STC Network Systems
Web Site Services & Consulting

Protecting Web pages from e-mail address harvesting

compiled by Michael Foster, STC Network Systems

That is the concern every Webmaster should have: how to protect e-mail addresses from “harvesting”. While the very original techniques for building Web sites were sufficient in the early stages of the Internet, they open doors to hackers and spammers today. How your Web pages are coded and designed plays a very important role in your Web security. So, how can you protect Web pages from e-mail harvesting?

First, I should explain what e-mail address harvesting is, why it is bad and how it relates to you and then let's talk about how to avoid your Web site becoming a victim of e-mail address harvesting.

What is e-mail address harvesting?

E-mail harvesting is the process of obtaining lists of e-mail addresses using various methods for use in bulk e-mail or other purposes usually grouped as "spam". Speaking of Web sites, spammers have programs which crawl or "spider" through the Web looking for e-mail addresses. E-mail address harvesting is done using special software known as "harvesting bots", "harvesting robots", or just "harvesters" which crawl through Web pages and capture every e-mail address they find.

E-mail address harvesting is bad because once your or your client's, employee's or family member's e-mail address gets into spammers' lists, it will get flooded with spam and trash very quickly. And how does it relate to you? If you run, design or build a Web page, you need to take preventive steps to protect e-mail addresses from getting harvested. (If not, you could even be held liable in some commercial settings.) And if your e-mail address is displayed somewhere, well you do not want to have to create a new e-mail account soon just because it gets spammed, right?

Why would anyone want to harvest e-mail addresses?

Spam. Phishing. Spoofing. Direct marketing. All these techniques are used with one goal: to sell you something or to conduct some illegal activity leading to getting some monetary or other benefit from you. If someone with bad intentions has your e-mail address, you can become target of his or her bad intentions. Harvesting e-mail addresses alone can make money as well. There are many spammers that just collect e-mail address lists only to sell them to marketing companies.

If you are responsible for a Web space, be aware that e-mail addresses published on your pages are vulnerable to being added to unsolicited e-mail (spam) lists and could thus receive unwanted e-mail. Spammers can collect e-mail addresses by running automated harvesting scripts to parse static Web pages, one by one, looking for strings of characters that appear to be e-mail addresses. Such programs can catch thousands of addresses in a very short time.

To test the security of your own address, visit a search engine such as Google and enter your e-mail address. You may see a number of results which is at least how many are of your e-mail address references were visible to harvesting scripts and then publicized, probably in undesirable places.

How we (and ‘bots) see e-mail addresses

Here’s how e-mail addresses are usually displayed at Web sites and then start with the easy stuff. E-mail addresses are often coded into Web pages like the following example:

```
<a href="mailto:foo@example.com">foo@example.com</a>
```

This example produces clickable foo@example.com. If you click this e-mail address, your mail client (i.e. Outlook) will open up with this e-mail address in the To: field. This e-mail format is a beauty for e-mail harvesting software, this is exactly what they are looking for and where they get majority of e-mail addresses.

Some people make the job for e-mail address harvesting software by writing out the e-mail address as shown in the following two examples.

```
<a href="mailto:foo@example.com">foo[AT]example[DOT]com</a>
```

-and-

```
foo[AT]example[DOT]com
```

This is a bit better than the plain HTML format, but notice that the first example still includes your correct e-mail address in the mailto field, so e-mail harvesting software still can find you. The second option leaves out the A HREF tag, so the link will not be clickable anymore and the visitor will have to copy your e-mail address and paste it into his or her e-mail client. Substituting @ with [AT] and dot with [DOT] is a nice idea, but there is nothing easier than telling the e-mail harvesting software "if you find [AT], replace it with @".

A good way to protect your e-mail address in a Web page is to fake it for the e-mail harvesting robot and let the human know that it has been faked.

```
<a href=mailto:foo@example[REMOVETHIS].com>foo@example[REMOVETHIS].com</a>
```

-or-

```
<a href=mailto:foo@com.example>foo@com.example</a>
```

These examples are not bad, but you have to really let the visitor know that he or she needs to fix the e-mail address before sending e-mail to it. Many people just blindly click, copy, past, so you really have to make this visible (perhaps by displaying the [REMOVETHIS] in red color or formatting with a strikethrough line). This e-mail harvesting protection technique works well against e-mail harvesting bots because even though they get the e-mail, it is an invalid one, hence you are safe. On the other hand, e-mails in this format may cause confusion to the user, if the idea is not described well.

Options and ideas to protect you and your client e-mail addresses

For members and employees of many organizations, consider obtaining an organization account and listing that rather than the personal e-mail address.

To help protect e-mail addresses from harvesting scripts, consider the methods listed below (though none is fully guaranteed).

* Use generic, changeable addresses
* Re-format addresses
* Substitute ASCII codes in addresses
* Web forms
* Build the mailto: link using a server script or JavaScript
* Use graphics in displaying addresses

Use Generic, Unchangeable addresses

Many organizations and Web masters aren't aware that their domain name can be used for e-mail as well as for browsing Web pages.

jsmith@organization.org
john.smith@domain.edu
marys@organization.org
mary_s@domain.edu

Most companies do this every successfully. Consider assigning generic e-mail addresses to positions, committees, departments and sections of your organization to protect an individual's personal or professional e-mail address:

president@organization.org
sub-committee@organization.org

If one of these addresses becomes "compromised", it can always be changed, perhaps on an annual or rotational basis (although once an e-mail address is on the spam lists, it will never get off no matter how many thousands of e-mails "bounce" back to the send as invalid):

president.2009@organization.org
secretary_09@organization.org

Re-format addresses

The simplest method for hiding addresses is to present them in a way that contains all necessary information but makes the address unusable without some modification. For example, insert spaces into the address:

username @ domain.edu
user AT organization.org

You can also list only the username next to an individual's name, and note the domain elsewhere on the page. The main drawback is that this method renders the address un-clickable. You may wish to add an explanatory statement to your page, for example:

"Email addresses on this page are displayed in a manner that will deter automatic address harvesting programs. This step is taken to reduce unsolicited e-mail sent to Indiana University addresses. We regret any inconvenience caused for our legitimate visitors."

Substitute ASCII codes in addresses

Present e-mail addresses by substituting ASCII codes for certain characters in the address, trusting the user's browser to translate the codes back into the correct characters. The format for ASCII codes is the & (ampersand symbol), followed by the # (pound sign), followed by a number corresponding to the character to be displayed, followed by a ; (semicolon). In an address, for example, you could substitute the ASCII code for both the @ (at sign), which is 64, and the . (period), which is 46, as follows:

username&#64;domain&#46;edu

When you enter the above code in your HTML, browsers render it as username@domain.edu , but harvesting scripts looking at the source will see only the ASCII codes; unless they have been designed to translate ASCII codes, they will be unable to recognize the code as an address. This technique can be effective in both the target and text of a mailto: link.

Consult an ASCII code table for information on other characters.

Web forms

Below are two options for controlling or limiting access to e-mail addresses using HTML Web forms:

* Create a link to a Web form asking users to enter their own address. Upon submission, the form e-mails the requested address to the user, and writes the transaction to a log.

* You can create a link going to a Web form where users enter a message, and the form then submits the message using a server script. For instructions for doing this on IU departmental Web pages, see Preventing Email Harvesting.

Build the mailto: link using a server script or JavaScript

Use scripts to emulate the function of a mailto: URL. The idea is to create a link on your page that submits the username and domain of the e-mail address to a program that builds the mailto: URL dynamically and returns it to the user's browser.

* For instructions and examples of this method for server scripts, see James Thornton's Redirect mailto: for Spam Prevention software page

http://jamesthornton.com/software/redirect-mailto.html.

* The following JavaScript function can also obscure mail addresses:

```
<A Href='javascript:window.location="mail"+"to:"+"user"+"@"+"domain"+"."+"com";'
onMouseOver='window.status="mail"+"to:"+"user"+"@"+"domain"+"."+"com"; return true;'
onMouseOut='window.status="";return true;'>Click here to send mail.</A>
```

This returns a mailto: link to user@domain.edu , but the username and domain appear broken up in the source HTML file, protecting them from harvest scripts. [This technique was taken from Mac Efficiency 101: Preventing Spam.]

The following is a JavaScript-only solution. It goes in the header of any page where you have e-mail addresses:

```
<SCRIPT Language="javascript">
     var gsDomain = "domain.com"; // ENTER YOUR DOMAIN HERE
      var gsPath = window.location.pathname;
      var gsPage = gsPath.substring(gsPath.lastIndexOf('/') + 1);
      if (gsPage.length == 0)
      {
          gsPage = "index.html"; // ENTER DEFAULT PAGE IF BLANK
      }

      function SendTo(tsUid,tsUname,tsSubj,tbPage)
      {
          var tsPage = '';

          if (tbPage == 'T')
          {
            if (tsSubj.length > 0)
            {
                tsPage = ": " + "&upage=" + gsPage;
            }
            else
            {
                tsPage = "&upage=" + gsPage;
            }
          }

          top.window.location.href = 'mailto:"' + tsUname + '" <' + tsUid + '@’ +
gsDomain + ‘>?subject=' + tsSubj + tsPage;
      }
</SCRIPT>
```

And this goes wherever you wish to have e-mail addresses on your page:

```
<A href="javascript:SendTo('userID','Name or Description', 'Optional Subject',
'T');">E-mail</A>
```

This JavaScript function allows clickable e-mail by building the address on-the-fly (1st parameter). You can also include the person’s name or description (2nd paremeter) for a nicer, formatted “To:” line. In addition, a default “Subject:” can be included as the 3rd parameter. The 4th parameter to this function is “T” true or “F” false to say whether you want the actual page name to appear in the “Subject:” line regardless of whether you’ve opted for a subject or not. The domain name is assigned at the top of the script.

Note: These require your visitors to have JavaScript enabled in their browsers; you may want to note this on your page.

Use a server-side PHP script

You can use a PHP script on the server that is not dependent on JavaScript being enabled.

Store this code as a script called “sendto.php”:

```
<?php
// sendto.php

// GET CURRENT DOMAIN NAME FROM HOST OR ENTER OWN DOMAIN NAME
$gsDomain = getenv("HTTP_HOST");
$gsDomain = "yourdomain.com";

// GET USER E-MAIL ADDRESS FROM COMMAND LINE PARAMETER
// EXAMPLE: <A Href="sendto.php?uid=userid">E-Mail</A>
$gsUserID = $_GET[uid];

// GET USER NAME OR DESCRIPTION FROM COMMAND LINE PARAMETER
// EXAMPLE: <A Href="sendto.php?uname=username">E-Mail</A>
// EXAMPLE: <A Href="sendto.php?uid=userid&uname=username">E-Mail</A>
$gsUserName = $_GET[uname];

// GET SUBJECT FROM COMMAND LINE PARAMETER
// EXAMPLE: <A Href="sendto.php?usubj=subject">E-Mail</A>
// EXAMPLE: <A Href="sendto.php?uid=userid&uname=username&usubj=subject">E-Mail</A>
$gsSubject = $_GET[usubj];

// GET CURRENT PAGE NAME
$gsPage = $_GET[upage];

// IF SUBJECT IS AVAILABLE, MAKE IT A "mailto" PARAMETER
if (strlen ($gsSubject) > 0)
{
    // EXAMPLE: mailto:uid@domain.com?subject=subject>
    $gsSubject = "?subject=".$gsSubject." ".$gsPage;
}
else
{
    if (strlen ($gsPage) > 0)
    {
      $gsSubject = "?subject=".$gsPage;
    }
}
// IF USER NAME/DESCRIPTION IS AVAILABLE, REFORMAT "mailto" LINE
// EXAMPLE: mailto:"User Name <uid@domain.com>"
if (strlen ($gsUserName) > 0)
{
    header ('Location: mailto:"'.$gsUserName.'"
<'.$gsUserID.'@'.$gsDomain.'>'.$gsSubject);
}
else
{
    header ('Location: mailto:'.$gsUserID.'@'.$gsDomain.$gsSubject);
}
exit();
?>
```

Then, insert lines such as these wherever you want e-mail addresses:

```
<A Href="sendto.php?uid=userid&uname=User Name or Description&usubj=Web
Contact&upage=">E-mail</A>

<A Href="/scripts/sendto.php?uid=jsmith&uname=John Smith&usubj=US
Department&upage=index.html">Email John Smith</A>
```

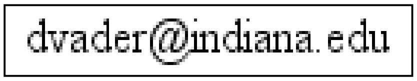
```
<A Href="/public_html/php/sendto.php?uid=support2&uname=Support Two &usubj=Our
Company&upage=staff.html">Staff Support</A>
```

Note that this is essentially the same script as the preceding JavaScript except that the page name is not determined automatically; you have to enter it if you want it to be shown on the “Subject:” line.

Usegraphics in displaying addresses

You could use graphics to display addresses. This works well as a companion to the previous methods in order to have a normal-looking, clickable e-mail address displayed on your page as the link to your CGI, JavaScript, or form. However, if your priorities require maximum security over user convenience, you should use this method by itself and instruct users to type the address into their e-mail program to send mail.

With this method, you create an image of some or all of each address. For highest security, represent the entire address with a graphic, for example:



dvader@indiana.edu

[human-readable text image]

Replacing the entire address requires the most work, as each graphic must be unique. However, this is the most secure, requiring a harvesting script to have optical character recognition or a human operator to harvest the address, if used in conjunction with one of the script methods above.

You could simply replace the @ sign with a picture of the same; however, the username and domain name are then readable and in close proximity to each other, and thus vulnerable. You might also consider using a graphic to represent everything in the address after the username, i.e., the @ sign and the domain.

For further explanation of the method of representing the @ sign graphically, see James Thornton's Graphic @ for Spam Prevention software page

http://jamesthornton.com/software/graphic@.html

-- Michael Foster <mfoster@stcllp.com>
Thanks to Indiana University Technology Services and Maxi-Pedia.com