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

C. Daboo
Cyrusoft International, Inc.

February 2004

SIEVE Email Filtering: Spamtest and VirusTest Extensions

Status of this Memo

Copyright Notice

Abstract

The SIEVE mail filtering language "spamtest" 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 values returned by the tests.

Table of Contents

1. Introduction and Overview

SIEVE scripts are frequently being used to do spam and virus filtering based on either 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. These tests return a string containing a range of numeric values that indicate the severity of spam or viruses in a message, or a string that indicates the message has not passed through any spam or virus checking tools. The spam and virus checks themselves are handled by the underlying SIEVE implementation in whatever manner is appropriate, and the implementation maps the results of these checks into the numeric ranges defined by the new tests. Thus a SIEVE implementation can have a spam test that implicitly checks for third-party spam tool headers and determines how those map into the spamtest numeric range.

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

Conventions for notations are as in [SIEVE] section 1.1, including use of [KEYWORDS].

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.

2. SIEVE Extensions

2.1. General Considerations

The "spamtest" and "virustest" tests described below both return a string that starts with a numeric value, followed by an optional space (%x20) character and optional arbitrary text. The numeric value can be compared to specific values using the SIEVE relational [RELATIONAL] extension in conjunction with the "i;ascii-numeric" comparator [ACAP], which will test for the presence of a numeric value at the start of the string, ignoring any additional text in the string. The additional text can be used to carry implementation specific details about the tests performed 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.

2.2. Test spamtest

Syntax: spamtest [COMPARATOR] [MATCH-TYPE] <value: string>

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

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

The spamtest result is a string starting 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
1	message was tested and is clear of spam
2 - 9	message was tested and has a varying likelihood of containing spam in increasing order
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 spamtest value greater than or equal to "3" is filed into a mailbox called "INBOX.spam-trap" in the user's mailstore.

2.3. Test virustest

Syntax: 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 virustest result matches the value. The type of match is specified by the optional match argument, which defaults to ":is" if not specified.

The virustest result is a string starting 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
1	message was tested and contains no known viruses
2	message was tested and contained a known virus which was replaced with harmless content
3	message was tested and contained a known virus which 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 virustest value equal to "4" is filed into a mailbox called "INBOX.quarantine" in the user's mailstore. Any message with a virustest value equal to "5" is discarded (removed) and not delivered to the user's mailstore.

3. Security Considerations

SIEVE implementations SHOULD ensure that "spamtest" and "virustest" tests can only occur for messages that have gone through a legitimate spam or virus check process. If such checks rely on the addition of special headers to messages, it is the responsibility of the implementation to ensure that such headers cannot be spoofed by the sender, to 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 [SIEVE] protocol, and these issues are discussed in [SIEVE].

4. IANA Considerations

The following templates specify the IANA registration of the Sieve extensions specified in this document:

4.1. spamtest registration

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

Capability name: spamtest
Capability keyword: spamtest
Capability arguments: N/A
Standards Track/IESG-approved RFC XXXX: this RFC
Person and email address to contact for further information:

Cyrus Daboo
Cyrusoft International, Inc.
5001 Baum Blvd., Suite 780,
Pittsburgh, PA 15213
U.S.A.

<mailto:daboo@cyrusoft.com>

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

4.2. virustest registration

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

Capability name: virustest
Capability keyword: virustest
Capability arguments: N/A
Standards Track/IESG-approved RFC XXXX: this RFC
Person and email address to contact for further information:

Cyrus Daboo
Cyrusoft International, Inc.
5001 Baum Blvd., Suite 780,
Pittsburgh, PA 15213
U.S.A.

<mailto:daboo@cyrusoft.com>

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

5. References

5.1. Normative References

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

[RELATIONAL] Segmuller, W., "Sieve Extension: Relational Tests", RFC 3431, December 2002.

[SIEVE] Showalter, T., "Sieve: A Mail Filtering Language", RFC 3028, January 2001.

5.2. Informative References

[ACAP] Newman, C. and J. Myers, "ACAP -- Application Configuration Access Protocol", RFC 2244, November 1997.

6. Acknowledgments

Thanks to Tony Hansen, Jutta Degener, Ned Freed, Ashish Gawarikar and Nigel Swinson for comments and corrections.

7. Intellectual Property Rights Statement

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

8. Author's Address

Cyrus Daboo
Cyrusoft International, Inc.
5001 Baum Blvd., Suite 780,
Pittsburgh, PA 15213
U.S.A.

EMail: daboo@cyrusoft.com

9. Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.