Defeating Spam Attacks

Using the Tectite FormMail Script to tackle Problem Spam
How do you get Spam Emails

Spam typically results when email addresses become known to spammers. They can find your email by finding it published on your website, by finding it buried in your website HTML code, or by simply tricking you into revealing your email address to other spoof websites that you may visit. In some cases, you can start receiving spam emails after you sign up for newsletters or other offers on legitimate websites who subsequently sell their email list to others.

In most cases, spam emails are just a nuisance and many can just be sent to your junk email folder. However, if you start receiving hundreds each day, it all becomes a little inconvenient.

As a website owner, spam can arrive in your mailbox from web forms published on your site. Web forms are a primary target for most spammers these days. All it needs is a trawl of the Internet looking for web forms that can be filled automatically and sent to you in large numbers. In some cases, the hope is that an auto response email will be returned, which will give the spammer the perfect opportunity of using your server as a spam gateway. Professional spammers usually employ individuals to test forms for the auto response feature. Once they realise the auto response works, your form will end up on a hit list of must-have web forms.

Using insecure scripts can also cause you a big headache, particularly if your form script allows for the distribution of form responses to many recipients through hidden fields in HTML web forms. This type of spamming doesn’t initially cause you any inconvenience, but when your email service provider starts receiving hundreds of bounce backs to your email account, it won’t be long before you find yourself blacklisted as a spammer.

The first rule of avoiding spam-attacks is to never publish your email address if you can avoid it. If you have to have a publicly viewable email address (maybe because of some stupid law), then use masked email addresses or aliases. These will direct emails to your real email address but without letting the spammers know what it is. If you start getting too much spam email to a masked or alias email address, you can simply delete it and create a new one. There are a number of online services that provide this facility. One free service is MaskMe. This service allows you to create masked email addresses on the fly while you are filling up web forms, or allows you to create them online for more general use. The nice thing about this service is that it allows you to keep a record of the websites where the masked emails have been used. This helps you trace the source of email address distribution. If you don’t want to use a third party service, you should be able to set up email aliases on your website’s own email server. The only downside to either approach is that you may have to delete and create new masked email addresses or aliases from time to time.

Whilst the simple solution above can help keep you protected from direct spam attacks to your email address, it doesn’t effectively deal with the problem of Form spam attacks. This problem is best handled by using a good form script. The remainder of this document explains some of the spam defences built into Tectite’s FormMail script. If you don’t know this script it is recommended that you read the Tectite Script configuration guide on the Tectite website. If you go through the simple instructions to configure the script, you’ll be well on your way to eliminating form spam.
What do the spammers want?

If you understand what spammers are attempting to do, you will have a better understanding of what has to be done to protect yourself. Most professional spammers are trying to distribute website links to unsuspecting recipients, others are trying to distribute malicious code that can track what you do on your computer and send the information back to some diehard crook. A few are just being troublesome. They simply spam you with some piece of garbage just to prove they can do it. The strange thing is, the professional spammers will often use the amateur trouble-maker to test email addresses and web forms, and may even pay such people for every good find they make. Good address and forms are then compiled into lists and sold to other spammers.

Tell-Tale Signs of Spam Messages

Spam messages will usually appear to come from a legitimate email address. In reality, these are often cloaked to disguise the real sender. The idea here is that if the email message is rejected by your server as spam, or if you reply to complain about the unsolicited message, some innocent sap will receive the response. In fully automated, mass spamming, this often leads to an innocent party being inundated with bounced email messages which results in blacklisting. In the majority of cases, the cloaked email address is of most use to spammers who compromise someone’s email account and uses it to directly send out spam messages en-mass. The form filler spammer doesn’t have to worry too much about email cloaking - he or she will simply enter any email address they like into a form field that requires an email address - normally someone’s real email. This is done so that if you commit the cardinal sin of auto responding, particularly if your auto response contains a copy of the original message, the spammer gets a real bonus. You will auto respond to a complete stranger and also send the spammers original message to boot.

The second common characteristic of spam messages is “gibberish” Very often one or more of your form fields will contain garbage such as “hjhjkhh” or “gkjggkjkj” This is a sure sign that someone is testing to see if the message gets rejected. If it doesn’t, they will assume that it’s ok to fill up your form with garbage and it will, most likely, be delivered. Unfortunately, most so-called spam filters don’t recognise gibberish as spam - they tend to only filter common words used in spam emails - this is why gibberish is so popular with spammers.

The Third common characteristic is links. These can be in fields or in the message body itself. Links are what most professional spammers are attempting to distribute so its very likely that almost all spam will contain one or more links somewhere. Spammers have become very clever at getting links into messages. One common method deployed is to send what appears to be a perfectly valid message, by direct email or through a form, advising you that you have been subscribed to some obscure web-based service. It will include an unsubscribe link that when clicked will actually take you to some crummy website that’s being promoted by the spammer. In some cases, the links will actually download and install malicious code on your computer. So, always be mindful of such emails.

A fourth characteristic is attachments. If a form allows users to upload a file, be careful. Files can contain all sorts of nasties. If the attachment comes in an unsolicited email also be wary about opening it.

Some spam messages are just sent out with some sob story designed to get you on the hook for some greater scam or other. These can simply be ignored. Never attempt to reply to the messages - not even to complain about receiving it in the first place. Even if you don’t get pulled into a scam by replying, you will be confirming to someone that your email address is alive and well, and that you are the sort of person that “RESPONDS”. That information can be subsequently passed on to a mass spammer.
A less common, but nonetheless important attack method is where the header of a submitted form or email message may attempt to fool your script or mail server into sending an email with a virus or malware file attachment. Although a physical form filler won't get the opportunity to add header code to your web form, more sophisticated spammers who don't actually visit your site to fill a form use techniques that DO send header information to your script. Whilst this is less common, you should make sure that any script on your server has no such vulnerabilities.

So, how does the FormMail script help you in defeating spam?

Fortunately, the FormMail script has most of it's spam attack detection built in and switched on by default. Some defences require you to make small modifications to the script, but these modifications are well within the capabilities of even novice coders.

The first default setting in the script is the ENABLE_ATTACK_DETECTION setting. The default setting is “true” and should be left that way. What this setting does is to tell the script to use attack detection. If you change the setting to false, attacks will not be spotted and you will still receive junk email. The good news is that even if you continue to receive junk, your script or server will not be compromised. You will be the only person inconvenienced by the junk mail.

The next setting is ATTACK_DETECTION_URL. This isn't set by default, but it is a setting you should include as a simple courtesy to your genuine visitors. Normally, only spam bots trigger attack detection in the script, but sometimes a genuine visitor may inadvertently trigger attack detection by entering the wrong type of data into a form. By setting this parameter with a URL you can forward a user to a page advising them that their form submission looks like server abuse. That page can then give some guidance on how the form should be completed to avoid being classified as server abuse. Don't worry about a real spammer seeing such a message - they are likely to make other errors that continue to trigger attack detection and they will eventually just give up. To activate this setting just edit the script with the the URL of your abuse page as follows:

```php
define("ATTACK_DETECTION_URL","http://www.mysite.com/serverabuse.html");
```

The next setting is ALERT_ON_ATTACK_DETECTION. By default this setting is set to “false”. However, if you wish to be notified of every attack, you can set this to “true”. If you leave it at the default value, all detected attacks will be ignored and confined to the trash can of Cyber-space - where they belong.

```php
define("ALERT_ON_ATTACK_DETECTION",false);
```

Next up is the ATTACK_DETECTION_MIME setting. This is set to “true” by default and should remain that way. What this setting does is look for any malicious code that may be injected into the header fields of any emails sent out by the script. Specific fields that are checked include recipients, cc, bcc, and subject.

```php
define("ATTACK_DETECTION_MIME",true);
```
The next setting is the ATTACK_DETECTION_JUNK setting. This is the setting that looks for all that gibberish we spoke about earlier. This is one of the settings that has a number of additional parameters to help you define what is junk by allowing you to set things like the number of consecutive consonants and vowels as well as the actual consonants and vowels that are used to make up gibberish phrases. These extra parameters are used to help you eliminate false triggers if people use specific gibberish phrases in normal communication with you - such as technical terms or some foreign languages. In most cases, the setting in the default setting will be just fine. The only thing you need to do is turn this feature on by setting ATTACK_DETECTION_JUNK from “false” to “true”

```php
define("ATTACK_DETECTION_JUNK",false);
define("ATTACK_DETECTION_JUNK_CONSONANTS","bcdfghjklmnpqrstvwxz");
define("ATTACK_DETECTION_JUNK_VOWELS","aeiouy");
define("ATTACK_DETECTION_JUNK_CONSEC_CONSONANTS",5);
define("ATTACK_DETECTION_JUNK_CONSEC_VOWELS",4);
define("ATTACK_DETECTION_JUNK_TRIGGER",2);
$ATTACK_DETECTION_JUNK_LANG_STRIP = array(
  ....
);
$ATTACK_DETECTION_JUNK_IGNORE_FIELDS = array();
```

Another useful setting is ATTACK_DETECTION_DUPS. This is a field array setting where you can define a list of fields on your forms that should NEVER contain duplicate content. It’s looks like this:

```php
$ATTACK_DETECTION_DUPS = array();
```

Many spam bots will attempt to fill in your form fields with duplicate content which can be the same string of gibberish, the same email address or the same URL. By adding fields to the array in this setting, any duplicate content will be treated as a possible spam attack. Here is the default setting:

```php
$ATTACK_DETECTION_DUPS = array("realname","address1","address2",
  "country","zip","phone","postcode","state","email");
```

If you don’t want this attack detection, remove all the default field names so that the setting looks like the first example above. If your form fields have different names, just change them in the array. Remember, only include fields that are likely to be common on all your forms and don’t include check boxes, radio button fields. Yes or No fields or fields with simple values such as “1”, “2”, etc.

Another good setting is the ATTACK_DETECTION_SPECIALS setting. This checks certain special fields in your form for the presence of email addresses. These special fields are normally HIDDEN fields in your form that are designed to control the script’s behaviour and which would never contain an email address (this is because hidden fields are not completed by real people - only spam bots). This setting is controlled by two other settings. These are ATTACK_DETECTION_SPECIALS_ONLY_EMAIL and ATTACK_DETECTION_SPECIALS_ANY_EMAIL.

The difference between these settings is that the first checks for fields containing ONLY an email address, whilst the second setting checks for the presence of an email address ANYWHERE in the field.
Here is the default setting:

```php
define("ATTACK_DETECTION_SPECIALS",true);

$ATTACK_DETECTION_SPECIALS_ONLY_EMAIL = array("derive_fields","required","mail_options","good_url","bad_url","good_template","bad_template");

$ATTACK_DETECTION_SPECIALS_ANY_EMAIL = array("subject");
```

In normal use, your forms should have at least a good_url and bad_url field to direct users to a success or failure page. However, if you have no hidden fields that need to be checked, you can simply set ATTACK_DETECTION_SPECIALS to “false”

The Next Setting is ATTACK_DETECTION_MANY_URLS. This setting is OFF by default because some web forms specifically allow URL's to be included. If you use zero (0) the default setting, this detection will be switched off altogether. If you set the value to “1” the form will be checked but any URL found in any field will be rejected. A Number higher than “1” will tell the script to accept that number as the maximum number of URL's that can be included in the form's data fields (hidden fields are excluded from this check because they often contain URL's that you've entered)

This is the default value

```php
define("ATTACK_DETECTION_MANY_URLS",0);
```

If you wanted to allow a maximum of two URL's to be submitted, change the default value as follows:

```php
define("ATTACK_DETECTION_MANY_URLS",2);
```

Aligned to the above ATTACK_DETECTION_MANY_URLS is the next setting which is ATTACK_DETECTION_MANY_URLS_FIELDS. It works in much the same way but checks for many URL's in a SINGLE field. It’s not uncommon for a spammer to list a number of URL's in the message area of a form. This setting will reject a message if there is more than a specified number of URL's contained in a single field.

The default value is:

```php
define("ATTACK_DETECTION_MANY_URL_FIELDS",0);
```

If you wish to enable this attack detection, change the setting value to the permitted number of URL's allowed in a single field. A value of 1 will disallow any URL's. Any other value defines the maximum number of URL's that may be included in a single field. E.g:

```php
define("ATTACK_DETECTION_MANY_URL_FIELDS",4);
```

This will allow 4 URL's in any particular field.
**TIP:** I recommend never allowing URL’s to be submitted in a form. Where a URL may need to be included, I advise visitors to enter URL’s without the www or http prefix and without the dot between the domain name and the domain extension. E.g: `yourdomain.com`. This prevents the form being rejected as spam, but does allow the submission of a non-clickable URL.

The URL detection settings above are controlled by a pattern matching setting named `ATTACK_DETECTION_URL_PATTERNS`. In most cases you won’t have to change this setting as its default is normally sufficient to detect most URL’s. However, you may wish to extend the pattern matching to include other top level domain extensions or country specific domain extensions.

The default setting is:

```php
$ATTACK_DETECTION_URL_PATTERNS = array(
    '(^|[^-a-z_.0-9\+])(?<!@)([-a-z0-9]+\.)+(com | org | net | biz | info | name | pro | tel | asia | cat)\b',
    '(^|[^-a-z_.0-9\+])(?<!@)([-a-z0-9]+\.)+(com{0,1}|org | net)\.([a-z][a-z])
);```

So that concludes the basic spam detection features of FormMail. These, in combination with a few other tricks, will ensure a massive reduction in the amount of spam email you get through your forms. It doesn’t need captchas, or zany puzzles, and it doesn’t need answers to questions. By simply enabling the options above, you will effectively kill almost all automated spam, and frustrate the majority of manual form-fillers. For the odd few that do get through, add some environment variables in a hidden field on your forms so that you get to know the sending IP of the spammer. It may be a cloaked IP, but it doesn’t matter. Simply add the IP address to your .htaccess deny list anyway. Sooner or later these manual spammers just give up when they get nothing back or get continued rejection of their form.

Other features of the script that should be used include the AT_MANGLE setting as detailed in the Configuration Guide. This allows you to effectively disguise the recipients hidden field in your form so that it isn’t recognised by spam bots as an email address. The recipients field is required so that your script knows where to send the form results. As such, it can be read by anyone viewing the source code of your form page. However, disguising the `"@"` symbol in the email address makes it more difficult (but not impossible) for a real person to decipher the address. If you don’t want to use the AT_MANGLE setting and you don’t want to display an email address in your recipient’s hidden field, you have another feature that can be deployed in the script. That is the FORM_INI_FILE setting. This setting simply points to a plain text file somewhere in a protected area of your domain (preferably above the root or public folder of the domain). The setting looks like this:

```php
$FORM_INI_FILE = "\\n```

Just add the URL of your text file between the quotes, like this:

```php
$FORM_INI_FILE = "/home/mysite/formmail.ini";
```

This will tell the script to go and get certain information from a text file named `formmail.ini`, such as the recipients email addresses.
The formmail.ini file *(or whatever name you give it)* would contain instructions similar to those shown below:

```ini
[email_addresses]
me = "youremail@yourdomain.com"
```

This is basically an alias that disguises your real email address. The alias is the word “me”. If you now go back to your form’s recipients hidden field, you can remove the email address from the value field and replace it with: me

Now, if a spammer looks at the source code of your form, he won’t be able to see any indication of an email address. However, your script will see the word me and check to see if it’s listed as an alias in your formmail.ini file. When it finds the alias me, it will substitute the recipients with the full email address associated with me and send the message to the specified email address.

It is important that people can’t access your formmail.ini file on your server. This is why I suggest you put it above the root of your domain. If this isn’t possible, you can prevent access to the file by adding the following to your domain’s .htaccess file:

```html
<Files "formmail.ini">
    Order Deny,Allow
    Deny from all
    Allow from none
</Files>
```

This will prevent anyone *(including you)* from viewing the file. If you ever need to edit the file, just remove the above lines from the .htaccess file, and put the lines back when you’re finished editing.

If your hosting company doesn’t allow the use of .htaccess, you will have to use some other method of protecting the ini file. On some servers, you may be able to automatically protect your file by placing a dot (.) in front of the name, such as .formmail.ini. Another option that may help protect your ini file is to use a really obscure name such as [?^df623{85df.ini. If people can’t instantly associate the file name with a particular function, such as formmail, they will probably just give up and find easier pickings elsewhere.

Hopefully, by now you will have a better understanding of the most common spamming techniques, and how the FormMail script helps defeat them. Ultimately, even if you still get a small amount of spam trickling through, you will at least know that your server cannot be compromised through the use of FormMail. Neither will anyone else get spam emails sent out by the script. At the very worst, you may get one or two of the really determined spam messages, but for the few you do get, it’s not such a big deal to toss them in the trash.