ZB BLOCK

PHP SITE PROTECTOR

TABLE OF CONTENTS

DEDICATIONS:

GOD: Gave me the brain and the health to do this project.

Susan: My love, patience of a saint, and understanding as I devote my resouces to this cause for the sake of humanity.

Michele: Like a sister to me, grubstaked the website, made many suggestions from the point-of-view of a non-programmer.

Ivo: Great friend, excellent debugger, and many suggestions.

Spudz: Another good friend. Helpful in debugging Joomla and Wordpress.

The StopForumSpam.com and TeMerc.com forum members for putting up with my inane drivel and constant questions.

You: For helping in the fight.

ZB BLOCK INSTRUCTION MANUAL

FIRST PRINTING / RELEASE
2009

MADE IN WYOMING,
THE UNITED STATES OF AMERICA

10 9 8 7 6 5 4 3 2 1

Why ZB Block?

There are a myriad of reasons for it, but the genesis was that I, a computerist of 25+ years, was spammed with no regard to my past. At one time, I was partially famous and ran with some of the scene groups in the '90s. These lowlifes just flew through my personal forum, and bombed me with spam.

It was war from that point on. ZB Block is my vendetta against those who have taken the fun out of the net in the name of ill-gotten profit.

Taking lessons from my past, and investigating their methods, I discerned that they were also trying to hack and deface my site. Hint: Don't play h4xx0r with a hacker. I also saw that the problems...

• Were coming from dynamic IP ranges that shouldn't be blocked outright.
• Were not always caused intentionally.
• Could not all be stopped by an .htaccess file.
• Were a mix of harmless but annoying spamming attempts, and harmful hacks.
• Some of the hacks were against my own PHP software, against software I was not using, against PHP itself, and some against MySQL.

Also, I remembered how restricted people were on some hosts, even those with PHP scripting, where the hosts would not let them use their own .htaccess files, and any thought of added server side security “plug-ins” were out. Most system administrators of mega-hosts are too afraid of exposing flaws to discus security with anyone outside the company. If you don't tell someone you have a problem, chances are you will never get it fixed. I have noticed, sadly, that like unto the scenario where some people think that because they own a computer, they have a right to use it competently ...

That some sysadmins think because they can get a server running, that they have done it competently, and no one can tell them differently.

Also, anything that was like my idea was using MySQL, and a few of the sites I checked, were already hacked, most likely because they exposed MySQL to the insecure environment, or worse, was using internal MySQL commands to search for the attack signature.

The sites that were still alive, invariably were selling their tools, one even claiming that their pay-2-play closed source solution was still a GPL license (that one is now dead). Also I couldn't afford a solution at that time (and still can't). Besides, I don't use IIS or Apache, so that sent most of the other software to the graveyard.

Thus the need for ZB Block was self-evident. Something people could use with no more permission than they already had on their host, work with generic PHP, use with any other PHP software, and would still effectively stop spam and hacks.

So, one may ask, “Just what is ZB Block *good for*?”

Saving money and bandwidth. Oddly enough, when one doesn't allow hostiles to access their site, the amount of wasted bytes decreases dramatically. ZB Block takes it one step further, and after 3 strikes from an IP, switches from a ~1Kbyte descriptive “403 Forbidden” to a much shorter “503 Service Unavailable”.

• Stopping MySQL/msSQL tampering / blind injection.
• Stopping Site Defacement.
• Stopping PHP script exploits.
• Stopping Remote File Include (RFI) exploits.
• Stopping directory traversal attacks. (.../../../../../../../../../proc/self/environ%00 type stuff)
• Blocking known bad IP addresses, and hostnames from accessing your site. Sure that can be done by .htaccess, but .htaccess is incredibly slow, and processor intensive, especially when checking hostnames, as it has to re-poll the server variable every time it wants to do a check.
• Stopping known bad user agents (many skript kiddies can't get around this).
• Deterring content theft and keyword scraping. Many of the functions above will work in concert with this goal.
• Stopping registrations and logins from known insecure networks and known spammers (via remote DB lookup), but allow non-registered read only access (TOR and known spammers).
• Stopping HTTP_REFERER Spam.
• Impeding some Cross Site Scripting (XSS) attacks.
• Stopping most forum spam (via remote DB lookup).

ZB Block however will not...

Protect non-php pages.

Sorry ASP sites, but unless your server also runs PHP, you are out in the cold.

Stop access to non-exploitable resource files like .jpg, .gif, .swf etc.

In addition, ZB Block is incredibly fast for a security script. Due to avoiding usually overloaded and exploitable MySQL servers, it's execution time is usually under 100 ms, and that is on an old PIII 930.

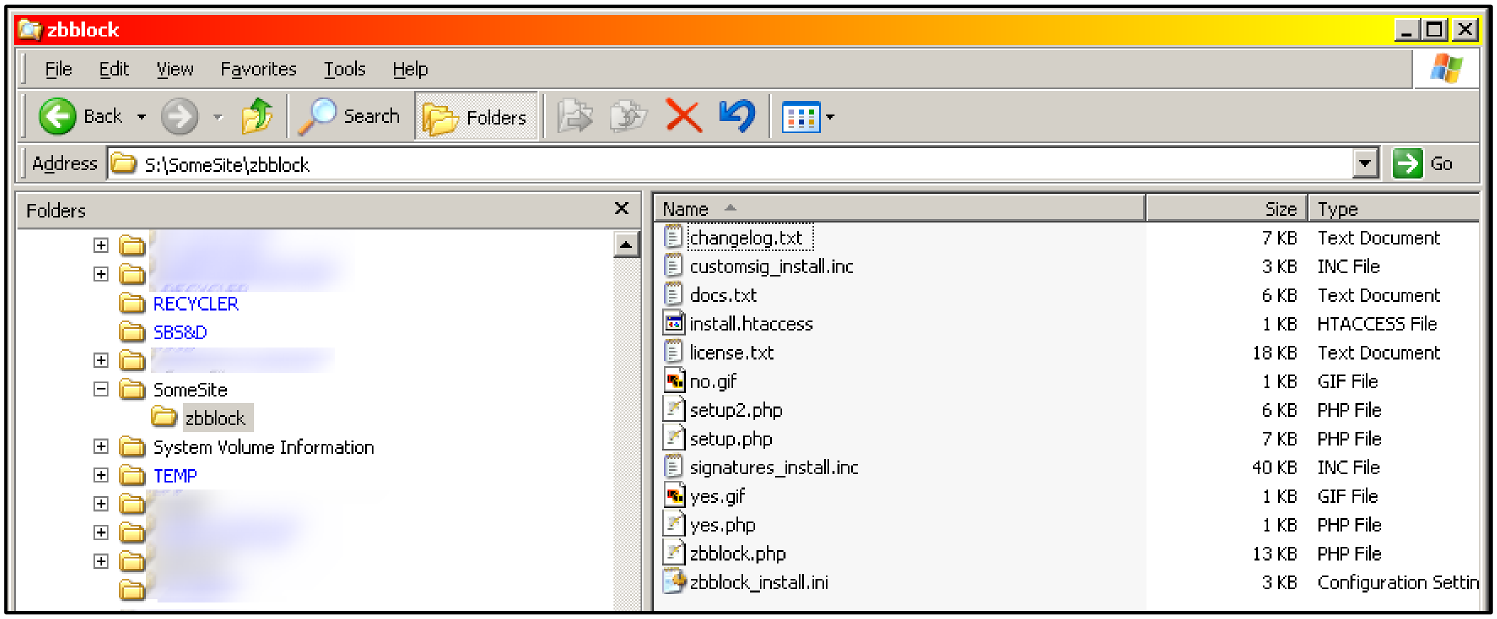
Also, here's some pointers on why ZB Block is ***BETTER*** than .htaccess security methods...

- Under certiain tasks, it is ***FASTER*** than htaccess due to only polling the server for data once per execution. An example of this is domain blocking.
- ***Even faster*** than that because it does not check files that have no need of security, like images, sounds, and embeddable elements.
- And the ***FASTEST***, because in a multiple file-load setting, such as phpBB, it only checks the accessed file for exploit attempts. Any sub-files called for page generation are intelligently skipped over, even if ZB Block is pointed to on that page.
- It will run on webservers that do not support the full gamut of .htaccess commands (And there are quite a few).
- It allows for intelligent detection of problem clients without previous knowledge of their address.
- It can sniff query strings to find attack sequences from all IPs, while allowing legitimate requests to go through.
- Through proper signature use, it can automatically remove some blocks that have met a condition. (such as registration of domain).
- It can ban whole whole ranges of IPs written in classic decimal quadot notation. You can put your own custom ones in the signatures like 193.189.126.5 through 193.189.127.252 . (.htaccess gets a big FAIL! on dealing with IPs as it uses tricky to maintain CIDR ranges that only work in a most signifigant bit (MSB) method, sometimes requiring multiple entries for oddball ranges. 'Did I really include all the IPs? Did I accidentally go to far?')
- Some hosts don't like custom 403s, so they don't allow you to use your own .htaccess. ZB Block doesn't care if the .htaccess is emplaced. ZB Block takes this regained capability even further, and on a certain amount of bad accesses (set in it's own .ini file), switches to a 503 permanent ban.
- It logs banned accesses for later review in plain, easy to read english, with a description as to why said session was blocked.
- It has a built in cache system to automatically ban bad IPs, thus decreasing load on your machine, and the remote databases it works with.
- It's **simple and easy to use**, and requires no authorization beyond the ability to upload files to your php equipped web-server. Despite all the power you have read about before in this, manual, it still really is a no-brainer to use.
- Most importantly, it slows down evil robot machines to a crawl (sometimes) and helps alleviate (we hope) your fellow hosts/webmasters from some of the unwanted traffic!

Installation is a SNAP with ZB Block.

1. Download the .zip, .7z, or .tar.gz.

2. Extract to a local drive so it looks like...


zbblock
File Edit View Favorites Tools Help
Back Search Folders
Address S:\SomeSite\zbblock
Go
Folders
RECYCLER
SBS&D
SomeSite
zbblock
System Volume Information
TEMP

Name	Size	Type
changelog.txt	7 KB	Text Document
customsig_install.inc	3 KB	INC File
docs.txt	6 KB	Text Document
install.htaccess	1 KB	HTACCESS File
license.txt	18 KB	Text Document
no.gif	1 KB	GIF File
setup2.php	6 KB	PHP File
setup.php	7 KB	PHP File
signatures_install.inc	40 KB	INC File
yes.gif	1 KB	GIF File
yes.php	1 KB	PHP File
zbblock.php	13 KB	PHP File
zbblock_install.ini	3 KB	Configuration Settin

(your install's actual files may vary depending on version)

3. Now, upload the \zbblock\ (or /zbblock/ if you use Linux) folder, in total, to the root of your web server.

4. go to http://<yoursite.tld>/zbblock/setup.php, where <yoursite.tld> is whatever your website's name is. *If you see a page that starts with:*

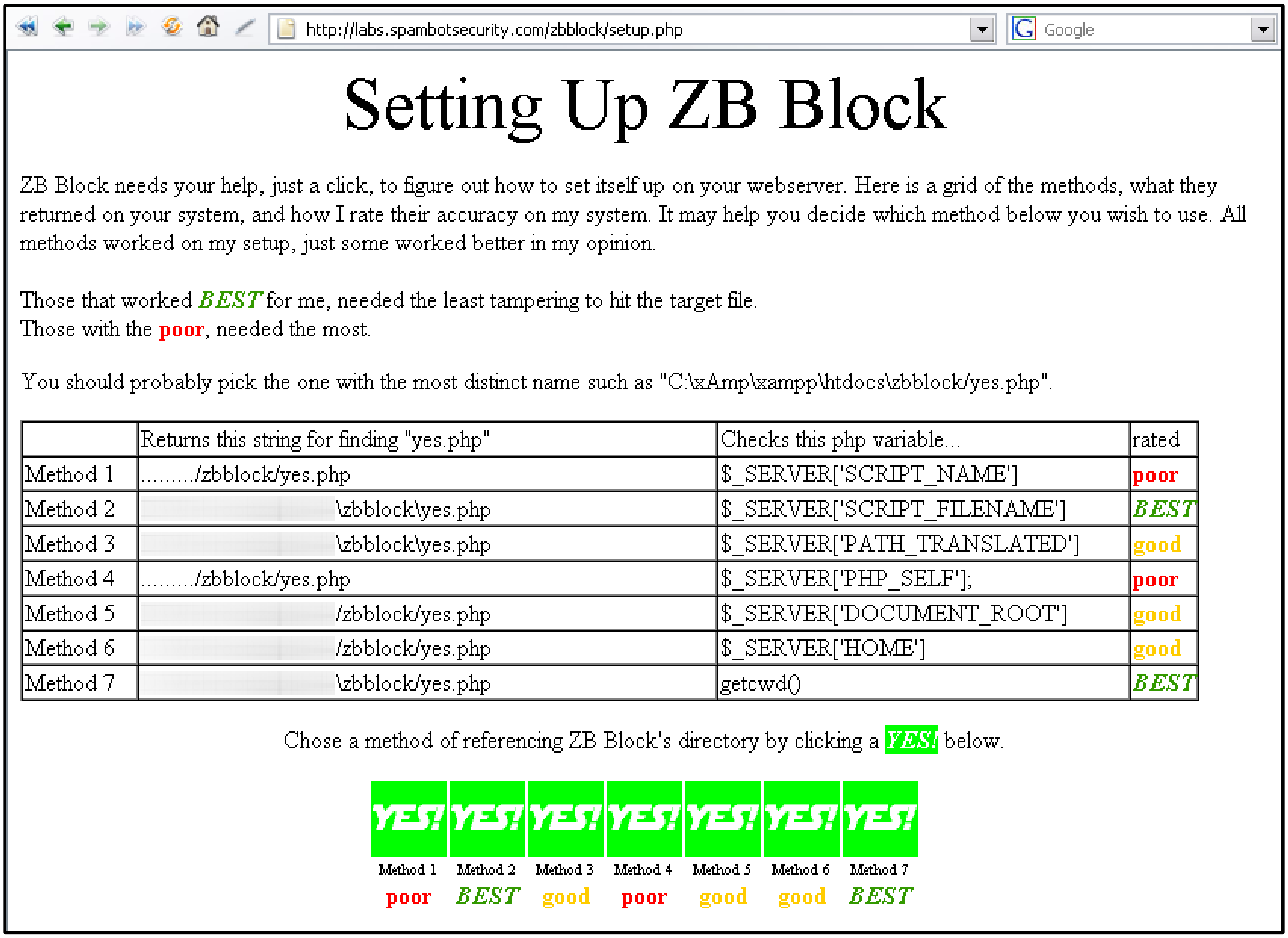
```
<?php
error_reporting(0);
```

Or if it just presents a blank page...

STOP! *You cannot use ZB Block on your site*. You must have the system administrator of your host install PHP or forgo using it. Delete the zbblock folder and it's contents from your site, for now. Hope to see you in the future.

If that instance above didn't happen, please go to the next page.

5. Assuming everything is working right, you should have got a page like this...



http://labs.spambotsecurity.com/zbblock/setup.php

Setting Up ZB Block

ZB Block needs your help, just a click, to figure out how to set itself up on your webserver. Here is a grid of the methods, what they returned on your system, and how I rate their accuracy on my system. It may help you decide which method below you wish to use. All methods worked on my setup, just some worked better in my opinion.

Those that worked ***BEST*** for me, needed the least tampering to hit the target file.
Those with the **poor**, needed the most.

You should probably pick the one with the most distinct name such as "C:\xAmp\xampp\htdocs\zbblock/yes.php".

	Returns this string for finding "yes.php"	Checks this php variable...	rated
Method 1	........./zbblock/yes.php	$_SERVER['SCRIPT_NAME']	**poor**
Method 2	\zbblock\yes.php	$_SERVER['SCRIPT_FILENAME']	***BEST***
Method 3	\zbblock\yes.php	$_SERVER['PATH_TRANSLATED']	**good**
Method 4	........./zbblock/yes.php	$_SERVER['PHP_SELF'];	**poor**
Method 5	/zbblock/yes.php	$_SERVER['DOCUMENT_ROOT']	**good**
Method 6	/zbblock/yes.php	$_SERVER['HOME']	**good**
Method 7	\zbblock/yes.php	getcwd()	***BEST***

Chose a method of referencing ZB Block's directory by clicking a ***YES!*** below.

YES!	YES!	YES!	YES!	YES!	YES!	YES!
Method 1	Method 2	Method 3	Method 4	Method 5	Method 6	Method 7
poor	***BEST***	**good**	**poor**	**good**	**good**	***BEST***

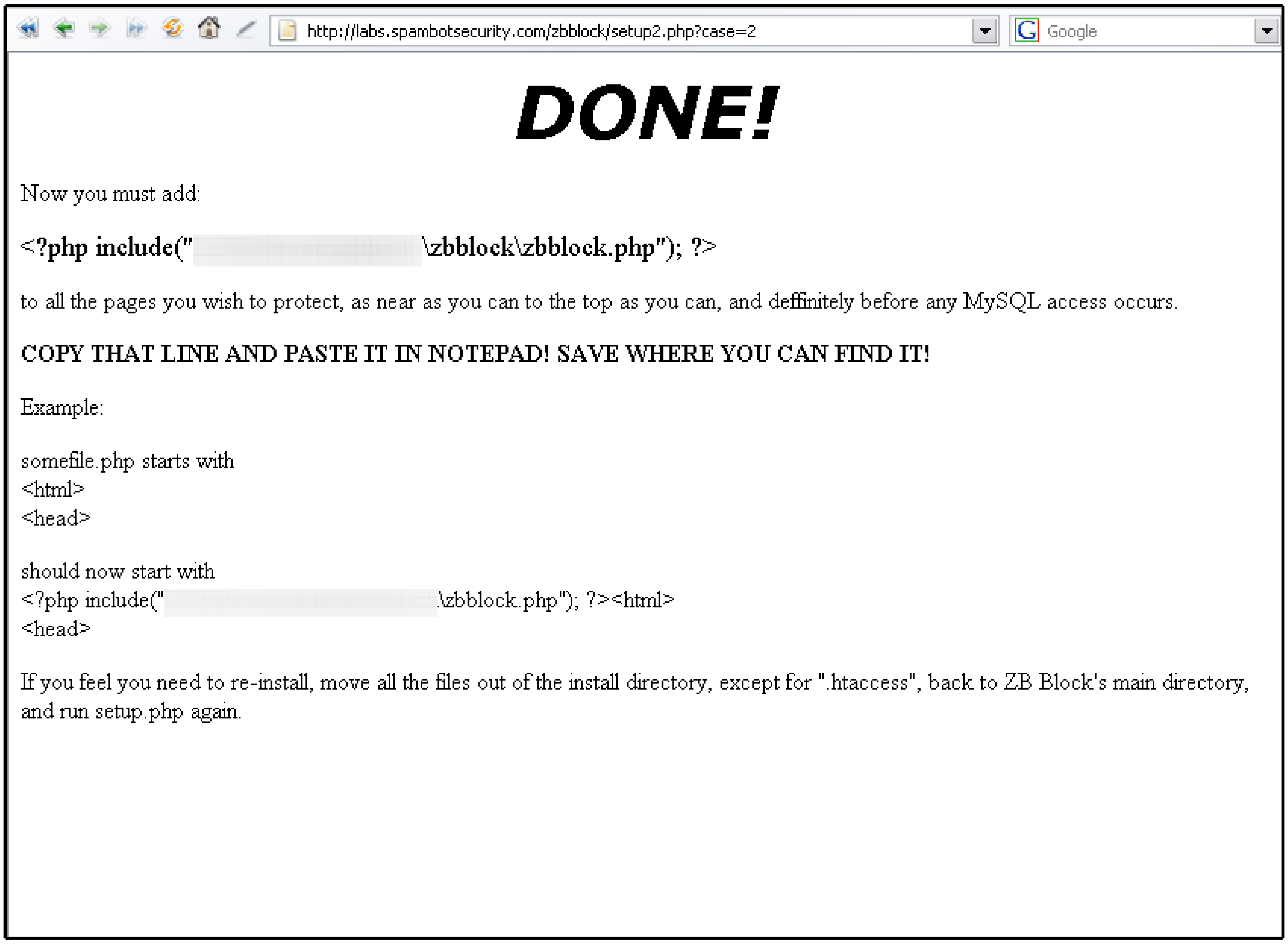
6. If so, this page is probably the most important page in setting up ZB Block on your site. It teaches ZB Block how to find itself on your machine.

Method 2 and 7 are the best in my opinion. 2 would be best for windows servers, and 7 for Linux servers. But any method that has a green "***YES!***" SHOULD work.

Also, not all servers will show green boxes for every method. This is not an error, it simply means your host doesn't allow addressing of a file by that method. This could be due to some crazy idea the sysadmin has regarding security, or just because the OS of the server doesn't support it.

Now click the ***YES!*** you have chosen, and go onto the next page.

7. Okay, the next, and last page that should pop-up should look kind-of like this one...



DONE!

Now you must add:

<?php include("[redacted]\zbblock\zbblock.php"); ?>

to all the pages you wish to protect, as near as you can to the top as you can, and deffinitely before any MySQL access occurs.

COPY THAT LINE AND PASTE IT IN NOTEPAD! SAVE WHERE YOU CAN FIND IT!

Example:

somefile.php starts with
<html>
<head>

should now start with
<?php include("[redacted]\zbblock.php"); ?><html>
<head>

If you feel you need to re-install, move all the files out of the install directory, except for ".htaccess", back to ZB Block's main directory, and run setup.php again.

(Please note, the greyed out areas are to protect my own directory structure. Your page will have actual data therein.)

8. The 3rd line down on this page, the one that starts "<?php include" is very *very* ***very*** important. Copy this line and paste it into a document or two. Print it out and put it in your wallet, stick it to the backside of your monitor, just make sure you don't lose it!

This little snippet of code, is how you integrate ZB Block into your PHP pages. So without it, the script is worthless. It will vary on every server too, so don't panic, expectations are often off on this one. (This is why this "aiming" part of the program was so important to have!)

The final stage in getting your ZB Block working with your php scripts is on the next page.

9. Now you must add that line at the beginning of the files you wish to protect. With phpBB, only one file, common.php, need have the protection put in it. But here's how you do it for ANY php file.

For the sake of instruction, a non functional include element will be used.

If your file starts off with “<?php” on the very first line, then the proper way, to use ZB Block is to add the include segment like this...

```
<?php include('yourdirectory/zbblock/zbblock.php'); ?><?php
```

The “<?php” on the end is the original one moved over. OR you may be able to do it this way...

```
<?php include('yourdirectory/zbblock/zbblock.php');
```

In this case “include('yourdirectory/zbblock/zbblock.php');” was added just after the original “<?php” before any other code.

These three examples **will not work**!

1.
```
<?php include('yourdirectory/zbblock/zbblock.php'); ?>
<?php
```

2.
```
<?php include('yourdirectory/zbblock/zbblock.php'); ?> <?php
```

In these examples, while ZB Block will execute right, if the next script wishes to set a header it will cause an error because of the new-line/carriage-return OR &nbsp; character already being in the output buffer of the server. Try to keep things on the same line, without spaces.

3.
```
<?php
//original page script
//would be here
?><?php include('yourdirectory/zbblock/zbblock.php'); ?>
```

In this example, the script executes before ZB Block, and thus is not protected. Worse, ZB Block will now throw an error because if it tries to set a header, text has already been sent to the output buffer.

If your page starts with HTML like...

```
<html>
<head>
```

Or perhaps even a <doctype> statement, then the proper place for ZB Block, is on the first line like...

```
<?php include('yourdirectory/zbblock/zbblock.php'); ?><html>
<head>
```

Restating here again, that there should be NO spaces, and NO newlines where ZB Block is added.

These will not work...

```
<?php include('yourdirectory/zbblock/zbblock.php');  html>
<head>
```

This is just bad syntax and may even error the browser.

```
<html><?php include('yourdirectory/zbblock/zbblock.php'); ?>
<head>
```

This will cause an error if ZB Block tries to throw it's own 403 or 503 error, as bytes have already been sent to the output buffer.

Once again, if ZB Block exits without detection, no bytes will be added before "<!DOCTYPE" and your page will be perfect when viewed remotely.

Oh, just in the case you didn't understand, ***ZB Block has to be on the first line of the source. No blank lines above it.*** (Some people have missed this).

Also, if the page is something.htm or something.html, you will have to rename it (and re-aim your links) to something.php for ZB Block to work. As of now, there is no safe way to use a rewrite rule to attach ZB Block to other filetypes.

CONGRATUALTIONS! YOU NOW HAVE ZB BLOCK UP, AND RUNNING, AND DOING IT'S JOB. But you should read the next 2 sections, especially the updating one, to remain safe.

CONFIGURING & TESTING ZB BLOCK

**(Also Known As The .ini File Explained and
How to Keep Yourself From Being Locked Out of Your Own Site)**

Now that you have protected your site, you may wish to take advantage of some of the advanced features of ZB Block. This is done through the .ini file and a few command line arguments.

As installed, ZB Block will work fine for most people, but a lock-out condition could happen if you trigger the warning more than 3 times in 1 day. By reading on, you will avoid this problem, and know for sure that ZB Block is operating fine.

First, download and open your zbblock.ini in notepad or any text editor. If you are nervous, just make a backup you can put back in place if from the /zbblock/vault/ directory of your site. You cannot see this file directly on the site, as vault/ is now a protected directory, and set up so during install. DO NOT REMOVE THE PROTECTING .htaccess FROM THIS DIRECTORY!

Each section of the .ini file is built like this...

```
; *** TITLE OF SETTING***
;
; Description of what
; the setting does
;
; values:
; Acceptable Values
;
; For The Setting
;
; default: Original Setting Upon Install

some_variable = some value or “some value”


; *** NEXT TITLE OF SETTING***
```

THIS IS THE ONLY LINE YOU WILL NEED TO CHANGE IN ANY SETTING.

Pretty simple huh?

Over the next pages we will discuss what each setting means, and what to do with it.

The first, and most important setting, is the password. While ZB Block may function fine forever without it, it is best you set it. This is how it appears on installation.

```
; *** ZB Block Password ***
;
; Password to control functions of ZB Block
; ?wlpw=<password> to add yourself to the whitelist
; and allow yourself back in.
;
; values:
; "" to neutralize password and turn of control
;    functions globally.
;
; "<password>" Password to control functions.
;
; default: zbb_pw = ""

zbb_pw = ""
```

You will want to pick a master password for ZB Block, this will be used to make sure you cannot get locked out of the system, and in the future, to access the control panel (yet to be developed). Set it LIKE this, but not with the example password.

```
zbb_pw = "ThisIsMyPassw0rd"
```

I recommend you set it to something at least 10 characters long, mix upper and lower case in it, and throw in some numbers. I recommend you do not (unless you know what you are doing) use punctuation, as this can confuse PHP.

And as with any password, try to **AVOID** pet names, family names, address number, birthdays, social security numbers, anything that a hacker would guess.

Now save and upload the .ini to the server. Why? We have to exclude you from being permanently banned.

Now that it's uploaded, we need to send a command to ZB Block. Enter this...

```
http://<yourwebsite.tld>/zbblock/zbblock.php?wlpw=<the password you just set>
```

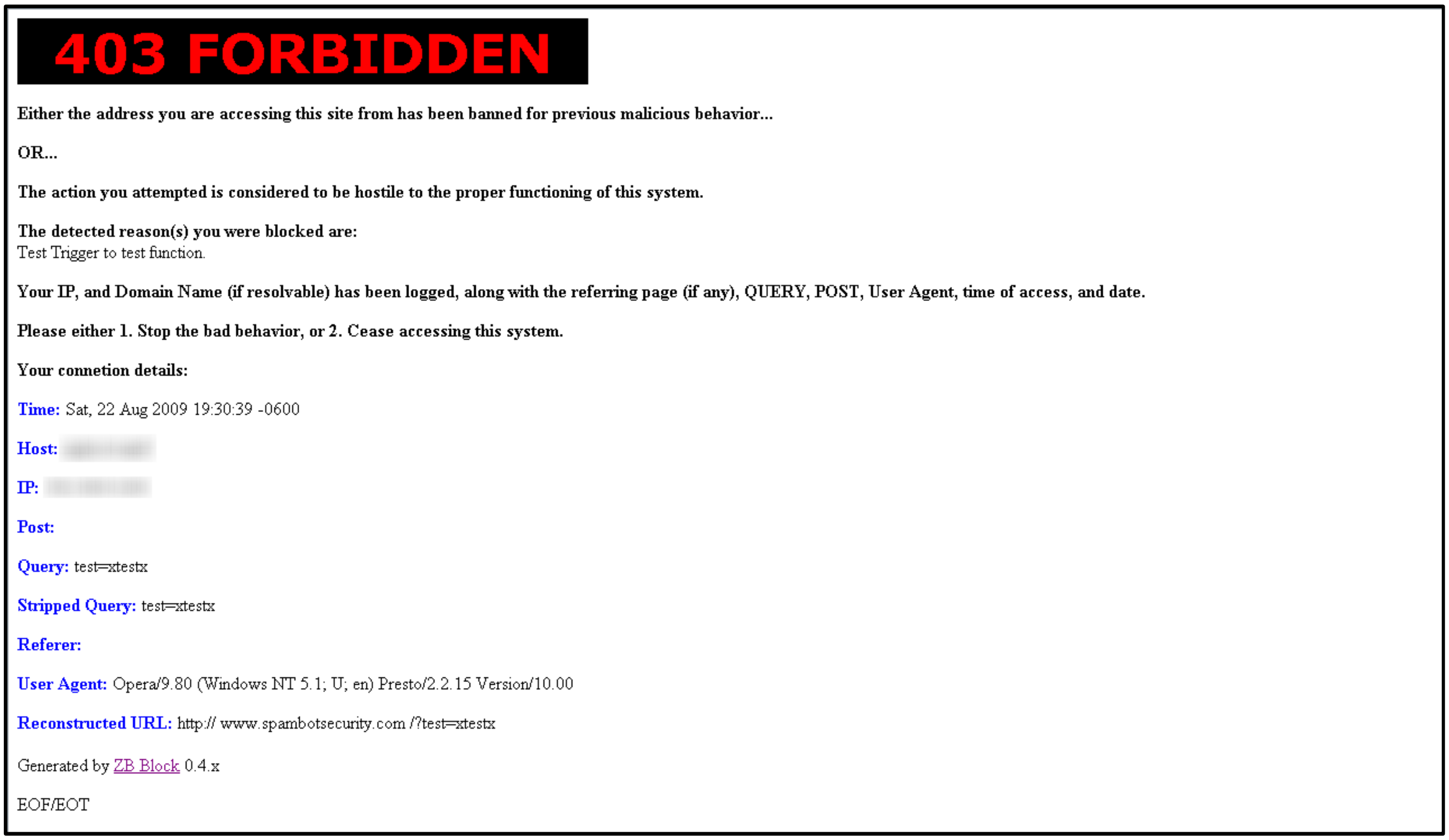
Wait 60 or so seconds.

When it finishes loading (a horrible long wait to keep people from brute-forcing the password), you are ready to give ZB Block a little test (since you cannot now be banned from your own site.)

The test is really simple...

Go to any protected page, and add ?test=xtestx to the line.

You should get a page that looks like this...



403 FORBIDDEN

Either the address you are accessing this site from has been banned for previous malicious behavior...

OR...

The action you attempted is considered to be hostile to the proper functioning of this system.

The detected reason(s) you were blocked are:
Test Trigger to test function.

Your IP, and Domain Name (if resolvable) has been logged, along with the referring page (if any), QUERY, POST, User Agent, time of access, and date.

Please either 1. Stop the bad behavior, or 2. Cease accessing this system.

Your connetion details:

Time: Sat, 22 Aug 2009 19:30:39 -0600

Host: [redacted]

IP: [redacted]

Post:

Query: test=xtestx

Stripped Query: test=xtestx

Referer:

User Agent: Opera/9.80 (Windows NT 5.1; U; en) Presto/2.2.15 Version/10.00

Reconstructed URL: http:// www.spambotsecurity.com /?test=xtestx

Generated by ZB Block 0.4.x

EOF/EOT

If that is what you saw on the test, then without a doubt, ZB Block is set up correctly, and working on your system.

Now, since you have already managed one setting in the .ini file, I can quickly run through the rest of the settings, and reasonings behind them without the excess descriptions.

debug_mode:

This setting engages a special debug mode used for finding problems in the signatures versus the proper functioning of your board. It should not be left on as this would give a boost to hackers looking to end-run ZB Block.

When active it will show the algorithm that found the problem, and what it was looking for, along with the normally output error.

timer_trap:

Defaults to 0, but this setting invokes an x second delay on registration or login on your site. This is good to avoid brute-force hammering. Has been somewhat obsoleted by the new automatic defense functions of ZB Block, primarily the automatic permanent ban, but is still here if you decide not to use it.

block_hph_registration:

This checks the blocklist on http://hosts-file.net to see if a registrant, or person logging in is known as a spammer. If so, turning this on (the default is on), will block them from registering or logging in.

block_sfs_registration:

Same as above, but checks the http://www.stopforumspam.com API. Default is on.

sfs_date:

Works with above setting to set a maximum age limit on the accepted data from stopforumspam.com . 0 would be for this year only, 1 would work for this year, and last year, and 99 will work for the last 99 years (virtually forever). Defaults to 10 years.

block_tor_registration:

Same as the other blocks above, but checks with http://www.torproject.org to see if the person is using the TOR anonymity network. Since registrations should have only verifiable data, this should not be allowed. Default is on.

block_tor:

Checks with torproject.org to see if the person is using TOR, and blocks their access site-wide if so.

ZB Block was never intended to be a censor of information, so this setting is defaulted to OFF. This will still allow TOR users to read your forums, and what have not, but not to register or log-in if the preceding block is on.

snooze:

This is the amount of time ZB Block forces the violator to wait for a status output. Can be very helpful in slowing down robots.

You want to balance it's value so it doesn't exceed the timeout value for the hostile, but still makes them wait. Default wait is 25 seconds.

log:

This determines whether the hits are logged to a killed_log.txt file. Default setting is on, as this can be very helpful in debugging why a certain someone was blocked, or if there is an incompatibility between ZB Block and your software.

If the log is getting too big on your site, simply delete it. ZB Block will start a new one.

log_dir:

Where to store the killed_log.txt. Defaults to protected vault/ directory. However, if you want to put your trophies on the wall and have skript kiddies hunting each-others scripts and compromised sites, you can change to “”, thus putting it in the unprotected ZB Block directory, where you can include them in your website.

csv:
AND
csv_dir:

Same as above, but generates a .csv for importation into a database for analysis. Usually only for developers and statisticians. Defaults to off, and vault/ .

rdb_timeout:

Controls how long ZB Block will wait for a host of a remote database to respond before giving up, and moving on. Default is 10 seconds, may be increased in the future.

fault_count:

Controls how many 403s an attacker is presented, before they are permanently banned from the system, and served the processor, and bandwidth saving “503: Service Temporarily Unavailable” The default value is 3.

UPDATING ZB BLOCK

Even the best software always leaves something unchecked, undone, or the previously unknown, has been made known to the author. ZB Block was written with this in mind, thus updating the signature file, and even the main script is easy.

To update the signature file (signatures.inc):

Simply download a fresh signatures.inc from:

http://www.spambotsecurity.com/zbblock_download.php

Unzip on your local hard drive, and then upload the .inc to your server, overwriting the old one. And you are done.

This should be done especially if there is on the site a new signature file, that has a later date than the last main script update.

Speaking of that...

To update the main script:

Either download a fresh installer, and install over your old version, following all the installation steps, or...

If your version is recent enough, download the incremental installer package, and just overwrite the files it contains over the old ones on your server.

As before, you should get your files from:

http://www.spambotsecurity.com/zbblock_download.php

Told you that it was easy!

USAGE

How to write your own signatures for the customsig.inc file:

The first thing you have to do is learn which variables contain what data in ZB Block.

Variables used in ZB Block:

Here's a quick rundown on them.

$address :	The IP address of the visitor.
$hoster :	The IP address derived hostname of the visitor (reverse lookup).
$fromhost :	The lowercased HTTP_REFERER of the visitor.
$query :	The lower-cased query string sent to the server (everything after the first ?)
$querydec :	The URL decoded version of the above.
$useragent :	The raw user-agent string sent by the visitor.
$lcuseragent :	The lowercased version of above (better for detections).
$thishost :	What site was called? Good for sniffing out odd 3rd level hostname attempts.
$requesturi :	The full request string sent.
$lcrequesturi :	The lowercased version of above (better for detections).
$pathinfo :	True if they tried to path your file as in: http://<yoursite.tld>/somefile.php//someotherfile.php?somevar=hack
$rawpost :	Raw post data sent by the host.
$whyblockout :	Error messages to be sent to user. Also useful for seeing if an error flag of a certain type has already been thrown.
$ax:	The “score” of the connection. Any positive value will trigger a detection and blockage of the visitor. Can be subtracted from to remove some transgressions.

All this is good data, but not much help in writing custom signatures without the descriptions of the search algorithms on the next page...

Description of search algorithms:

lmatch: Check for a match at the left side of the variable only.

Format is:

$ax = $ax + (lmatch([variable to be searched],"[pattern]","[what is the reason for the check]"));

Will check the left side of a variable for pattern. Good for IP and third level domain checks.

If pattern is “109.120” then...

“109.120.225.76” will be true.

“111.109.120.76” will be false.

If pattern is “mail” then...

“mail.somedomain.com” will be true.

“www.mailmaster.com” will be false.

rmatch: Check for a match at the right side of the variable only. (Good for hostnames)

Format is:

$ax = $ax + (rmatch([variable to be searched],"[pattern]","[what is the reason for the check]"));

Will check the right side of a variable for pattern. Good for domain name checks.

If pattern is “party.com” then...

“badgirl.sexparty.com” will be true.

“grandoldparty.com” will be true.

“partyamerica.com” will be false.

inmatch: Check for a match anywhere in the variable to be searched.

Format is:

```
$ax = $ax + (inmatch([variable to be searched],"[pattern]","[what is the reason for the check]"));
```

Will check anywhere in the variable for the pattern. The heaviest used detection.

If pattern is “badhack” then...

“time=now&page=badhack&youre=pwned” is true.

“time=then&page=bad_hack&youre=pwned” is false.

“badhack=badhack&thisisnt=funny&page=badhack” is true.

minmatch: Check for a match anywhere in the variable to be searched, and see if it occurs more often than the limit.

Format is:

```
$ax = $ax + (minmatch([variable to be searched],"[pattern]",[decimal allowed occurrence limit],"[what is the reason for the check]"));
```

Will check for pattern matches anywhere in variable, but will only trigger if the allowed occurrence is exceeded.

If pattern is “badhack” and allowed occurrence is 2 then...

“time=now&page=badhack&youre=pwned” is false.

“time=then&page=bad_hack&youre=pwned” is false.

“badhack=badhack&thisisnt=funny&page=badhack” is true.

Iprange: IP range match.

Format:

```
$ax = $ax + (iprange($address,"[low ip]","[high ip]","[host/reason]"));
```

[low ip] : Classic quadot (www.xxx.yyy.zzz) beginning of range IP address.

[high ip] : Classic quadot (www.xxx.yyy.zzz) end of range IP address.

Used to determine if visitor is from a banned range.

If Low IP is 192.168.60.0 and High IP is 192.168.225.255 then...

“192.168.0.1” is false.

“192.168.75.55” is true.

“192.168.075.055” is true.

“203.2.55.91” is false.

Master these 5 simple detection routines, and you can make a pattern for anything. Remember, help is always available at http://www.spambotsecurity.com/forum .

How to turn off problematic signatures:

The simplest way is to go into the signatures.inc file and add a double forward slash in front of the signature you want turned off, like this...

```
$ax = $ax + (inmatch($querydec,"*/","RFI attack/SQL injection"));
```

Becomes

```
//$ax = $ax + (inmatch($querydec,"*/","RFI attack/SQL injection"));
```

But this is an incomplete solution, due to the fact the next signature update from the ZB Block site will restore it. A more permanent solution can be had by copying the erroneous signature to your customsig.inc file and changing the “+” after the second $ax to a “-”. This will effectively do the same thing, but be permanent. The above line so modified would look like this in your customsig.inc:

```
$ax = $ax - (inmatch($querydec,"*/","RFI attack/SQL injection"));
```

APPENDICES

I. Useful Websites:

http://www.spambotsecurity.com The home site of ZB Block.

Research sites:

http://www.stopforumspam.com A great source on forum spammer tactics, and has a manually entered, high confidence blacklist of IPs known to be spammers.

http://www.hosts-file.net Another great source of spammer info, and has the second remote database this program uses.

http://issviews.com/ A good source of computer security info, and a debugger of ZB Block.

http://www.temerc.com/ Another good source (in the forums) for info.

http://dnsbl.tornevall.org A DNSBL that may be integrated into ZB Block soon.

http://www.robtex.com Great site to research a suspicious IP or hostname. Also has links to many of the blacklists if an IP is listed on them.

http://www.senderbase.org A good site for detecting bad addresses, and visualizing ranges of bad addresses.

http://www.trustedsource.org A good site to see the disposition of address and their network as far as spam and hostility goes.

http://www.projecthoneypot.org A great multi-mode blacklist.

http://www.uceprotect.net/en/ Another multi-mode blacklist.

http://www.botsvsbrowsers.com A good place to find history of user-agents emanating from a particular IP.

http://www.google.com/ Just Google a suspicious IP or address in quotes and you are bound to find something.

Other sites:

http://zaphodb777.dyndns.org The author's personal site.

http://www.michelespaintshop.com The author's friend's site. Great graphic design tutoring and ideas.

http://wasteland-bg.com (Bulgarian) The diabolic one's website, a great help in debugging ZB Block.

APPENDICES

II. The compatibility.inc File

This file was introduced in version 0.4.5 to allow for easy compatibility with most sites on installation of ZB Block, with an optional “Tightening” of the rules after installation by dropping in a custom compatibility.inc. Also it is, and should remain well documented internally for the sake of hand editing.

The one distributed with ZB Block is in it's weakest allowable form. Weak is a misnomer as ZB Block is still quite strong even with this file in place. But, there are ways to strengthen it. Let's have a look at some of the sections in it (for the sake of hand editing).

Pathing Detection: This section allows you to turn off detection of some hacks that resemble “friendly” page names used by some content management systems. It is normally off to allow these non-standard http: requests (as per RFC 2616)

I would like to take a moment to comment on the fact that no search engine cares about filenames anymore as 1. No_Human_I_Know_Of_Is_Going_To_Remember_A_Page_Name_Like_This,_Nor_Wish_To_Type_It_In_Order_To_Get_Back_To_It. They instead will navigate through your site's menus, use your site's search, or use a search engine to re-find the data they want. 2. All search engines base results much more heavily on content, and minorly on webmaster chosen keywords, and almost not at all on filename, or no one would find poop in a forum.

So this “search engine friendly” file naming scheme is up to you. But since it breaks standards, it also allows hacks which break standards.

Specific Bypass for Wordpress Login: Wordpress uses an internal forwarding command passed in the query of a URL. Incredibly bad form, and just waiting for an exploit. Inactive in default file. Data should be passed to next process via hidden form POST.

Site Calls Itself in Query: Same as above, but all encompassing. This is normally active by default, since I cannot know what ills any one content management system will have. This is the most “leery” of allowances, but still does not defeat ZB Block's security unless the site has already been compromised.

Directory Traversal / Tree Climbing: This detection is against a favorite hack on Linux based Apache servers. It is normally active by default, but with badly written valid scripts popping up all the time, gosh knows if you will have to deactivate it.

Illegal URL Format: Believe it or not, I know of one script that makes/allows URLs that look like http://www.mysite.tld///somefile.php . Ick, usually found in hostile scripts. So, for the sake of that script, this detection is shipped OFF.

Registration / Login Check Bypasses: Sometimes the remote databases get wild hairs in their bums, and mark 192.168.xxx.yyy addresses, or 127.0.0.1 as hostile. OOPS on their part. No biggie. The normally on bypasses here allow for it.

Also, sometimes a friend might be marked as a spammer for whatever reason. You can easily add a bypass here (several parked blanks are ready for you), to allow them to register or log-in.

Please note, this section ONLY affects registrations, confirmations, and logins. Normal protections are not bypassed by it!

Infected Servers: Several things can be done to detect a hostile script attacking. These rules, normally inactive, can allow you to detect sweeping range of php bot attackers. Although they do occasionally kill a friendly user. Use at your own discretion. If you are smart enough to be messing with them, I don't need to tell you anymore.