Catch additional hacked files!

Find the Files

These are all the options you have to find files that may in fact be malicious. These also find non-malicious files at times! Be careful!! You want to run this in the public_html of the account, though:

Find files, processor-unfriendly

```
for i in $(find . -ctime -100 -type f -iname "*.php" ); do if [[ -n `head -n2 $i |
grep -le "eval" -le "strlen" -le "strto" -le "auth_pass" -le "GLOBALS" -le "_dl"` ]] ;
then echo $i; fi ; done
```

This is a more processor-friendly and quick way to do it, and you don't need to public_html it (you always can if you want with a small edit!):

Find files, processor-friendly

```
find /home/*/public_html/ -ctime -100 -type f -iname "*.php" -exec awk 'FNR==1 &&
/eval/ || /strlen/ || /strto/ || /auth_pass/ || /_dl/ || /GLOBALS/  { print FILENAME
}; FNR>1 {nextfile}' {} +
```

If you want you can write the results to a file without a wrapper for loop like so:

```
find /home/*/public_html/ -ctime -100 -type f -iname "*.php" -exec awk 'FNR==1 &&
/eval/ || /strlen/ || /strto/ || /auth_pass/ || /_dl/ || /GLOBALS/  { printf FILENAME
"\n" }; FNR>1 {nextfile}' {} + >> /root/filelist.txt
```

Here's something Dan recommends, finding php files in folders that really shouldn't have php:

Find files that are in js and css folders but are PHP files

```
find `pwd` -type d -name css -exec sh -c  'find {} -name "*.php"' \;
find `pwd` -type d -name js -exec sh -c  'find {} -name "*.php"' \;
```

You can find files that are a specific filesize if you are noticing the same file popping up several times. Replace $NUMBER with the file size in bytes here:

```
for i in $(find /home/ -size $NUMBERc); do echo $i; done
```

You can also find just files that have a specific number of characters within them, which is useful for if you have a lot of files that are all the same size and have random names:

Find files of a specific filename length

```
find /home/*/public_html/ -type f -print| awk -F/ ' length($NF)  == 11 ' | grep .php

Make sure you're excluding everything with grep -ve! There can be a ridiculous number
of files to exclude ...
```

If you have email sending out and can't find where it's sending out from, this is a malicious file. You should be able to find it by doing something stupid, like:

ps -U nobody ; strace individual threads from there. One of them will be the malicious file executing, if you can strace the right one it will point to the file. Or you can just add /mail.ru/ to the first find command. 🙂

The recommended one-liner to find malicious files, if you just want to fire and forget

Personally this is what I would run out of these commands:

```
find /home/*/public_html/ -ctime -100 -type f -iname "*.php" -exec awk 'FNR==1 &&
/eval/ || /strlen/ || /strto/ || /auth_pass/ || /_dl/ || /GLOBALS/  { print FILENAME
}; FNR>1 {nextfile}' {} + ;
find `pwd` -type d -name css -exec sh -c  'find {} -name "*.php"' \;
find `pwd` -type d -name js -exec sh -c  'find {} -name "*.php"' \;
```

On Removing files and Automation

You can run a one-liner to run these and then catch some of the malicious files, something like this for example if you are getting a lot of people with malicious code between the first <?php and second line:

```
for i in $(find /home/*/public_html/ -ctime -100 -type f -iname "*.php" -exec awk
'FNR==1 && /eval/ || /strlen/ || /strto/ || /auth_pass/ || /_dl/ || /GLOBALS/ { print
FILENAME }; FNR>1 {nextfile}' {} +); do echo "Backing up $i";
new_dir="/root/filebackup${i%/*}"; mkdir -p "$new_dir"; cp -n "$i" "$new_dir"; echo
"Fixing $i"; sed -i -r '1 s/(.{5}).*/\1/' $i; done
```

Or, this will only hit clamscan-marked directories (dp more grep -v 's if you don't want it to hit /themes or etc.):

```
for i in $(cat /root/clamscans/scan.last | grep ":" | head -n -8 | cut -d: -f1); do
echo "Backing up $i"; new_dir="/root/filebackup${i%/*}"; mkdir -p "$new_dir"; cp -n
"$i" "$new_dir"; echo "Fixing $i"; sed -i -r '1 s/(.{5}).*/\1/' $i; done
```

You can also make it ignore specific directories if you think those are the directories that are breaking the site when you 'fix' the sites (I've seen this happen every time I test this fix). Something like this could work:

This ignores the themes directory!

```
for i in $(find /home/*/public_html/ -ctime -100 -type f -iname "*.php" -not -path
"/home/*/public_html/wp-content/themes/*" -exec awk 'FNR==1 && /eval/ || /strlen/ ||
/strto/ || /auth_pass/ || /_dl/ { print FILENAME }; FNR>1 {nextfile}' {} +); do echo
"Backing up $i"; new_dir="/root/filebackup${i%/*}"; mkdir -p "$new_dir"; cp -n "$i"
"$new_dir"; echo "Fixing $i"; sed -i -r '1 s/(.{5}).*/\1/' $i; done
```

I would recommend probably running this on one site at a time, though it depends on the extent of the issues. You can fix any errors that have created a broken site by just copying the files back from the filebackup with something like this:

the \ is because all of our cp commands disable overwriting but you explicitly want to overwrite in this case.

```
\cp -R /root/filebackup/home/* /home/
```

If you're looking for a way to fix these all outside of doing it by hand or that command, here are a bunch of different possible ways.

-You can remove it with a sed command.

SimpleSed(tm)

```
sed -i -r '1 s/(.{5}).*/\1/' (file)
```

This grabs the first five letters of a string and removes the rest, for every file you point it at, which is appropriate for php scripts with line 1 injections. There are ways this will fail, if the original code was `<?php somecode(); ?>`, the code and trailing slash are going to be broken.

A lot more files than you'd expect start their script with <?php somecode(); ?>.

-You can sometimes remove it with a vi macro. Here's how you would set that up:

Add what you want to be done to a file, like /root/fixmacro.sh. This removes that first line with the strtolower information and replaces it with a blank <?php (not fully tested, test beforehand if you're using it!):

Add this to a 'fixmicro.sh' file:

```
:1/ua=strtolower/s/^.*<?php$/<?php/
:wq
```

Run this to loop this on all php files:

This is a stupid loop to use with the vi commands in the prior file

```
find (folder) -name *.php -exec vi -s (location of macro script) {} \;
```

You can also do something like this to find anything of a specific size and quarantine it:

Find and move files of a specific size

```
replace 'somesize' with the size in bytes of the file (so if you ll the file the size
will be there):
for i in $(find /home/ -size 'somesize'c); do new_dir="/root/abuse${i%/*}"; mkdir -p
"$new_dir"; mv -v "$i" "$new_dir"; done
```

I just add this bit a lot to the top one-liners to either move or copy files over by just placing it in a for loop (if you want to move files, replace 'cp' with 'mv'):

```
do echo "Backing up $i"; new_dir="/root/filebackup${i%/*}"; mkdir -p "$new_dir"; cp -n
"$i" "$new_dir"; done
```

From there, you can remove the files, or cat them to read through them, etc., as much as you'd like. Or you can do ... whatever.

If you have any additions or questions, just let me know.