CBL Lookup Utility

Automated/scripted bulk lookups are forbidden. Upon detection, automated scripts will be denied access, and the source IP may be listed in the CBL.

Enter an IP address: Lookup

IP Address 1.2.3.4 **is listed** in the CBL. It appears to be infected with a spam sending trojan or proxy.

It was last detected at 2012-02-24 00:00 GMT (+/- 30 minutes), approximately 9 hours, 30 minutes ago.

This IP is sending email in such a way to indicate that it is, or is NATting for a web server that is infected with a spam sending script, like Darkmailer, DirectMailer, r57shell, or some analogous Perl, PHP or CGI script.

If, however, you are running the "sendmagic" mail server package, please contact your vendor for an update.

If you are not running "sendmagic", the following should help you identify where the spam sending script is located so that you can remove it as well as innoculate your web server so it doesn't happen again.

This IP is infected with, or is NATting for a machine that is infected with, a PHP-based backdoor trojan. The most common target of this infestation is Plesk hosting environments using Qmail and Wordpress or CPanel hosting environments.

The WebMaster should pay very close attention to the following:

This infects web hosting environments. ONLY the hosting company can fix these infections properly.

If you are not the administrator of this hosting environment, there is probably nothing you can do to fix this infection, you MUST refer this listing to them. The hosting administrator has to do the fix. We recommend you forward this email to them.

This is a checklist of things the administrator needs to do before delisting.

- Turn off PHP script support within the web server except where absolutely necessary.
- Many of these compromises appear to be uploaded via a vulnerability in older versions of WordPress. Make sure you have the latest version.
- Identify, kill and remove the backdoor trojan scripts currently on the web server (see below).
- Implement port 25 blocking so that only your mail server software userid can make outbound port 25 connections from this machine.

How to find web-based PHP (and other) backdoor trojan scripts.

How to prevent your site from getting hacked. How to repair a damaged site. Website security precautions. and its companion document Website security: How to find backdoor PHP shell scripts on a server have detailed instructions on how to find various kinds of web server trojans and back doors.

There is a simple script at the end of this web page that you can download and run. This script is known to detect at least some of these backdoors.

In one case, two malicious files were involved. One was a malicious .htaccess file that could be found by: (substitute rootdir with the directory of your web hosting directories):

```
find rootdir -name '.htaccess' -print | xargs grep 'RewriteRule'
```

This yielded a line like this naming the malicious PHP script:

```
RewriteRule . /phpinfo.php [L]
```

Alternately, the script itself could be found with the following command:

```
find rootdir -name '*.php' -print | xargs grep 'eval(base64_decode'
```

Which yields a line like this:

```
eval(base64_decode('DQpAZXJ ...
```

Followed by many lines of Base64 encoding.

Securing your server from future infections

There are a variety of things to do.

- Ensure that your web server does not allow clients to modify ".htaccess" files.
- Only permit PHP where necessary.
- You **SHOULD** configure your web server to prevent such infections being able to spam the Internet. Once you've done this correctly, it doesn't matter whether the spammer can still upload this malware, they can't spam with it, so they'll leave you alone.

There are a variety of ways to do this. In short, you're implementing a firewall restriction that only permits root and the mail server userid to make outbound port 25 (email) connections. You can either do it yourself with a software firewall, or, use third party software to do the same thing.

configserver.com has a variety of products and services that can deal with this issue. Note that the CBL has no connection whatsoever with ConfigServer. If you know of other software packages that can deal with Darkmailer please let us know and we'll mention them here.

The most commonly used ConfigServer product appears to be ConfigServer Security and Firewall (CSF) and it's FREE. The feature you want to turn on is "CSF SMTP_BLOCK" which, as far as we can tell, does exactly the firewall restrictions we describe above.

Another product that ConfigServer offers is ConfigServer eXploit Scanner (CXS). This software is not free ($75 regular price, currently $50). This software monitors FTP uploads in real-time, will automatically detect Darkmailer and other malicious downloads, and remove them.

Most Cpanel implementations already have something called "SMTP Tweak" (aka "WHM SMTP Tweak") available. It apparently doesn't do the firewall configuration we describe, but it wouldn't hurt to turn it on too.

If necessary, you can implement the firewall restrictions yourself without using any extra software.

```
iptables -A OUTPUT -d 127.0.0.1 -p tcp -m tcp --dport 25 -j ACCEPT
iptables -A OUTPUT -p tcp -m tcp --dport 25 -m owner --gid-owner mail -j ACCEPT
iptables -A OUTPUT -p tcp -m tcp --dport 25 -m owner --gid-owner mailman -j ACCEPT
iptables -A OUTPUT -p tcp -m tcp --dport 25 -m owner --uid-owner root -j ACCEPT
iptables -A OUTPUT -p tcp -m tcp --dport 25 -j REJECT --reject-with icmp-port-unreachable
```

You may need to add or change the "-m owner ... ACCEPT" to be consistent with your mail server. Eg: you'll need different entries for Qmail.

You will also have to ensure that these iptables commands are executed every time the system reboots, perhaps by an init script.

If you're using cPanel and APF, APF by default will wipe out iptables rules you enter manually leaving the server vulnerable. If you are using APF, you should make the above change via APF and that will take care of reissuing the commands upon reboot or reset.

Note: in some virtual hosting environments, the above commands will return error messages. This generally means that the host (not virtually hosted) operating system does not support the iptables kernel modules. If you do get such errors, make sure that the base operating system has the iptables module fully installed.

r57shell Infestations

In one case, it turned out to be a file called "info.php" in the user's images directory. Info.php turned out to be a modified copy of the "r57shell" PHP script which provides a backdoor through which an attacker can do virtually anything on your web server.

Thus, even though you have changed the passwords, the spammer could still upload the spamming scripts at will - this was found by noticing invocations of the PHP file in the web server logs from the same IP address the original FTP connections came from. You will need to search for such files as well, and we recommend preventing the execution of scripts (.php, .pl, .cgi, etc) in directories that do not need it. Eg: only the cgi-bin directory should permit execution. Nullamatix has a discussion on some simple ways to find r57shell.

It is known that Symantec EndPoint Protection can detect the original r57shell. The index.php file described above was a modified r57shell, and SEP doesn't detect it. A handful of AV detectors detect it as Backdoor.PHP.Rst!, but this doesn't help on Linux/UNIX. Note in particular, ClamAV **not** detect the "info.php" variant mentioned above. The script in the next section should find most versions of common trojan shells.

A Simple Trojan finding script in Perl

The following Perl script should find most trojan shells. It was chosen to be short and simple to understand so that an administrator would feel safe cut-and-pasting the script and running it. The script only looks in the most likely places, and therefore should run very fast as a quick check.

It is not expected that this script would find every possible trojan, so if this script fails to find anything malicious, you will need to resort to the other methods described above. Suggestions for improvement are welcome.

Instructions (only for webmasters with shell access to web hosting directories):

- Cut and paste the script to a file, we'll use "cblphpfind" for the purposes of these instructions.
- Make the script executable:

```
chmod 755 cblphpfind
```

- Run the script as a user with read access to the web hosting directories like this:

```
./cblphpfind [list of directories]
```

Note:if this script finds a directory it cannot open, it will stop. If it does stop, you will need to run this script as a user that has read access to the entire directory tree. The userid that runs the web server is a good choice. You can use "root" if you really have to.

On a Ubuntu system running Apache (not a hosting environment), the following command checked the web hosting directory:

```
./cblphpfind /var/www
```

If the web server uses aliases to redirect some web queries (including virtual hosts) to other directories, you should specify those too.

If you know which virtual host has the infection, you can try running cblphpfind on just the directories for that virtual host alone first.

- When it finds something suspicious it will print a line like:

```
/var/www/malicious/.htaccess: Suspicious(RewriteRule): RewriteRule . /phpinf
```

Which will name the file regarded suspicious, why it's suspicious (above found "RewriteRule"), and a fragment of the file showing the pattern.

- Anything that the script finds as "suspicious" should be visually inspected and removed if it turn out to be malicious.

```
#!/usr/bin/perl
# The above line may need to be changed to point at your version of Perl

# Very simple web malware detection module.
# Author: CBL Team <cbl@cbl.abuseat.org>
# Version 0.02
# Change history:
#       .01->.02: search 100 lines, add socket to scriptpat (2011/11/25)

# List of access-control files to check
my $access = '(\.htaccess)';
# Patterns to look for in access-control files
my $accesspat = '(RewriteRule)';

my $MAXLINES = 100;

# List of files to check
my $scripts = '\.(php|pl|cgi)$';
# Patterns to look for
my $scriptpat = '(socket|r57|c99|web shell|passthru|shell_exec|phpinfo|base64_decode|edoced_46esab|PHPShell)';

for my $dir (@ARGV) {
    &recursion($dir, $access, $accesspat);
    &recursion($dir, $scripts, $scriptpat);
}

sub recursion {
    my ($dir, $filepat, $patterns) = @_;
    my (@list);
    opendir(I, "$dir") || die "Can't open $dir: $!";
    @list = readdir(I);
    closedir(I);
    for my $file (@list) {
        next if $file =~ /^\.\.?$/;     # skip . and ..
        my $currentfile = "$dir/$file";
        if (-d $currentfile) {
            &recursion($currentfile, $filepat, $patterns);
        } elsif ($currentfile =~ /$filepat/) {
#print $currentfile, "\n";
            open(I, "<$currentfile") || next;
            my $linecount = 1;
            while(<I>) {
                chomp;
                if ($_ =~ /$patterns/) {
                    my $pat = $1;
                    my $string = $_;
                    if ($string =~ /^(.*)$pat(.*)$/) {
                        $string = substr($1, length($1)-10, 10) .
                                  $pat .
                                  substr($2, 0, 10);
                    }
                    #$string =~ s/^.*(.{,10}$pat.{,10}).*$/... $1 .../;
                    print "$currentfile: Suspicious($pat): $string\n";
                    last;
                }
                last if $linecount++ > $MAXLINES;
             }
             close(I);
            #print $currentfile, "\n";
        }
    }
```

```
}
```

WARNING: If you continually delist 1.2.3.4 without fixing the problem, the CBL will eventually stop allowing the delisting of 1.2.3.4.

If you have resolved the problem shown above and delisted the IP yourself, there is no need to contact us.

Click on this link to delist 1.2.3.4.

[<< Back to CBL homepage]