Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: August 26, 2008

T. Hansen
AT&T Laboratories
C. Daboo
Apple Inc.

February 23, 2008

Sieve Email Filtering: MIME part Tests, Iteration, Extraction, Replacement and Enclosure
draft-ietf-sieve-mime-loop-04

Abstract

This document defines extensions to the Sieve email filtering language to permit analysis and manipulation of the MIME body parts of an email message.

Note

This document is being discussed on the MTA-FILTERS mailing list, ietf-mta-filters@imc.org.

Table of Contents

1. Introduction

MIME messages ([RFC2045]) are often complex objects, consisting of many parts and sub-parts. This extension defines mechanisms for performing tests on MIME body parts, looping through the MIME body parts, extracting information from a MIME body part, changing the contents of a MIME body part, and enclosing the entire message within a wrapper.

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].

3. Sieve Loops

The base Sieve language has no looping mechanism. Given that messages may contain multiple parts, in order to support filters that apply to any and all parts, we introduce a new control command: "for_every_part", which is an iterator that walks though every MIME part of a message, including nested parts, depth first, and applies the commands in the specified block to each of them. The iterator will start with the first MIME part (as its current context) and will execute a command block (Sieve commands enclosed by {...}). Upon completion of this command block, the iterator advances to the next MIME part (as its current context) and executes the same command block again.

The iterator can be terminated prematurely by a new Sieve command, "break".

Usage: for_every_part block

Usage: break;

"for_every_part" commands can be nested inside other "for_every_part" commands. When this occurs, the nested "for_every_part" iterates over the MIME parts contained within the MIME part currently being targeted by the nearest enclosing "for_every_part" command. If that MIME part is a terminal MIME part (i.e. does not contain other MIME parts) then the nested "for_every_loop" is simply ignored.

Sieve implementations MAY limit the number of nested loops that occur

within one another, however they MUST support at least one nested loop inside another loop.

4. Changes to Sieve tests

This specification extends the base Sieve "header", "address" and "exists" tests to support targeting those tests at a specific MIME part or at all MIME parts in the enclosing scope.

4.1. Test "header"

The "header" test is extended with the addition of a new ":mime" tagged argument and its associated options.

Usage: header [:mime] [:anychild] [MIMEOPTS]
[COMPARATOR] [MATCH-TYPE]
<header-names: string-list> <key-list: string-list>

Usage: The definition of [MIMEOPTS] is:

Syntax: ":type" / ":subtype" / ":contenttype" /
":param" <param-list: string-list>

When the ":mime" tagged argument is present in the "header" test, it will parse the MIME header lines in the message so that tests can be performed on specific elements.

When used outside the context of a "for_every_part" iterator, and without an ":anychild" tagged argument, the "header" test will examine only the outer top-level RFC2822 headers of the message.

When used inside the context of a "for_every_part" iterator, and without an ":anychild" tagged argument, the "header" test will examine the headers associated with the current MIME part context from the loop.

When used outside the context of a "for_every_part" iterator, and with an ":anychild" tagged argument, the "header" test will examine all MIME body parts and return true if any of them satisfies the test.

When used inside the context of a "for_every_part" iterator, and with an ":anychild" tagged argument, the "header" test will examine the current MIME part context and all it's nested MIME body parts, returning true if any of them satisfies the test.

The "header" test with the ":mime" tagged argument can test various

aspects of certain structured MIME headers. These options are available:

:type parses the header assuming it has the format of a "Content-Type:" MIME header field, and tests the value of the MIME type specified in the header.

:subtype parses the header assuming it has the format of a "Content-Type:" MIME header field, and tests the value of the MIME subtype specified in the header.

:contenttype parses the header assuming it has the format of a "Content-Type:" MIME header field, and tests the combined value of the MIME type and subtype specified in the header.

:param parses the header looking for MIME parameters in the header. The supplied string-list lists the names of any parameters to be tested. If any one named parameter value matches the test string value, the test will return true.

Example:

```
require ["mime", "fileinto"];

if header :mime :type "Content-Type" "image"
{
    fileinto "INBOX.images";
}
```

In this example, any message that contains a MIME image type part at the top-level is saved to the mailbox "INBOX.images".

Example:

```
require ["mime", "fileinto"];

if header :mime :anychild :contenttype
          "Content-Type" "text/html"
{
    fileinto "INBOX.html";
}
```

In this example, any message that contains any MIME part with a content-type of "text/html" is saved to the mailbox "INBOX.html".

Example:

```
require ["mime", "for_every_part", "fileinto"];

for_every_part
{
    if allof (
      header :mime :param "filename" :contains
         "Content-Disposition" "important",
      header :mime :subtype "Content-Type" "pdf",
      size :over "100K")
    {
        fileinto "INBOX.important";
        break;
    }
}
```

In this example, any message that contains a MIME part that has a content-disposition with a filename parameter containing the text "important", has a content-subtype of "pdf" and is bigger than 100 Kb is saved to the mailbox "INBOX.important".

4.2. Test "address"

The "address" test is extended with the addition of a new ":mime" tagged argument, which takes a number of other arguments.

```
Usage:  address [:mime] [:anychild] [COMPARATOR]
   [ADDRESS-PART] [MATCH-TYPE]
   <header-list: string-list> <key-list: string-list>
```

When the ":mime" tagged argument is present in the "address" test, it will parse the MIME header lines as if they were standard address header lines in a message so that tests can be performed on specific elements.

The behavior of the ":anychild" tagged argument and the interaction with the "for_every_part" iterator is the same as for the extended "header" test Section 4.1.

Example:

```
require ["mime", "fileinto"];

if address :mime :is :all "content-from" "tim@example.com"
{
    fileinto "INBOX.part-from-tim";
}
```

In this example, any message that contains a MIME Content-From header

at the top-level matching the text "tim@example.com" is saved to the mailbox "INBOX.part-from-time".

4.3. Test "exists"

The "exists" test is extended with the addition of a new ":mime" tagged argument, which takes one other argument.

Usage: exists [:mime] [:anychild] <header-names: string-list>

When the ":mime" tagged argument is present in the "exists" test, the test is extended to check for the existence of MIME headers in MIME parts.

The behavior of the ":anychild" tagged argument and the interaction with the "for_every_part" iterator is the same as for the extended "header" test Section 4.1.

Example:

```
require ["mime", "fileinto"];

if exists :mime :anychild "content-md5"
{
    fileinto "INBOX.md5";
}
```

In this example, any message that contains a MIME Content-MD5 header in any MIME part is saved to the mailbox "INBOX.md5".

5. Action Replace

Usage: replace [:mime] [:subject string] [:from string]
<replacement: string>

The "replace" command is defined to allow a MIME part to be replaced with the text supplied in the command.

When used in the context of a "for_every_part" iterator, the MIME part to be replaced is the "current" MIME part. If the current MIME context is a multipart MIME part, the entire multipart MIME part is replaced, which would alter the MIME structure of the message by eliminating all of the children of the multipart part. (Replacing a non-multipart MIME part within a "for_every_part" loop context does not alter the overall message structure.) If the MIME structure is altered, the change takes effect immediately: the "for_every_part" iterator that is executing does not go into the no-longer existing

body parts, and subsequent "for_every_part" iterators would use the new message structure.

When used outside the context of a "for_every_part" loop, the MIME part to be replaced is the entire message.

If the :mime parameter is not specified, the replacement string is a text/plain part in UTF-8.

If the :mime parameter is specified, then the replacement string is, in fact, a MIME entity as defined in [RFC2045] section 2.4, including both MIME headers and content.

If the entire message is being replaced, a ":subject" parameter specifies a subject line to attach to the message that is generated. UTF-8 characters can be used in the string argument; implementations MUST convert the string to [RFC2047] encoded words if and only if non-ASCII characters are present. Implementations MUST preserve the previous Subject header as an Original-Subject header.

If the entire message is being replaced, a ":from" parameter may be used to specify an alternate address to use in the From field of the message that is generated. The string must specify a valid [RFC2822] mailbox-list. Implementations SHOULD check the syntax and generate an error when a syntactically invalid ":from" parameter is specified. Implementations MAY also impose restrictions on what addresses can be specified in a ":from" parameter; it is suggested that values that fail such a validity check simply be ignored rather than causing the replace action to fail. Implementations MUST preserve the previous From header as an Original-From header.

6. Action Enclose

Usage: enclose <:subject string> <:headers string-list> string

A new Sieve action command is defined to allow an entire message to be enclosed as an attachment to a new message. After enclosure, subsequent actions affecting the message header or content use the newly created message instead of the original message; this means that any use of a "replace" action or other similar actions should be executed before the "enclose" action.

If multiple "enclose" actions are executed by a script, only the text specified on the last one is used when creating the enclosed message. This action does not affect messages that are forwarded via a "redirect" action.

Specifically, the original message becomes a multipart/mixed message with two parts: a text/plain portion with the string argument as its body, and a message/rfc822 portion with the original message enclosed. The Content-Type: header field becomes multipart/mixed. The Subject: header is specified by the :subject argument. Any headers specified by :headers are copied from the old message into the new message. If not specified by :headers, Date: and From: headers should be synthesized to reflect the current date and the user running the Sieve action.

7. Action extract_text

Usage: extract_text [MODIFIER] [":first" number] <varname: string>

The extract_text action may be used within the context of a "for_every_part" loop. Servers MUST support transcoding of any textual body part into UTF-8 for use with this action. This requires decoding any transfer encoding as well as transcoding from the indicated character set into UTF-8. It stores at most :first characters of the transcoded content of the current MIME body part in the variable identified by varname. If the :first parameter is not present, the whole content of the current MIME body part is stored. In either case the actually stored data MAY be truncated to conform to implementation-specific limit on variable length and/or on MIME body part length. If the transfer encoding or character set is unrecognized by the implementation or recognized but invalid, an empty string will result.

If extract_text is used outside the context of a "for_every_part" loop, the action will set the variable identified by varname to the empty string.

Modifiers are applied on the extracted text before it is stored in the variable. See [RFC5229] for details.

8. Sieve Capability Strings

A Sieve implementation that defines the "for_every_part" and "break" actions will advertise the capability string "for_every_part".

A Sieve implementation that defines the ":mime" and ":anychild" tagged arguments to the "header", "address" and "exists" tests will advertise the capability string "mime".

A Sieve implementation that defines the "replace" action will advertise the capability string "replace".

A Sieve implementation that defines the "enclose" action will advertise the capability string "enclose".

A Sieve implementation that defines the "extract_text" action will advertise the capability string "extract_text". Note that to be useful, the "extract_text" action also requires the "variables" [RFC5229] and "for_every_part" capabilities.

9. Examples

9.1. Example 1

A Sieve script to replace all the Windows executable attachments in a message would be:

```
require [ "for_every_part", "mime", "replace" ];
for_every_part
{
  if anyof (
         header :mime :contenttype :is "Content-Type" "application/exe",
         header :mime :param "filename"
           ["Content-Type", "Content-Disposition"] :matches "*.com" )
  {
    replace "Executable attachment removed by user filter";
  }
}
```

9.2. Example 2

A Sieve script to warn the user about executable attachment types would be:

```
require [ "for_every_part", "mime", "enclose" ];

for_every_part
{
  if header :mime :param "filename"
     ["Content-Type", "Content-Disposition"] :matches
       ["*.com", "*.exe", "*.vbs", "*.scr",
        "*.pif", "*.hta", "*.bat", "*.zip" ]
  {
    # these attachment types are executable
    enclose :subject "Warning" :text
WARNING! The enclosed message contains executable attachments.
These attachments types may contain a computer virus program
that can infect your computer and potentially damage your data.

Before clicking on these message attachments, you should verify
with the sender that this message was sent by them and not a
computer virus.
.
    break;
  }
}
```

9.3. Example 3

A Sieve script to extract subject and text out of messages from the boss:

```
require ["mime", "variables", "extract_text"];

if header :contains "from" "boss@example.org"
{
  # :matches is used to get the value of the Subject header
  if header :matches "Subject" "*"
  {
    set "subject" "${1}";
  }

  # extract the first 100 characters of the first text/* part
  for_every_part
  {
    if header :mime :type :is "Content-Type" "text"
    {
      extract_text :first 100 "msgcontent";
      break;
    }
  }

  # if it's not a 'for your information' message
  if not header :contains "subject" "FYI:"
  {
    # do something using ${subject} and ${msgcontent}
    # such as sending a notification using a
    # notification extensions
  }
}
```

10. Acknowledgements

Comments from members of the MTA Filters Working Group, in particular Ned Freed, Kjetil Torgrim Homme, Mark Mallett, Alexey Melnikov, Aaron Stone and Nigel Swinson are gratefully acknowledged.

11. Security Considerations

The "enclose" action creates an entirely new message, as compared to just redirecting or forwarding the existing message. Therefore, any site policies applicable to message submission should be enforced.

The looping specification specified here provides easier access to information about the message contents, which may also be achieved through other sieve tests. This is not believed to raise any additional security issues beyond those for the Sieve "envelope" and "body" [I-D.ietf-sieve-body] tests.

The system MUST be sized and restricted in such a manner that even malicious use of mime part matching does not deny service to other users of the host system.

Any change in a message content may interfere with digital signature mechanisms that include the body in the signed material.

All of the security considerations given in the base Sieve specification also apply to these extensions.

12. IANA Considerations

The Original-Subject: and Original-From: headers are to be registered in the Permanent Message Header Fields table.

The following templates specify the IANA registrations of the Sieve extensions specified in this document. This information should be added to the list of sieve extensions given on http://www.iana.org/assignments/sieve-extensions.

12.1. for_every_part capability

To: iana@iana.org

Subject: Registration of new Sieve extension

Capability name: for_every_part

Description: adds the "for_every_part" and "break" actions for iterating through MIME parts of a message.

RFC number: This RFC

Contact address: The Sieve discussion list <ietf-mta-filters@imc.org>.

12.2. mime capability

To: iana@iana.org

Subject: Registration of new Sieve extension

Capability name: mime

Description: adds the ":mime" and ":anychild" tagged arguments to the "header", "address" and "exists" tests.

RFC number: This RFC

Contact address: The Sieve discussion list
<ietf-mta-filters@imc.org>.

12.3. replace capability

To: iana@iana.org

Subject: Registration of new Sieve extension

Capability name: mime

Description: adds the "replace" action for replacing a MIME body part of a message.

RFC number: This RFC

Contact address: The Sieve discussion list
<ietf-mta-filters@imc.org>.

12.4. enclose capability

To: iana@iana.org

Subject: Registration of new Sieve extension

Capability name: mime

Description: adds the "enclose" action for enclosing a message with a wrapper.

RFC number: This RFC

Contact address: The Sieve discussion list
<ietf-mta-filters@imc.org>.

12.5. extract_text capability

To: iana@iana.org

Subject: Registration of new Sieve extension

Capability name: mime

Description: adds the "extract_text" action for extracting text from a MIME body part.

RFC number: This RFC

Contact address: The Sieve discussion list
<ietf-mta-filters@imc.org>.

13. Change History (to be removed prior to publication as an RFC)

13.1. draft-ietf-sieve-mime-02

minor syntax glitches in examples

Add clarification on "replace" affecting subsequent for_every_part loops?

Add IANA considerations for Original-Subject: and Original-From:.

Add note on "enclose" creating From: and Date: headers.

13.2. draft-ietf-sieve-mime-01

what happens when nested for_every_loop's

a "mime" shorthand for testing the type/subtype, without requiring

interactions with variables
notifications
notifications to calendar service
address tests, exists tests
mimeheader, mimeparameter tests

13.3. draft-ietf-sieve-mime-00

Changed title and text to emphasize MIME Tests.

Changed for.every.part to for_every_part.

Added :anychild to mime test. Default is to use the current context or outer envelope; specifying :anychild will look at all children.

Added clarifications to replacing parts affecting the structure.

Added :mime option to replace, ala draft-ietf-sieve-vacation-06.

Various other minor nit fixes.

13.4. draft-hansen-sieve-loop-01

Merged with draft-daboo-sieve-mime-00.txt.

13.5. draft-hansen-sieve-loop-02

Update to 3028bis reference.

Added 2119 conventions section.

Terminology/title tweaks.

Added informative references to body and editheader extensions.

Added description of nested loops.

Replaced mime test by extensions to header, address and exists tests.

13.6. draft-hansen-sieve-loop-03

after enclosure, subsequent actions affect newly created message

synthesis of Date/From headers by the enclose action is no longer controversial

Filled in Security Considerations

Picked up extract_text action from draft-ietf-sieve-notify

Expanded the IANA considerations section

13.7. draft-sieve-mime-loop-04

update reference for recent published rfcs

extract-text now required to do decode transfer encoding and transcode to UTF-8

removed editheader reference since its not actually used

several text changes as suggested by Nigel Swinson, including re-writes to abstract and introduction

tweaked IANA registrations

14. References

14.1. Normative References

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[RFC2047] Moore, K., "MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text", RFC 2047, November 1996.

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

[RFC2822] Resnick, P., "Internet Message Format", RFC 2822, April 2001.

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

14.2. Informative References

[I-D.ietf-sieve-body] Guenther, P. and J. Degener, "Sieve Email Filtering: Body Extension", draft-ietf-sieve-body-07 (work in progress), December 2007.

[RFC5229] Homme, K., "Sieve Email Filtering: Variables Extension", RFC 5229, January 2008.

Authors' Addresses

Tony Hansen
AT&T Laboratories
200 Laurel Ave.
Middletown, NJ 07748
USA

Email: tony+sieveloop@maillennium.att.com

Cyrus Daboo
Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
USA

Email: cyrus@daboo.name
URI: http://www.apple.com/

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).