Thinking outside the **[mail]** *box!*

A strategy for combating email address harvesting software

by

Geoff Vines

Contents

Who should read and apply this information?

Any **webmaster** who wants to have email links, on a web site that they are responsible for, without creating a ready meal for email harvesting software (spambots).

Any **web site owner** who is fed up with 'spam' email and would like their webmaster to help do something about it.

What's the problem?

You have your new web site complete with helpful email links, where appropriate, and it gets submitted up to your hosting facility. What happens next?

The 'spambots' beat a metaphorical path to your door, harvest the email addresses and flood your email inbox forever more.

The other rich source for email addresses is, of course, your email client's address book. So called 'spyware' that has managed to infiltrate your machine, collects up the addresses and sends them back to base. The solution to this, is as they say, another story.

Common solutions – so far!

Obfuscation – Encrypt the email address and when an end-user clicks on a link, have JavaScript decode it into a useable email address. The problem with this approach is that if JavaScript is turned off, the link doesn't get decoded and your end-user can't email you.

Cold link – By this I mean that the email address is shown but it isn't hot therefore not 'clickable'. This requires you to copy the email address into your email client before you can use it. Not exactly user friendly.

It also doesn't stop harvesting, as any software scanning the file, using regular expression searches, will easily recognise a valid email address. So you haven't actually achieved anything other than annoy the end-user.

Manual interpretation – This refers to an approach I saw recently on the web site of a 'security guru'. An email address is displayed on the page in the form – ***name <at> domain <dot> org***. You can't even cut-n-paste this. It requires the viewer to manually translate that into a valid email address.

Hide in graphic – This variant displays the email address as a graphic. That certainly stops the 'spambots' from reading it but if your end-user is partially sighted, using a screen reader or just has graphics turned off, then they can't read it either. Also, again, it's not directly 'clickable'.

And what about the accessibility issues?

Rethink the problem – and the solution

An end-user making email contact with you or someone in your organisation, via your web site, for the first time, is most unlikely to need all the sophistication of a full-blown email client.

I would suggest that 99.9% of the time, all that's required is a recipient, a sender, a subject and the message.

So why not do away with the need to use an email client at all?

Effectively construct your own mini email client within a web page using form features and embedded PHP code. PHP is widely available across all the major server platforms, is handled server side and has extensive email capabilities that are very simple to use, at least at the level we need for this solution. And in case you are wondering, it doesn't require database access either.

For the benefit of the non-technical reader, e.g., a web site owner, PHP code embedded in an html file is processed on the server before the file is served to the user's browser. In doing so, the 'raw' PHP code is removed from the file and therefore never seen by the outside world.

Because you have been polite and told them on the page, the end-user knows the name of the person or department with whom they are going to communicate but never knows the actual email address.

Importantly therefore – neither can the email harvesters.

Only when you respond to that first contact, assuming you do, using your normal email client will the originator of the communication become aware of your actual email address.

You can see this strategy in use at: 1ontheweb.net.

The code – and how it works

The following is not the code for a complete web page. It is the PHP and form code that you need to embed into your own page structure.

Also I leave it to you to provide the CSS code used in styling the form. I have assumed that whoever is making use of this information, from a coding point of view, will have their own preferences in this area.

Although the PHP code can be embedded within the web page, it is good practice and adds a level of security to place the code in a separate file, which is placed in a directory outside the web sites normal directory tree, and ‘included’ into your web page. An example is shown below:

```
<?php
    include ('../private/mailcheck.inc');
?>
```

As a minor aid to security, all my web page files have the file extension – html. The following line included in your server’s .htaccess file forces the file to be processed through the PHP interpreter before being served to the browser:

```
AddType application/x-httpd-php .html
```

The **very first** line in your html file must be:

```
<?php ob_start(); ?>
```

This stops any output being sent to the browser before we are ready.

Now the main parts of the code, the PHP section and the form definition, which are also available in separate files ready to insert into your page.

The PHP:

```
<?php
    // If the send button has been pressed, process this block.
    if (isset($_POST['sendasmail'])) {
    // These first four variable are also used to provide ‘sticky data’ in the form.
        $mailfor = $_POST['recipient'];
        $mailfrom = $_POST['sender'];
        $topic = $_POST{'subject'};
        $msgprompt = $_POST['mailmsg'];
        $block = '';

    // Now validate the data.
        if (!isset($mailfrom)) {
           $block = $block.'An email from: address is required.<br />';
        }
        else {
    // Define the regular expression to validate an email address.
           $email_pattern = '/^((".+")|(\w+([\.-]?\w+)*))@\w+([\.-
]?\w+)*(\.\w{2,6})+$/';
           if (!preg_match($email_pattern, $mailfrom)) {
               block = $block.'The email from: address is not a valid format.<br />';
           }
        }
        if (!isset($topic) || ($topic == 'please enter a subject')) {
           $block = $block.'A subject is required.<br />';
        }
        if (!isset($msgprompt) || ($msgprompt == 'enter your message here')) {
           $block = $block.'Please enter a message.';
        }
    // Check for completion errors and if none, send email.
        if (!$block == '') {
           $block = '<p>Please correct the following:<br /><br />'.$block.'</p>';
           print ($block);
        }
        else {
    // This is where the real email address is prepared for use. Obviously you
        need to replace the text in italics with something valid for your domain.
           switch ($mailfor) {
              case 'mail1': $mailto = 'mail1@yourdomain';
                 break;
              case 'mail2': $mailto = 'mail2@yourdomain';
                 break;
    // Add more case statements here as required.
           }

    // Now send the actual email.
           if (mail($mailto, $topic, $msgprompt, 'From: '.$mailfrom)) {
    // This is a checkbox on the form.  If it's checked, then
    // the end-user has requested a copy of the email.
              if (isset($_POST['copyplease'])) {
    // Set the recipient to the sender so as to get a copy. Note that whilst you
        should provide a From: email address in a valid format, it does not need
        to actually exist.
                 $mailto = $mailfrom;
                 mail($mailto, $topic, $msgprompt, 'From: copy@yourdomain');
              }
    // Redirect to the thank you page.  This is why we needed
    // to buffer the output.
              header('Location: http://yourdomain/thankyou.html');
           }
           else {
              print ('<p>Email could not be sent.&nbsp; Please try again later.</p>');
           }
        }
    }

   // Send button not pressed, so process this block.
    else {
        $mailfor = $_GET['mailfor'];
        $mailfrom = 'valid email address required';
        $topic = $_GET{'topic'};
        $msgprompt = $_GET['msgprompt'];
    }
    ob_end_flush();  // Allow buffered output to go.
?>
```

The form definition:

```
<!-- In the form definition that follows, note how the value of each of the input
    fields is set equal to a mini piece of php code.  This is what gives you the
    'sticky data'. Note that the input field for the recipient is readonly. This
    must not be changed as the content of this field is used by the case statements
    in the php code. -->

        <form id='mailform' name='mailform' action ='' method='post' onsubmit='return(
validateMail())'>
            <fieldset>
                <legend>Email header information</legend>
                <label for='recipient'>mail to:</label>
            <input id='recipient' name='recipient' readonly='readonly' type='text' value=<?php
print ("'".$mailfor) ?>'  /><br />
                <label for='sender'>mail from:</label>
                <input id='sender' name='sender' type='text' value=<?php print ("'".$mailfrom)
?>' onfocus='this.select()' onblur='fieldBlur(this, "valid email address required")' /><br />
                <label for='subject'>subject:</label>
        <input id='subject' name='subject' type='text' value=<?php print ("'".$topic) ?>'
onfocus='this.select()' onblur='fieldBlur(this, "please enter a subject")' /><br />
                <label id='copylabel' for='copyplease'>if you would like a copy of this email,
please check this box:</label>
                <input id='copyplease' name='copyplease' type='checkbox' /><br />
            </fieldset>
            <textarea id='mailmsg' name='mailmsg' onfocus='this.select()'
onblur='fieldBlur(this, "enter your message here")' ><?php print ($msgprompt) ?></textarea><br
/>
            <input class='button' id='sendasemail' name='sendasmail' type="submit" value='send
email' />
        </form>
```

Note that the form definition code contains JavaScript code. This code is available for download with the other code from our web site.

Calling the file and <a> link format
Let's assume that the mail page is called sitemail.html. Let's further assume that you wish to email 'John Smith' on the subject of 'after sales service' and you wish to start the message with 'I would like to make the following observation on your after sales service'.

Then the format for an <a> link would be as follows:

<a href="sitemail.html?mailfor=John Smith&amp;topic=After sales service&amp;msgprompt=I would like to make the following observation on your after sales service">John Smith</a>

Note that the spaces in the above link text should be represented as %20 but they have been left out for clarity.

A strategy for managing your web site mailboxes
This does assume of course that you have access to and the right to set up mailboxes, alter settings, etc.

First things first. Most web site control panels give you the opportunity to select what to do with incorrectly addressed emails – bounce them or set up a 'catchall' email box, which is usually an existing email box.

On the face of it, the 'catchall' facility seems like a useful option because someone may, mistakenly, use an incorrect address or misspell an existing one.

Forget it! This is an open door for 'spammers'. As long as they have a valid domain name, anything gets sent to the 'catchall' address. So they pick up your domain name from the Internet and anything - *rubbish@yourdomain, xyz@yourdomain, etc.,* all goes to the 'catchall' address.

The 'bounce' facility usually gives you the opportunity to specify a message, so select 'bounce' with a message something like – 'This address no longer in use. Please make contact via our web site – *website address*'.

Note: this is the web site address such as *www.oursite.com* and not an email address such as *info@oursite.com*.

No genuine email sender will object to this. If they are sure they were correct, they will look at the email address they used and correct as required. Alternatively, they will make contact via the web site as requested. Spamming software, which is automated, will do neither of these things.

A strategy for forum emails
When you join an Internet forum, as part of the registration process, you are required to enter a valid email address. This is used both to validate your registration and for any subsequent email contact you allow as part of your registration.

Again, this assumes that you have a domain name on which you can set up email addresses. The strategy is not applicable to email addresses at Yahoo, Hotmail, etc.

First set up an email address such as – *forums@domainname*. Set up other mailboxes as required, one for each forum, such as:

> sitepoint@*domainname*
> phpfreaks@*domainname*
> etc.

Now, to minimise the account settings in your email client, turn these mailboxes 'off', so they don't hold a copy, and redirect each of them to your *forums@domainname* mailbox.

Your *forums@domainname* email address is never seen by the outside world and therefore by 'spambots'.

If you start to get 'spam' email through this mailbox, you can check the header to see which email address has been obtained / hijacked, and send a very 'well worded' email to the administrator of the site in question.