php-findsock-shell

This tool is designed for those situations during a pentest where you have upload access to a webserver that’s running PHP, you want an interactive shell, but the Firewall is doing proper egress and ingress filtering – so bindshells and reverse shells won’t work.

Upload php-findsock-shell to somewhere in the web root then run it by accessing the appropriate URL via netcat (as opposed to via a browser).  Instead of getting a normal HTTP response back, your HTTP session becomes an interactive shell session:

$ nc -v target 80

target [10.0.0.1] 80 (http) open

GET /php-findsock-shell.php HTTP/1.0

sh-3.2$ id

uid=80(apache) gid=80(apache) groups=80(apache)

sh-3.2$

... you now have an interactive shell ...

This will be a proper interactive shell in which you can run interective programs like telnet, ssh and su.  It differs from web form-based shells which allow you to send a single command, then return you the output.

Download

[php-findsock-shell-1.0.tar.gz](http://pentestmonkey.net/tools/php-findsock-shell/php-findsock-shell-1.0.tar.gz)

MD5sum: aecfea69fc6b482709f339756d6b419b

SHA1sum: 96e1a89cb15dcb64d81a13c2211faf98e80d3518

Preamble

I recently read about some (old) [vulnerabilities](http://www.securityfocus.com/archive/1/archive/1/449298/100/0/threaded) stemming from the fact that PHP scripts inherit some of Apache’s file descriptors and that programs run via PHP (e.g. via the “system” function) also inherit these file descriptors.  Vulnerabilities include:

* Overwriting apache log file – even though it’s owned by root; and
* Hijacking apache’s listening TCP port, then killing apache so you can respond to incoming web requests

This whole “inheriting file handles” class of vulnerability seems like an interesting area – albeit not a new area.

It occurred to me that if PHP could directly access the file handle corresponding to the TCP connection between the user’s browser and the web server, PHP might be able to attach a shell (/bin/sh) to this connection.  You’d then have an interactive shell which would pass straight through firewalls doing simple ingress and egree filtering.

Alas, I couldn’t figure out how to implement this idea purely in PHP.  However, with the help of a C program it’s possible to demonstrate that the idea works.

Walk Through

Modify the source to prevent unauthorised access

To prevent someone else from abusing your backdoor – a nightmare scenario while pentesting – you should modify the source code to prevent unauthorised access to your script.  This is left as an exercise to the reader.  I haven’t implemented this in v1.0.

Compile findsock.c

You need to compile findsock.c so that it’ll run on the web server you’ve gained access to.  If you’re running the same OS and architecture run the following locally:

$ gcc -o findsock findsock.c

If not, then you might have to compile elsewhere, or hope there’s a C compiler on the web server.

Upload “findsock” and “php-findsock-shell.php”

Upload the compiled binary plus the PHP script to the web server.

Enjoy your new shell

Access the PHP script using netcat (not using a browser).  If all went well, your HTTP connection should turn into an interactive shell:

$ nc -v target 80

target [10.0.0.1] 80 (http) open

GET /php-findsock-shell.php HTTP/1.0

sh-3.2$ id

uid=80(apache) gid=80(apache) groups=80(apache)

sh-3.2$

... you now have an interactive shell ...

FAQs

When is this useful?

When you can’t use a bindshell or reverse shell because of Firewall filtering.

Proper interactive shells are more useful than web-based shell in some circumstances, e.g:

* You want to change your user with “su”
* You want to upgrade your shell using a local exploit
* You want to log into another system using telnet / ssh

Isn’t the shell connection just going to be severed when the web server times out the PHP script?

No.  It doesn’t seem to on the systems that I’ve tested it on (Gentoo Linux only so far).

Isn’t there going to be a rather suspicious looking shell process when the admin runs “ps”?

Yeah.  :-)

Will this work with PERL as well as PHP?

I doesn’t seem to.  The PERL script doesn’t to inherit any interesting file handles.  It just has its STDIN, STDOUT and STDERR attached to a pipe.  However, I’m sure that there’s more than one way to configure PERL with Apache, so maybe other configurations will be prone the above problem.