axigen

SIEVE Language for Mail Filtering

September 8, 2006

GECAD Technologies

10A Dimitrie Pompei Blvd., BUCHAREST 2, ROMANIA

Tel.: +40 21 303 20 80

Fax: +40 21 303 20 81

Last modified: 9/22/2006

What is SIEVE?

SIEVE is a language created and used for mail filtering that broadens the filtering options generally provided by mail servers or Antispam/Antivirus applications. They work basically by comparing different keys using different comparators and comparison methods, against headers of a mail message. Based on the result of the comparison, you can apply different actions to the corresponding mail message, i.e. reject, discard, redirect, etc.

This language provides an extremely flexible filtering methodology, as users can define any number of script filters according to their needs. Designed to be easily implemented on either a mail client or mail server (such as Sendmail, Qmail, Axigen and so on), using SIEVE scripts does not depend on access protocol, mail architecture, and operating system.

SIEVE is designed as a proposed Internet Standard, as a result of a multi-vendor effort that has been discussed in various technical and standards-oriented public and private meetings since at least 1994.

Why Use SIEVE Filters

Mail traffic for most users has been increasing due to increased usage of e-mail, the emergence of unsolicited email as a form of advertising, and increased usage of mailing lists.

There are a number of reasons to use the SIEVE filtering system:

- You can create efficient and flexible rules. Scripts written in SIEVE are executed during final delivery, when the message is moved to the user-accessible mailbox. Therefore, it is reasonable to sort when the MTA deposits mail into the user's mailbox.
- SIEVE scripts are a safe filtering method since they do not operate on the mail content but only extract information from the mail header and take actions according to the pre-defined rules.
- As and addition to Antispam and Antivirus applications, you can use SIEVE scripts to also filter all legitimate emails, thus gaining speed and efficiency when using email communication.

The SIEVE Language

1. General aspects

SIEVE has a fixed form described as a standard but in can be improved by using extensions. The extension mechanism works if the system implements those extensions. In order to use an extension, it must be advertised at the beginning of the file (script) with a require clause.

require "extension_name" or require ["extension_name1", "extension_name2"]

The structure of SIEVE as described in the standard defines 5 actions: ***keep, fileinto, reject, discard, redirect*** which are self-explanatory. It also defines 3 control commands:

- <stop> - which stops the processing to that point
- <if elsif else> structure
- require command - which defines an extension of the language.

The if structure has the form:

if <test> <block>
elsif <test> <block>
else <block>

where a block is a block of commands (actions and control commands - including other ifs)

In the standard form, without any extensions, the test can be one of the following:

- ***address*** - tests a set of the address headers against a set of keys using different comparison methods;
- ***envelope*** - optional test;
- ***header*** - tests a set of the headers against a set of keys using different comparison methods;
- ***true, false*** – constants;
- ***allof*** <other tests> - logic and between several tests;
- ***anyof*** <other tests> - logic or between several tests;
- ***not*** <test> - negation of a test;
- ***exists*** - test if a set of headers exist;
- ***size*** - test against the size of a message;

A test can take 2 values: true or false.

2. Examples

A simple example of a SIEVE script that will move all mails that have "Spam" in the subject or are received from "spammer@example.com", in the Spam folder can be written as follows:

```
require "fileinto"

if anyof(address :is ["From", "Sender"] "spammer@example.com",
                header :matches "Subject" "Spam")
        {
                fileinto "Spam";
        }
```

Another, more complex example is a filter that will reject all mails that have a virus and are outgoing mails and not incoming mails. This latter example uses several extensions that need to be implemented.

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

if allof(virustest :value "eq" :comparator "i;ascii-numeric" "5",
        envelope :contains "From" ["domain1.org", "domain2.org"],
        not envelope :contains "To" ["domain1.org", "domain2.org"]) {
                reject "This mail is from domain.org to the world and contains a virus";
        }
```

For a more detailed presentation of the SIEVE language we also advise you to read the dedicated RFC 3028.

SIEVE filters in the AXIGEN Mail Server

Currently, AXIGEN uses SIEVE language for script filter definition. Different user-defined SIEVE scripts can be included in any AXIGEN Filtering System. When activated in AXIGEN, each filter is assigned a priority value. The notion of priority is used to define the order of filters in the filtering chain. This means that filters with higher priority will be applied first. All SIEVE filters can be activated at multiple levels: server, domain or account/mail list.

AXIGEN also implements the vacation SIEVE extension. This means that SIEVE script files can be created and applied for generating out-of-office type automatic replies. Thus, auto-generated messages can be send when the user of the account for which the vacation applies, is on vacation, out of office or in general away for an extended period of time. Although it is not a security function, vacation extension is an extra functionality available via script files. For an easy out-of-the office implementation in our mail server, please see this example available in our Knowledgebase:

For detailed instructions on SIEVE language and scripts implementation in AXIGEN, please see our online documentation.