookie *C* from the Subject: header. If no such cookie is found, or if a previous submission using the same cookie has been submitted, the report is rejected.

2. It retrieves the secret *S* associated with *C* and validates the message hash. If the message cannot be validated, it is rejected.

3. If the signatures have been encrypted, they are decrypted using our RPTN private key.

4. The server notes that cookie *C* can no longer be reused, and it places the report in an incoming spool area. In addition, it stores the login name of the user who submitted the report (obtained by looking up *C* in the authentication database.)

5 Quality Assurance

Because RPTN distributes Bayes data to many customers, we need to ensure that it's difficult for bad data to contaminate the aggregated statistics. Quality assurance is done using two methods: *static* analysis and *dynamic* analysis.

5.1 Static Analysis

In the static analysis phase, we extract the list of signatures submitted by each user and use them to run Bayesian analysis against a corpus of spam and non-spam messages. Signature sets that have an unacceptably high error rate are discarded.

5.2 Dynamic Analysis

In dynamic analysis, we compare each signature set against *all* the other signature sets. If a signature from customer i is marked as `spam`, but the data from the other $N-1$ customers would rate it as non-spam, we count a false-positive. Similarly, if i is marked as `non-spam`, but all the other signatures say that it is spam, we count a false-negative. Signature sets with an unacceptable high false-positive or false-negative rate are discarded; we weight false-positives higher than false-negatives when determing whether or not to drop a signature set.

6 Aggregation and Download

Those signature sets that survive the quality assurance phase are aggregated and dumped to a CSV file. The file is compressed with `bzip2`, signed with `gpg` and made available for CanIt installations to download.

When a CanIt installation notices a new RPTN data set available (indicated with a special DNS lookup), it downloads the aggregated statistics, verifies the GPG signature, uncompresses the file, and loads the statistics into the Bayes database. The statistics are loaded into a stream called `@@RPTN`. Streams can choose to inherit Bayes data from this stream if they wish to use RPTN statistics.