Network Working Group
Request for Comments: 5235
Obsoletes: 3685
Category: Standards Track

C. Daboo

January 2008

Sieve Email Filtering: Spamtest and Virustest Extensions

Status of This Memo

Abstract

The Sieve email filtering language "spamtest", "spamtestplus", and "virustest" extensions permit users to use simple, portable commands for spam and virus tests on email messages. Each extension provides a new test using matches against numeric "scores". It is the responsibility of the underlying Sieve implementation to do the actual checks that result in proper input to the tests.

Table of Contents

1. Introduction and Overview

Sieve scripts are frequently being used to do spam and virus filtering either based on implicit script tests (e.g., tests for "black-listed" senders directly encoded in the Sieve script), or via testing messages modified by some external spam or virus checker that handled the message prior to Sieve. The use of third-party spam and virus checker tools poses a problem since each tool has its own way of indicating the result of its checks. These usually take the form of a header added to the message, the content of which indicates the status using some syntax defined by the particular tool. Each user has to then create their own Sieve scripts to match the contents of these headers to do filtering. This requires the script to stay in synchronization with the third-party tool as it gets updated or perhaps replaced with another. Thus, scripts become tied to specific environments and lose portability.

The purpose of this document is to introduce two Sieve tests that can be used to implement "generic" tests for spam and viruses in messages processed via Sieve scripts. The spam and virus checks themselves are handled by the underlying Sieve implementation in whatever manner is appropriate, so that the Sieve spam and virus test commands can be used in a portable way.

In order to do numeric comparisons against the returned strings, server implementations MUST also support the Sieve relational [RFC5231] extension, in addition to the extensions described here. All examples below assume the relational extension is present.

2. Conventions Used in This Document

Conventions for notations are as in [RFC5228] Section 1.1.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

The term "spam" is used in this document to refer to unsolicited or unwanted email messages. This document does not attempt to define what exactly constitutes spam, or how it should be identified, or what actions should be taken when detected.

The term "virus" is used in this document to refer to any type of message whose content can cause malicious damage. This document does not attempt to define what exactly constitutes a virus, or how it should be identified, or what actions should be taken when detected.

3. Sieve Extensions

3.1. General Considerations

The "spamtest" and "virustest" tests described below evaluate the results of implementation-specific spam and virus checks in a portable way. The implementation may, for example, check for third-party spam tool headers and determine how those map into the way the test commands are used. To do this, the underlying Sieve implementation provides a normalized result string as one of the inputs to each test command. The normalized result string is considered to be the value on the left-hand side of the test, and the comparison values given in the test command are considered to be on the right-hand side.

The normalized result starts with a digit string, with its numeric value within the range of values used by the specific test, indicating the severity of spam or viruses in a message or whether any tests were done at all. This may optionally be followed by a space (%x20) character and arbitrary text, or in one specific case a single keyword is returned. The numeric value can be compared to specific values using the Sieve relational [RFC5231] extension in conjunction with the "i;ascii-numeric" comparator [RFC4790], which will test for the presence of a numeric value at the start of the string, ignoring any additional text in the string. The optional text can be used to carry implementation-specific details about the tests and descriptive comments about the result. Tests can be done using standard string comparators against this text if it helps to refine behavior; however, this will break portability of the script as the text will likely be specific to a particular implementation.

In addition, the Sieve relational [RFC5231] ":count" match type can be used to determine if the underlying implementation actually did a test. If the underlying spam or virus test was done, the ":count" of the normalized result will return the numeric value "1", whilst if the test was not done, or the Sieve implementation could not determine if a test was done or not done, the ":count" value will be "0" (zero).

3.2. Test spamtest

```
Usage:    spamtest [":percent"] [COMPARATOR] [MATCH-TYPE]
          <value: string>
```

Sieve implementations that implement the "spamtest" test use an identifier of either "spamtest" or "spamtestplus" for use with the capability mechanism.

If the ":percent" argument is not used with any spamtest test, then one or both of "spamtest" or "spamtestplus" capability identifiers MUST be present.

If the ":percent" argument is used with any spamtest test, then the "spamtestplus" capability identifier MUST be present. Sieve implementations MUST return an error if the ":percent" argument is used and "spamtestplus" is not specified.

In the interests of brevity and clarity, scripts SHOULD NOT specify both "spamtestplus" and "spamtest" capability identifiers together.

The "spamtest" test evaluates to true if the normalized spamtest result matches the value. The type of match is specified by the optional match argument, which defaults to ":is" if not specified.

3.2.1. spamtest without :percent Argument

When the ":percent" argument is not present in the "spamtest" test, the normalized result string provided for the left-hand side of the test starts with a numeric value in the range "0" (zero) through "10", with meanings summarized below:

spamtest value	interpretation
0	message was not tested for spam, or Sieve could not determine whether any test was done
1	message was tested and is clear of spam
2 - 9	message was tested and may contain spam; a higher number indicates a greater likelihood of spam
10	message was tested and definitely contains spam

The underlying Sieve implementation will map whatever spam check is done into this numeric range, as appropriate.

Examples:

```
require ["spamtest", "fileinto", "relational", "comparator-
         i;ascii-numeric"];
```

```
if spamtest :value "eq" :comparator "i;ascii-numeric" "0"
{
    fileinto "INBOX.unclassified";
}
elsif spamtest :value "ge" :comparator "i;ascii-numeric" "3"
{
    fileinto "INBOX.spam-trap";
}
```

In this example, any message that has not passed through a spam check tool will be filed into the mailbox "INBOX.unclassified". Any message with a normalized result value greater than or equal to "3" is filed into a mailbox called "INBOX.spam-trap" in the user's mailstore.

3.2.2. spamtest with :percent Argument

When the ":percent" argument is present in the "spamtest" test, the normalized result string provided for the left-hand side of the test starts with a numeric value in the range "0" (zero) through "100", with meanings summarized below:

spamtest value	interpretation
0	message was tested and is clear of spam, or was not tested for spam, or Sieve could not determine whether any test was done
1 - 99	message was tested and may contain spam; a higher percentage indicates a greater likelihood of spam
100	message was tested and definitely contains spam

The underlying Sieve implementation will map whatever spam check is done into the numeric range, as appropriate.

To determine whether or not the message was tested for spam, two options can be used:

a. a test with or without the ":percent" argument and ":count" match type, testing for the value "0" as described in Section 3.1.

b. a test without the ":percent" argument using the ":value" match type, testing for the normalized result value "0" as described in Section 3.2.1.

Examples:

```
require ["spamtestplus", "fileinto", "relational",
         "comparator-i;ascii-numeric"];

if spamtest :value "eq"
            :comparator "i;ascii-numeric" "0"
{
    fileinto "INBOX.unclassified";
}
elsif spamtest :percent :value "eq"
               :comparator "i;ascii-numeric" "0"
{
    fileinto "INBOX.not-spam";
}
elsif spamtest :percent :value "lt"
               :comparator "i;ascii-numeric" "37"
{
    fileinto "INBOX.spam-trap";
}
else
{
    discard;
}
```

In this example, any message that has not passed through a spam check tool will be filed into the mailbox "INBOX.unclassified". Any message that is classified as definitely not containing spam (normalized result value "0") will be filed into the mailbox "INBOX.not-spam". Any message with a normalized result value less than "37" is filed into a mailbox called "INBOX.spam-trap" in the user's mailstore. Any other normalized result value will result in the message being discarded.

Alternatively, the Sieve relational [RFC5231] ":count" match type can be used:

Examples:

```
if spamtest :percent :count "eq"
            :comparator "i;ascii-numeric" "0"
{
    fileinto "INBOX.unclassified";
}
```

```
elsif spamtest :percent :value "eq"
               :comparator "i;ascii-numeric" "0"
{
    fileinto "INBOX.not-spam";
}
elsif spamtest :percent :value "lt"
               :comparator "i;ascii-numeric" "37"
{
    fileinto "INBOX.spam-trap";
}
else
{
    discard;
}
```

This example will result in exactly the same behavior as the previous one.

3.3. Test virustest

```
Usage:    virustest [COMPARATOR] [MATCH-TYPE]
          <value: string>
```

Sieve implementations that implement the "virustest" test have an identifier of "virustest" for use with the capability mechanism.

The "virustest" test evaluates to true if the normalized result string matches the value. The type of match is specified by the optional match argument, which defaults to ":is" if not specified.

The normalized result string provided for the left side of the test starts with a numeric value in the range "0" (zero) through "5", with meanings summarized below:

virustest value	interpretation
0	message was not tested for viruses, or Sieve could not determine whether any test was done
1	message was tested and contains no known viruses
2	message was tested and contained a known virus that was replaced with harmless content
3	message was tested and contained a known virus that was "cured" such that it is now harmless
4	message was tested and possibly contains a known virus
5	message was tested and definitely contains a known virus

The underlying Sieve implementation will map whatever virus checks are done into this numeric range, as appropriate. If the message has not been categorized by any virus checking tools, then the virustest result is "0".

Example:

```
require ["virustest", "fileinto", "relational", "comparator-
         i;ascii-numeric"];

if virustest :value "eq" :comparator "i;ascii-numeric" "0"
{
    fileinto "INBOX.unclassified";
}
if virustest :value "eq" :comparator "i;ascii-numeric" "4"
{
    fileinto "INBOX.quarantine";
}
elsif virustest :value "eq" :comparator "i;ascii-numeric" "5"
{
    discard;
}
```

In this example, any message that has not passed through a virus check tool will be filed into the mailbox "INBOX.unclassified". Any message with a normalized result value equal to "4" is filed into a

mailbox called "INBOX.quarantine" in the user's mailstore. Any message with a normalized result value equal to "5" is discarded (removed) and not delivered to the user's mailstore.

4. Security Considerations

Sieve implementations SHOULD ensure that "spamtest" and "virustest" tests only report spam and virus test results for messages that actually have gone through a legitimate spam or virus check process. In particular, if such checks rely on the addition and subsequent checking of private header fields, it is the responsibility of the implementation to ensure that such headers cannot be spoofed by the sender or intermediary and thereby prevent the implementation from being tricked into returning the wrong result for the test.

Server administrators must ensure that the virus checking tools are kept up to date, to provide reasonable protection for users using the "virustest" test. Users should be made aware of the fact that the "virustest" test does not provide a 100% reliable way to remove all viruses, and they should continue to exercise caution when dealing with messages of unknown content and origin.

Beyond that, the "spamtest" and "virustest" extensions do not raise any security considerations that are not present in the base [RFC5228] protocol, and these issues are discussed in [RFC5228].

5. IANA Considerations

The following templates specify the IANA registration of the Sieve extensions specified in this document. The registrations for "spamtest" and "virustest" replace those from [RFC3685]:

5.1. spamtest Registration

```
To: iana@iana.org
Subject: Registration of new Sieve extension

Capability name: spamtest
Description:     Provides a test to check for varying likelihood of
                 an email message being spam.
RFC number:      RFC 5235
Contact address: The Sieve discussion list <ietf-mta-filters@imc.org>
```

This information has been added to the list of Sieve extensions given on http://www.iana.org/assignments/sieve-extensions.

5.2. virustest Registration

To: iana@iana.org
Subject: Registration of new Sieve extension

Capability name: virustest
Description: Provides a test to check for varying likelihood of there being malicious content in an email message.
RFC number: RFC 5235
Contact address: The Sieve discussion list <ietf-mta-filters@imc.org>

This information has been added to the list of Sieve extensions given on http://www.iana.org/assignments/sieve-extensions.

5.3. spamtestplus Registration

To: iana@iana.org
Subject: Registration of new Sieve extension

Capability name: spamtestplus
Description: Provides a test to check for varying likelihood of an email message being spam, possibly using a percentage range.
RFC number: RFC 5235
Contact address: The Sieve discussion list <ietf-mta-filters@imc.org>

This information has been added to the list of Sieve extensions given on http://www.iana.org/assignments/sieve-extensions.

6. References

6.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4790] Newman, C., Duerst, M., and A. Gulbrandsen, "Internet Application Protocol Collation Registry", RFC 4790, March 2007.

[RFC5228] Guenther, P., Ed., and T. Showalter, Ed., "Sieve: An Email Filtering Language", RFC 5228, January 2008.

[RFC5231] Segmuller, W. and B. Leiba, "Sieve Email Filtering: Relational Extension", RFC 5231, January 2008.

6.2. Informative References

[RFC3685] Daboo, C., "SIEVE Email Filtering: Spamtest and VirusTest Extensions", RFC 3685, February 2004.

Appendix A. Acknowledgments

Thanks to Mark E. Mallett, Tony Hansen, Jutta Degener, Ned Freed, Ashish Gawarikar, Alexey Melnikov, Nigel Swinson, and Aaron Stone for comments and corrections.

Appendix B. Important Changes since RFC 3685

Listed below are some of the major changes from the previous specification [RFC3685], which this one supersedes.

1. A ":percent" argument has been added to the "spamtest" test adding a new 0-100 numerical range for test results.

2. A "spamtestplus" requires item has been added to indicate the presence of this extension in scripts.

3. The "count" match type from [RFC5231] can now be used to determine whether or not a message was tested.

4. Clarified that "test not done" also means "Sieve system could not determine if a test was done".

Author’s Address

Cyrus Daboo

EMail: cyrus@daboo.name

Full Copyright Statement

Intellectual Property