CREATIVE MATH. & INF.
15 (2006), 113-116

A filtering servlet for improving the security of e-mail addresses

OVIDIU COSMA

ABSTRACT. Spam is a phenomenon that grows each day, because e-mail addresses are easy to get. It is relatively simple to write a spider program to search, index and validate the e-mail addresses published in the Internet. This article presents a filter for HTTP servers, that masks the e-mail addresses in both the static and dynamically created documents, leaving them visible only for the clients.

1. INTRODUCTION

The spam phenomenon is growing each day, because e-mail addresses are easy to get. It is relatively simple for anyone to build a database of e-mail addresses, because the majority of them are presented in HTML documents that are published in the Internet. For this purpose a spider application can be built, to automatically search and store e-mail addresses. They can be easily recognized, because of the '@' character in their format.

The spider applications are based on the HTTP protocol, which specifications can be found in [3]. There are also several spiders available for download in the Internet.

The e-mails can be easily verified, because the SMTP [2] protocol allows their validation, without actually sending e-mail messages. For this purpose, it is sufficient to open a connection on port 25 with the host that runs the SMTP server, then send the following messages:

HELO <some client domain>
MAIL FROM: <some return e-mail address>
RCPT TO: <e-mail to be verified>

The server response after the last message can be *250 OK* which means that the address exists on the server, or *550* followed by an error message in the case that the address is incorrect.

For the spammers that do not want to bother to run a spider application, there are several offers of huge databases of e-mail addresses in the Internet. It is relatively difficult for someone to stop unwanted messages, once his e-mail address was added to such a database. There are several applications built for blocking doubtful messages, based on sender, subject or content, but they are far from perfect.

Received: 11.09.2006. In revised form: 21.10.2006.
2000 *Mathematics Subject Classification*. 94A99
Key words and phrases. *Spam, servlets, filters, mask e-mail addresses.*

This article presents a filtering servlet that masks the e-mail addresses from the documents send by HTTP servers, in such a way that they can be presented on the client applications screens, but are unrecognizable for the spiders. The basic idea is that information can be somehow hidden from automatic computer programs, by inserting it in a distorted image.

The new HTML pages could be built using this artifice. E-mail addresses should be inserted in images, and never specified in clear text.

The filter presented in this paper is useful for improving the security of the existing HTML pages. It changes the format of all the e-mail addresses sent by the HTTP server, by replacing the '@' character with an image, in order to mask them from spider applications.

2. FILTERING THE HTTP SERVER'S RESPONSES

The filtering servlet is based on the following three classes: *GenericFilter*, *GenericResponseWrapper* and *EmailFilter*.

The *GenericFilter* class is a minimal implementation of the *Filter* interface, according to the Java Servlet Technology specification [1], [4].

```
//class GenericFilter
//-------------------
import javax.servlet.*;
import java.io.IOException;

public class GenericFilter implements Filter{
    private FilterConfig filterConfig;
    public void doFilter(final ServletRequest request,
    final ServletResponse response, FilterChain chain)
    throws IOException, ServletException{
      chain.doFilter(request,response);
    }
    public FilterConfig getFilterConfig(){
        return filterConfig;
    }
    public void init(FilterConfig config){
        this.filterConfig = config;
    }
    public void destroy(){
        this.filterConfig = null;
    }
}
```

The *GenericResponseWrapper* class is responsible for directing the HTTP server's response to a *CharArrayWriter* output stream, for future analysis. This class contains besides the constructor, a method for converting the response to the *String* type, and the *getWriter* method that is called by the next filter to find out where to send its output data. All the other necessary methods are implemented in the *HTTPServletResponseWrapper* class in the *javax.servlet.http* package [1], [4].

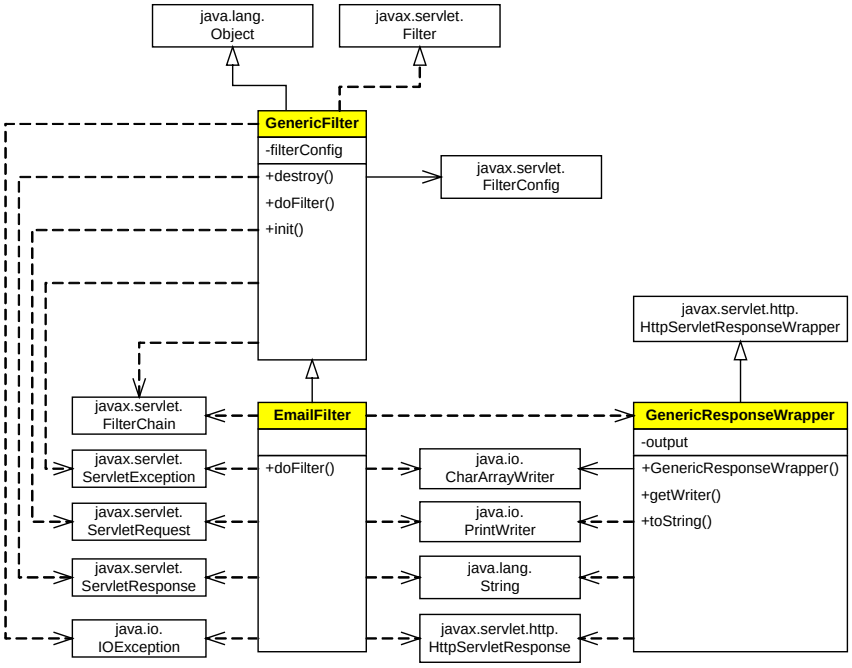
```
//class GenericResponseWrapper
```

```
//--------------------------------
import javax.servlet.http.*;
import java.io.*;
public class GenericResponseWrapper extends HttpServletResponseWrapper{
   CharArrayWriter output;
   public GenericResponseWrapper(HttpServletResponse response){
      super(response);
      output=new CharArrayWriter();
   }
   public String toString(){
      return output.toString();
   }
   public PrintWriter getWriter(){
      return new PrintWriter(output);
   }
}
```

The *EmailFilter* class adds to the *GenericFilter* class the necessary functionalities for masking the e-mail addresses. The filtering operations are performed by the *doFilter* method, that overrides a method defined in the *GenericFilter* class.

```
//class EmailFilter
//-----------------
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.HttpServletResponse;
public class EmailFilter extends GenericFilter {
   public void doFilter(final ServletRequest request,
   final ServletResponse response, FilterChain chain)
   throws IOException, ServletException{
      PrintWriter out=response.getWriter();
      GenericResponseWrapper responseWrapper = new
      GenericResponseWrapper((HttpServletResponse) response);
      chain.doFilter(request, responseWrapper);
      String responseData=responseWrapper.toString();
      CharArrayWriter filteredContent=new CharArrayWriter();
      int left=0, right;
      while((right=responseData.indexOf('@',left))!=-1){
         //replace all '@' characters with images
         filteredContent.write(responseData.substring(left,right));
         filteredContent.write("<img src=\"/images/aronde.gif\">");
         left=right+1;
      }
      //copy the end of the page (after the last '@')
      filteredContent.write(responseData.substring(left,
      responseData.length()));
      //set the content length of the response
      response.setContentLength(filteredContent.toString()
      .length());
      //send the filtered response
      out.write(filteredContent.toString());
      out.flush();
      out.close();
    }
}
```

The next image presents a UML diagram that indicates the relations between the classes of the e-mail filter.



3. CONCLUSIONS AND FUTURE WORK

A more sophisticated implementation would replace all the characters in the e-mail addresses with images, or would represent each e-mail address with a single image. In this case, the image encoder presented in [5] could be used. Eventually the images generated for representing e-mail addresses could be distorted by adding some noise, to make them even harder to be interpreted by computer applications.

The solution presented in this paper has a single disadvantage: the visitors of the filtered HTML pages will not be able to send e-mail messages with a single mouse click, and will have to complete the destination e-mail addresses by hand. But this is a fair price for the benefits of the enhanced security of the e-mail addresses.

REFERENCES

[1] Eric Armstrong, Jennifer Ball, Stephanie Bodoff, Debbie Bode Carson, Ian Evans, Dale Green, Kim Haase, Eric Jendrock, *The J2EE™ 1.4 Tutorial For Sun Java System Application Server Platform Edition*, Sun Microsystems 2005

[2] IETF, RFC 821 (SMTP), www.ietf.org

[3] IETF, RFC 2068 (HTTP 1.1), www.ietf.org

[4] Jason Hunter, Servlet 2.3: New features exposed, Java World, june 2001

[5] Jef Poskanzer, *Gif Encoder*, http://www.acme.com

NORTH UNIVERSITY OF BAIA MARE
DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE
VICTORIEI 76, 430122 BAIA MARE, ROMANIA
E-mail address: cosma@alphanet.ro