

Design spam-proof homepages

Private Address

Purveyors of unsolicited mail harvest most of their addresses from Internet homepages. This article tells you how to publish your own address and at the same time keep it hidden from automatic harvesting programs.

BY TOBIAS EGGENDORFER



Hannes Keller, visipix.com

Spam is becoming more of a nuisance and really hard to fight. Although spam filters use more or less successful **heuristics** to separate the chaff from the grain, in contrast to the computer virus scenario spammers are often one step ahead of defensive techniques, and continually develop new methods to bypass all the rules that are used by the filters.

The Root of the Problem

Strangely enough, over-eager administrators contribute to the downfall of their own defenses by sending discarded messages back to spammers, detailing why the filter treated each message as spam. Even without help spammers are continually on the lookout for new ways of bypassing filters. We should thus avoid thinking of a filter as an all-encompassing solution.

Another approach is to squash spam at its source, as evidenced by an increasing lobby in favor of using authenticated **SMTP** as a standard. A user wanting to transmit an email message would then need to authenticate to the mail server

using his IP address or a password – few providers have already implemented this method.

Spammers mainly harvest target addresses from publicly accessible websites [1] according to a study by the "Center for Democracy and Technology" (CDT). During the process of its investigations, the CDT deliberately published email addresses specifically created for the purpose of the experiment on homepages, in newsgroups and with various Web services. 97.3% of the 8842 messages received by these addresses, and

classified as unsolicited email advertising, were addressed to the mail addresses published on Web pages (see Figure 1).

Based on the results of the investigation, it would certainly seem to make sense to avoid using mail addresses that can be exploited by spammers when designing Web pages. In fact, the authors of the report state that it is worthwhile obfuscating mail addresses on existing Web pages as there was a drop in spam messages after removing the offending addresses (see Figure 2).

Listing 1: Storing in the HTML head

```

01 <HTML>
02 <HEAD>
03 <TITLE>Sample Page</TITLE>
04 <SCRIPT LANGUAGE="JavaScript">
05 <!--
06 mailaddress =
    'user@example.com';
07 //-->
08 </SCRIPT>
09 </HEAD>
10 <BODY>
11 [...]
12 <SCRIPT LANGUAGE="JavaScript">
13 <!--
14 document.write('<A
    HREF="mailto:'+mailaddress+'>'+mailaddress+'</A>');
15 //-->
16 </SCRIPT>
17 [...]
18 </BODY>
19 </HTML>

```

Address Harvesting

The approach that most spammers adopt is quite primitive. Starting on an arbitrary website, they simply store any *mailto:* links (that is references to mail addresses) they find. They then go on to follow any other links on the pages, and repeat the procedure until that tree of page links is exhausted.

By continuing this process, spammers will eventually arrive at a page referred to only by a single link. In doing so, they use techniques typical of search engines to scan the whole of the Web. It is not difficult to write a program that automates this task – a so-called “spider” or “harvester”. After removing duplicate addresses, the spammer is left with a collection of potential victims to use in spamming.

It is possible to design a simple email address harvester using standard Linux tools: *wget*, *sed*, *tr*, *sort*, and *uniq*. The results that you can achieve are quite amazing.

wget walks across websites, *sed* searches the pages for email addresses. *tr* provides uniform capitalization, *sort* sorts the mail addresses alphabetically, and *uniq* eliminates any duplicates. When I tested this approach on my own homepage, I harvested over 90 different email addresses in only in only eight minutes, none of them pointing to me. Choosing a starting page with more links and ignoring the convention of parsing the **robots.txt** [2] file, will return far more results in the same time.

Not supplying an email address at all is typically not the kind of solution that website owners prefer. After all, the idea of a website is to provide an additional communication channel. In some countries, website owners are required by law to provide an address (this is the case in Germany, for example).

Obfuscation Techniques

There are any number of popular techniques for obfuscating mail addresses, such as the *remove_to_mail_me* approach, which would use *someone@remove_to_mail_me.example.com* instead of *someone@example.com*, for example. Not all users remember to remove the middle section before clicking on *Reply*, and some times it is not sure what is meant to be removed. Again, legal restrictions may prevent some commercial website owners from using obfuscated addresses.

There is also an increasing tendency for spammers to spoof sender addresses. Error messages do not reach the spammer, but an unsuspecting third party, and this in turn can lead to mail servers crashing due to excessive loads.

The CDT [1] report mentioned previously suggests encoding mail addresses on homepages as **HTML entities**, where *user@example.com* would become:

```
&#117;&#115;&#101;&#114;&#064;&#101;&#120;&#097;&#109;&#112;&#108;&#101;&#46;&#099;&#111;&#109
```

Browsers have no trouble reading the address, but harvesters fail to recognize the searched pattern held within the source code.

In the CDT report, addresses encoded in this way did not receive any spam. Our primitive harvester found 10 addresses of this kind when we ran it.

As the use of this format is on the increase, it is to be expected that spider programs will soon be capable of automatically converting entity addresses back to valid email addresses. In the long term, the entity address has no real advantage over cleartext for protection against harvesters.



Figure 1: Spammers mainly harvest target addresses off the Web.

JavaScript to the Rescue?

Most spider programs cannot handle **JavaScript**. This allows website owners to use Java to obfuscate email addresses.

Listing 1 shows how you can specify an address in the *Head* of a HTML page, and then use the *document.write()* JavaScript function to output the address. This variant means that the website visitor still sees the address in cleartext on the page, but a specialized search will not come up with the goods.

There are various options for assigning the email-address to the variable. The easiest -- and maintainer-friendliest -- is loading the value from an external javascript file. Harvesters do not currently evaluate this type of file.

Unfortunately, some browsers only load external javascripts after having displayed the page. In this case, *document.write* will fail, as it tries to output an empty variable. Its value is assigned to late.

Also, a cunning spammer might find a method to harvest all the JavaScript files on a page, and then search these files for email addresses.

There is a workaround for the *document.write()* problem that affects some browsers. Instead of using a HTML link

GLOSSARY

Heuristics: (From Greek “*heuristic*”: to find, discover) Searching for patterns based on rules of thumb that have a high probability of success. Theoretically this makes the results unreliable, but the search operation is a lot quicker than a precise computation.

SMTP: The “Simple Mail Transfer Protocol” accepts and forwards email messages.

robots.txt: Files of this name on Web servers contain information on the pages that search

engines should not automatically search, among other things.

ASCII: The “American Standard Code for Information Interchange” assigns a number, or ASCII code, to all letters, numbers and special characters.

JavaScript: Script language for use on websites. If JavaScript is enabled for a browser, the browser interprets the language and executes the embedded commands.

XOR: “Exclusive OR” is a computational method common in binary math, and represented by the ^ symbol in JavaScript. It can be used for symmetrical data encryption.

HTML entity: This HTML encoding prints characters in a format with a combination of the &# prefix, followed by the ASCII code and a semicolon at the end: thus “u” corresponds to “u”.

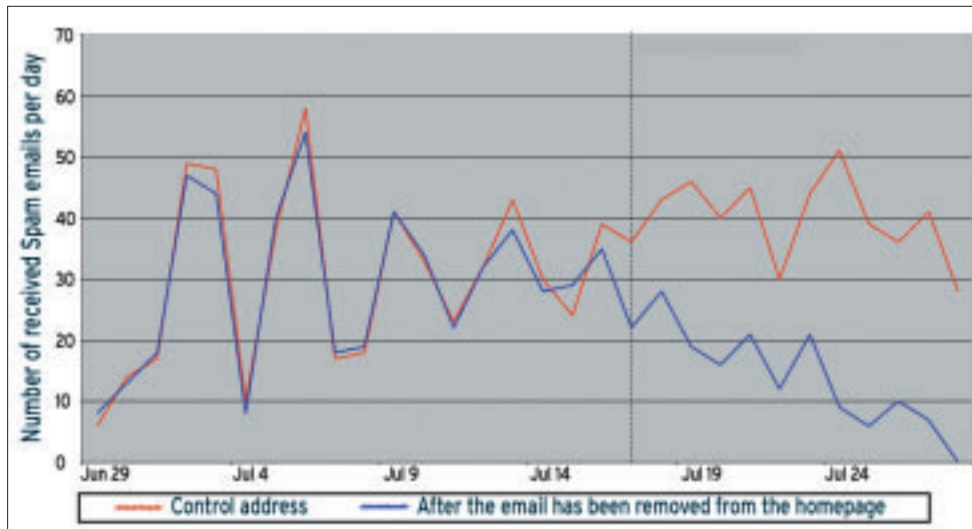


Figure 2: It is worthwhile removing existing email addresses from a homepage.

(``) to point directly to the mail address, you can have JavaScript take care of the job. Instead of using JavaScript to output HTML, Listing 2 shows the JavaScript link function, `document.location.href`.

Extremely basic XOR encryption is all it takes to prevent a search program from finding a mail address. This encryption method provides very little in the line of security from a cryptographic point of view, but it is extremely effective in the case of spammers who need maximum results in as short a time as possible. The address uses an encrypted format. Doing so requires time and brain, both is not available to spammers respectively their harvesters. The spammer would need to understand, and reverse engineer, the JavaScript commands to obtain a clear-text address.

Listing 3 below, encrypts the user name in a mail address, that is the name

component before the @; the `document.location.hostname` JavaScript function retrieves the missing elements in clear text from the browser address line. This example only works if the server domain matches the domain for the email address. As an alternative, you could use the same algorithm below to encrypt the remaining address components.

The encryption procedure is easily extensible, however, both the encryption procedure and the key are available within the script. This allows anyone who can run the JavaScript code to view the mail address in the clear, and of course this is necessary to allow human visitors to read the address.

Big Advantages

The biggest advantage that this method offers is that clients that can not interpret JavaScript will not recognize the email address. At present (and let's hope it stays that way), this is an ability that spi-

der programs for harvesting addresses do not possess.

You can easily combine the JavaScript snippets listed in this article, adding the encryption facilities from Listing 3 to the JavaScript link in Listing 2, for example.

It would be great to be able to say that only address harvesters trip up over JavaScript, but unfortunately text-based browsers like Lynx have the same problem. Also, many users disable JavaScript in graphical browsers for security reasons. Note that JavaScript pages will prevent these users from getting in touch with you.

Downsides

The obfuscation methods described so far work for email addresses and Web links. If these techniques were applied globally, address harvesters would have a hard time of it: on the downside, so would search engines like Google, and human surfers without JavaScript-capable browsers.

To avoid forcing the visitors to your website to use JavaScript, you might like to use a simple trick, which unfortunately again rules out users with text-only browsers. Include an image file on the page that displays an image of your email address. As the address does not show up in the text content of the page, harvesters have no chance of gleaning it. Spammers can not resort to OCR software to automatically harvest addresses from images – after all, any image file on the Web could contain a mail address. This also provides a loophole for website owners faced with a legal requirement to publish an electronic mail address.

You can link an image without revealing an email address: use a contact form without a visible target address to forward legitimate messages from website visitors to the site owner.

Listing 2: JavaScript with address links

```

01 <HTML>                                10 }
02 <HEAD>                                  11 //-->
03 <TITLE>Sample Page</TITLE>             12 </SCRIPT>
04 <SCRIPT LANGUAGE="JavaScript">        13 </HEAD>
05 <!--                                     14 <BODY>
06 mailaddress =                          15 [...]
   'user@example.com';                    16 <A
07 function mailMe()                       HREF="javascript:mailMe();">Ma
08 {                                         il sender</A>
09                                         17 [...]
   document.location.href="mailto        18 </BODY>
   :"+mailaddress;                        19 </HTML>

```

GLOSSARY

Flash: This proprietary format by Macromedia supports combinations of animation, video, sound, and images on websites. Browsers need a plug-in to view content in Flash format.

A **Flash** animation that displays a clickable address is yet another variant. However, you should be aware that this approach will exclude more visitors.

Hitting Back

If you have your own Internet domain, you can use a dynamic website to dis-

cover the origins of address harvesters. To do so, create an email link that is continually updated on the page – use the time of day, the date and the IP address of the current user to update the link, thus forcing visitors to supply their email addresses when they need to load your page.

If this address is spammed, you can then just read the source address for the unsolicited message from the address that they have sent to – and this can be important, if you need to go to litigation against the spammers. ■

Listing 3: Encrypted mail address

```

01 <HTML>                                12 //-->
02 <HEAD>                                  13 </SCRIPT>
03 <TITLE>Sample page</TITLE>             14 </HEAD>
04 <SCRIPT LANGUAGE="JavaScript">        15 <BODY>
05 <!--                                    16 [...]
06 local = new Array                      17 <SCRIPT LANGUAGE="JavaScript">
   (194,196,210,197);                    18 <!--
07 local_part = '';                       19 document.write('<A
08 for (i=0;                               HREF="mailto:'+mailaddress+'">
09     i<local.length;                    '+mailaddress+'</A>');
10     local_part +=                       20 //-->
   String.fromCharCode(local[i] ^        21 </SCRIPT>
   183), i++) ;                            22 [...]
11 mailaddress = local_part +             23 </BODY>
   String.fromCharCode(64) +             24 </HTML>
   document.location.hostname;
```

THE AUTHOR

Tobias Eggendorfer has been working as a freelance IT consultant and lecturer since 1999. Tobias's main focus is IT security, spam prevention, net-working technologies, and databases.



In his leisure time Tobias works as a volunteer for the local ambulance service, particularly in crisis intervention. His homepage can be found at <http://www.eggendorfer.info>.

INFO

[1] "Why am I getting all this spam?": <http://www.cdt.org/speech/spam/030319spamreport.html>

[2] robots.txt: <http://www.robotstxt.org/>