This site is dedicated to sharing information about the practice, ideas, concepts and patterns regarding computer security.

How to securely isolate and execute Netcat from Kali Linux

Version 0.1, Last Updated: 6th Aug, 2023

Table of Contents

1. Introduction . . . 1
2. Requirements . . . 2
    2.1. Writing Conventions . . . 2
    2.2. VirtualBox . . . 2
        2.2.1. Clean VirtualBox Networking . . . 2
        2.2.2. Add VirtualBox Networking . . . 4
    2.3. Vagrant . . . 4
    2.4. Kali Linux and Damn Vulnerable Web Application (DVWA) . . . 5
        2.4.1. Vagrantfile . . . 5
        2.4.2. bootstrap.sh . . . 7
3. Netcat . . . 13
    3.1. Netcat Syntax . . . 13
    3.2. Netcat port scan . . . 16
    3.3. Netcat client and server testing connectivity . . . 17
    3.4. Transfer a file with Netcat . . . 18
    3.5. Reverse Shell . . . 18
4. Conclusion . . . 19
5. Appendix . . . 20

Chapter 1. Introduction

The motivation behind this paper is to explore using the tool Netcat that comes with Kali Linux.

Chapter 2. Requirements

2.1. Writing Conventions

If you see the following $ symbol on a command line to execute, what that means is that the command is executed as a regular user; meaning an account that does not have administrative privileges. Ignore the leading $ and execute the rest of the command.

```
$ command to execute as a regular user
```

If you see a command line lead with the # symbol, then that means that the command is executed as the root user. This implies you need to elevate to the root user before running the command, e.g. with: `sudo su  root`.

```
# command to execute as the root user
```

2.2. VirtualBox

Go to: https://www.virtualbox.org/wiki/Downloads and download VirtualBox.

The author is running on Ubuntu 18.04, so following to this URL: https://www.virtualbox.org/wiki/Linux_Downloads

For Ubuntu, double click on the .deb file, i.e. virtualbox-5.2_5.2.0-118431-Ubuntu-zesty_amd64.deb, and install VirtualBox on your local workstation.

2.2.1. Clean VirtualBox Networking

This section is here in case you already had virtualbox installed from before. The intent is to clean up the previous networking. If you do not need to do this, skip to Add VirtualBox Networking

Run these two commands from a Terminal:

```
$ VBoxManage list natnetworks
$ VBoxManage list dhcpservers
```

Output (example):

```
NetworkName:    192.168.139-NAT
IP:             192.168.139.1
Network:        192.168.139.0/24
IPv6 Enabled:   No
IPv6 Prefix:    fd17:625c:f037:2::/64
DHCP Enabled:   Yes
```

```
Enabled:        Yes
loopback mappings (ipv4)
        127.0.0.1=2


NetworkName:    192.168.139-NAT
Dhcpd IP:       192.168.139.3
LowerIPAddress: 192.168.139.101
UpperIPAddress: 192.168.139.254
NetworkMask:    255.255.255.0
Enabled:        Yes
Global Configuration:
    minLeaseTime:     default
    defaultLeaseTime: default
    maxLeaseTime:     default
    Forced options:   None
    Suppressed opts.: None
        1/legacy: 255.255.255.0
Groups:               None
Individual Configs:   None

NetworkName:    HostInterfaceNetworking-vboxnet0
Dhcpd IP:       172.20.0.3
LowerIPAddress: 172.20.0.101
UpperIPAddress: 172.20.0.254
NetworkMask:    255.255.255.0
Enabled:        Yes
Global Configuration:
    minLeaseTime:     default
    defaultLeaseTime: default
    maxLeaseTime:     default
    Forced options:   None
    Suppressed opts.: None
        1/legacy: 255.255.255.0
Groups:               None
Individual Configs:   None
```

Now, delete ALL of the pre-installed VirtualBox networks (one at a time following the syntax below):

```
VBoxManage natnetwork remove --netname <NetworkName_from_above>
VBoxManage natnetwork remove --netname 192.168.139-NAT
```

Repeat as many times as necessary to delete all of them.

Now, delete ALL of the pre-installed DHCP services:

```
VBoxManage dhcpserver remove --netname <DHCP_Server_NetworkName_from_above>
```

```
VBoxManage dhcpserver remove --netname 192.168.139-NAT
```

Repeat as many times as necessary to delete all of them.

2.2.2. Add VirtualBox Networking

Now, add the new VirtualBox networks so the Kali Linux guides work.

```
VBoxManage natnetwork add \
    --netname 192.168.139-NAT \
    --network "192.168.139.0/24" \
    --enable --dhcp on

VBoxManage dhcpserver add \
    --netname 192.168.139-NAT \
    --ip 192.168.139.3 \
    --lowerip 192.168.139.101 \
    --upperip 192.168.139.254 \
    --netmask 255.255.255.0 \
    --enable

VBoxManage hostonlyif create

VBoxManage hostonlyif ipconfig vboxnet0 \
    --ip 172.20.0.1 \
    --netmask 255.255.255.0

VBoxManage dhcpserver add \
    --ifname vboxnet0 \
    --ip 172.20.0.3 \
    --lowerip 172.20.0.101 \
    --upperip 172.20.0.254 \
    --netmask 255.255.255.0

VBoxManage dhcpserver modify \
    --ifname vboxnet0 \
    --enable
```

VirtualBox install complete.

2.3. Vagrant

Go to: https://www.vagrantup.com/downloads.html, follow the appropriate link to your OS and 32 or 64 bit version representing your local workstation. Download.

For Ubuntu, double click on the .deb file, i.e. vagrant_2.0.1_x86_64.deb, and install Vagrant on your local system.

NOTE Update vagrant vm: `vagrant box update`

2.4. Kali Linux and Damn Vulnerable Web Application (DVWA)

The author highly recommends to create a directory structure that is easy to navigate and find your code. As an example, you could use something similar to:

```
${HOME}/Source_Code/Education/vagrant-machines/kali-linux-vm/
```

Go ahead and make this structure with the following command (inside a Terminal):

```
$ mkdir p ${HOME}/Source_Code/Education/vagrant-machines/kali-linux-vm/
```

From a Terminal, change directory to:

```
$ cd ${HOME}/Source_Code/Education/vagrant-machines/kali-linux-vm/
```

2.4.1. Vagrantfile

Inside of the kali-linux-vm directory, populate a new file with the exact name, “Vagrantfile”. Case matters, uppercase the “V”. This file will contain both virtual machines for Kali Linux as well as setting up the DVWA virtual machine. Aggregating both virtual machines into one file has saved the author a lot of time. The coolness here is setting up the variables at the top of the Vagrantfile mimicing shell scripting inside of a virtual machine (passed in with provision: shell). I tested using: `apt-get update && apt-get upgrade -y`, but opted to take it out since it took over 45 minutes on my slower (old) hardware. See comment about downloading this file immediately preceding the code block.

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

$os_update = <<SCRIPT
apt-get update
SCRIPT

VAGRANTFILE_API_VERSION = "2"


Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
    config.vm.define "kali-linux-vagrant" do |conf|
        conf.vm.box = "kalilinux/rolling"

        # For Linux systems with the Wireless network, uncomment the line:
```

```
        conf.vm.network "public_network", bridge: "wlo1", auto_config: true

        # For macbook/OSx systems, uncomment the line and comment out the Linux
Wireless network:
        #conf.vm.network "public_network", bridge: "en0: Wi-Fi (AirPort)",
auto_config: true

        conf.vm.hostname = "kali-linux-vagrant"
        conf.vm.provider "virtualbox" do |vb|
            vb.gui = true
            vb.memory = "4096"
            vb.cpus = "2"
            vb.customize ["modifyvm", :id, "--vram", "32"]
            vb.customize ["modifyvm", :id, "--accelerate3d", "off"]
            vb.customize ["modifyvm", :id, "--ostype", "Debian_64"]
            vb.customize ["modifyvm", :id, "--boot1", "dvd"]
            vb.customize ["modifyvm", :id, "--boot2", "disk"]
            vb.customize ["modifyvm", :id, "--audio", "none"]
            vb.customize ["modifyvm", :id, "--clipboard", "hosttoguest"]
            vb.customize ["modifyvm", :id, "--draganddrop", "hosttoguest"]
            vb.customize ["modifyvm", :id, "--paravirtprovider", "kvm"]
        end
        conf.vm.provision "shell", inline: $os_update
    end

    config.vm.define "dvwa-vagrant" do |conf|

        conf.vm.box = "ubuntu/xenial64"

        conf.vm.hostname = "dvwa-vagrant"

        # For Linux systems with the Wireless network, uncomment the line:
        conf.vm.network "public_network", bridge: "wlo1", auto_config: true

        # For macbook/OSx systems, uncomment the line and comment out the Linux
Wireless network:
        #conf.vm.network "public_network", bridge: "en0: Wi-Fi (AirPort)",
auto_config: true

        config.vm.network "forwarded_port", guest: 80, host: 8080, auto_correct: true
        config.vm.network "forwarded_port", guest: 3306, host: 3306, auto_correct:
true

        conf.vm.provider "virtualbox" do |vb|
            vb.memory = "1024"
            vb.cpus = "2"
            vb.gui = false
            vb.customize ["modifyvm", :id, "--vram", "32"]
            vb.customize ["modifyvm", :id, "--accelerate3d", "off"]
            vb.customize ["modifyvm", :id, "--ostype", "Ubuntu_64"]
            vb.customize ["modifyvm", :id, "--boot1", "dvd"]
```

```
            vb.customize ["modifyvm", :id, "--boot2", "disk"]
            vb.customize ["modifyvm", :id, "--audio", "none"]
            vb.customize ["modifyvm", :id, "--clipboard", "hosttoguest"]
            vb.customize ["modifyvm", :id, "--draganddrop", "hosttoguest"]
            vb.customize ["modifyvm", :id, "--paravirtprovider", "kvm"]
        end
        conf.vm.provision "shell", inline: $os_update
        conf.vm.provision :shell, path: "bootstrap.sh"
    end
end
```

Save and write this file.

You can also download from:

```
$ curl -o Vagrantfile  http://securityhardening.com/files/Vagrantfile_20200928.txt
```

2.4.2. bootstrap.sh

Inside of the kali-linux-vm directory, populate a new file with the exact name, `bootstrap.sh`. Case matters, all lowercase. See comment about downloading this file immediately preceding the code block. `bootstrap.sh` (include the shebang in your file: the first line with `#!/usr/bin/env bash`):

```
#!/usr/bin/env bash
PHP_FPM_PATH_INI='/etc/php/7.0/fpm/php.ini'
PHP_FPM_POOL_CONF='/etc/php/7.0/fpm/pool.d/www.conf'
MYSQL_ROOT_PW='Assword12345'
MYSQL_dvwa_user='dvwa'
MYSQL_dvwa_password='sunshine'
DVWA_admin_password='admin'
recaptcha_public_key='u8392ihj32kl8hujalkshuil32'
recaptcha_private_key='89ry8932873832lih32ilj32'


install_base() {
    add-apt-repository -y ppa:nginx/stable
    sudo apt-get update
    sudo apt-get dist-upgrade -y
    sudo apt-get install -y \
        nginx \
        mariadb-server \
        mariadb-client \
        php \
        php-common \
        php-cgi \
        php-fpm \
        php-gd \
        php-cli \
```

```
        php-pear \
        php-mcrypt \
        php-mysql \
        php-gd \
        git \
        vim
}

config_mysql(){
    mysqladmin -u root password "${MYSQL_ROOT_PW}"
## Config the mysql config file for root so it doesn't prompt for password.
## Also sets pw in plain text for easy access.
## Don't forget to change the password here!!

cat <<EOF > /root/.my.cnf
[client]
user="root"
password="${MYSQL_ROOT_PW}"
EOF
    mysql -BNe "drop database if exists dvwa;"
    mysql -BNe "CREATE DATABASE dvwa;"
    mysql -BNe "GRANT ALL ON *.* TO '"${MYSQL_dvwa_user}"'@'localhost' IDENTIFIED BY
'"${MYSQL_dvwa_password}"';"

    systemctl enable mysql
    systemctl restart mysql
    sleep 2
}


config_php(){
    ## Config PHP FPM INI to disable some security settings:

    sed -i 's/^;cgi.fix_pathinfo.*$/cgi.fix_pathinfo = 0/g' ${PHP_FPM_PATH_INI}
    sed -i 's/allow_url_include = Off/allow_url_include = On/g' ${PHP_FPM_PATH_INI}
    sed -i 's/allow_url_fopen = Off/allow_url_fopen = On/g' ${PHP_FPM_PATH_INI}
    sed -i 's/safe_mode = On/safe_mode = Off/g' ${PHP_FPM_PATH_INI}
    echo "magic_quotes_gpc = Off" >> ${PHP_FPM_PATH_INI}
    sed -i 's/display_errors = Off/display_errors = On/g' ${PHP_FPM_PATH_INI}

    ## explicitly set pool options
    ## (these are defaults in ubuntu 16.04 so i'm commenting them out.
    ## If they are not defaults for you try uncommenting these)
    #sed -i 's/^;security.limit_extensions.*$/security.limit_extensions = \
    #.php .php3 .php4 .php5 .php7/g' /etc/php/7.0/fpm/pool.d/www.conf
    #sed -i 's/^listen.owner.*$/listen.owner = www-data/g'
/etc/php/7.0/fpm/pool.d/www.conf
    #sed -i 's/^listen.group.*$/listen.group = www-data/g'
/etc/php/7.0/fpm/pool.d/www.conf
    #sed -i 's/^;listen.mode.*$/listen.mode = 0660/g' /etc/php/7.0/fpm/pool.d/www.conf
```

```
    systemctl restart php7.0-fpm
}


config_nginx(){

cat << 'EOF' > /etc/nginx/sites-enabled/default
server
{
    listen  80;
    root /var/www/html;
    index index.php index.html index.htm;
    #server_name localhost
    location "/"
    {
        index index.php index.html index.htm;
        #try_files $uri $uri/ =404;
    }

    location ~ \.php$
    {
        include /etc/nginx/fastcgi_params;
        fastcgi_pass unix:/var/run/php/php7.0-fpm.sock;
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME $request_filename;
    }
}
EOF

    systemctl restart nginx
}


install_dvwa(){

    if [[ ! -d "/var/www/html" ]];
    then
          mkdir -p /var/www;
          ln -s /usr/share/nginx/html /var/www/html;
          chown -R www-data. /var/www/html;
    fi

    cd /var/www/html
    rm -rf /var/www/html/.[!.]*
    rm -rf /var/www/html/*
    git clone https://github.com/ethicalhack3r/DVWA.git ./
    chown -R www-data. ./
    cp config/config.inc.php.dist config/config.inc.php

    ### chmod uploads and log file to be writable by nobody
```

```
    chmod 777 ./hackable/uploads/
    chmod 777 ./external/phpids/0.6/lib/IDS/tmp/phpids_log.txt

    ## change the values in the config to match our setup (these are what you need to
update!
    sed -i '/db_user/ s/root/'${MYSQL_dvwa_user}'/'
/var/www/html/config/config.inc.php
    sed -i '/db_password/ s/p@ssw0rd/'${MYSQL_dvwa_password}'/'
/var/www/html/config/config.inc.php
    sed -i "/recaptcha_public_key/ s/''/'"${recaptcha_public_key}"'/"
/var/www/html/config/config.inc.php
    sed -i "/recaptcha_private_key/ s/''/'"${recaptcha_private_key}"'/"
/var/www/html/config/config.inc.php

}


update_mysql_user_pws(){
## The mysql passwords are set via /usr/share/nginx/html/dvwa/includes/DBMS/MySQL.php.
#  If you edit this every time they are reset it will reset to those.
#  Otherwise you can do a sql update statement to update them all (they are just md5's
of the string.
#  The issue is the users table doesn't get created until you click that button T_T to
init.

#mysql -BNe "UPDATE dvwa.users SET password = md5('YOUR_MYSQL_PW_HERE') WHERE user =
'admin';"
#mysql -BNe "UPDATE dvwa.users SET password = md5('YOUR_MYSQL_PW_HERE') WHERE user =
'gordonb';"
#mysql -BNe "UPDATE dvwa.users SET password = md5('YOUR_MYSQL_PW_HERE') WHERE user =
'1337';"
#mysql -BNe "UPDATE dvwa.users SET password = md5('YOUR_MYSQL_PW_HERE') WHERE user =
'pablo';"
#mysql -BNe "UPDATE dvwa.users SET password = md5('YOUR_MYSQL_PW_HERE') WHERE user =
'smithy';"

sed -i '/admin/ s/password/'${DVWA_admin_password}'/g'
/var/www/html/dvwa/includes/DBMS/MySQL.php
sed -i '/gordonb/ s/abc123/'${DVWA_admin_password}'/g'
/var/www/html/dvwa/includes/DBMS/MySQL.php
sed -i '/1337/ s/charley/'${DVWA_admin_password}'/g'
/var/www/html/dvwa/includes/DBMS/MySQL.php
sed -i '/pablo/ s/letmein/'${DVWA_admin_password}'/g'
/var/www/html/dvwa/includes/DBMS/MySQL.php
sed -i '/smithy/ s/password/'${DVWA_admin_password}'/g'
/var/www/html/dvwa/includes/DBMS/MySQL.php
}


install_base
config_mysql
```

```
install_dvwa
update_mysql_user_pws
config_php
config_nginx
```

Save and write this file.

If you have issues with copying and pasting the above file because code blocks in PDFs always copy correctly [NOT!], you could use curl, i.e. Make sure the bootstrap.sh file ends up in the same directory as the Vagrantfile.

```
$ curl -o bootstrap.sh  http://securityhardening.com/files/bootstrap_sh_20200928.txt
```

From a Terminal, change directory to:

```
$ cd ${HOME}/Source_Code/Education/vagrant-machines/kali-linux-vm/
```

Then run (inside the directory kali-linux-vm):

```
$ vagrant up
```

This will download the appropriate images and start the virtual machines. Once running, through the VirtuaBox GUI, login as root. Password is “toor”, root backwards. Edit the following file: `/etc/ssh/sshd_config`

And change the line: `#PermitRootLogin prothibit-password` To: `PermitRootLogin yes` Meaning strip the comment out on the beginning of the line and alter `prohibit-password` to `yes`.

Then restart the ssh daemon:

```
# kill HUP $(pgrep sshd)
```

Notice, you are on a Bridged adapter, this will open the instance to allow root to ssh in with the most unsecure password in the world. Only make this change (allowing root to login via SSH) if you require root SSH access. You can change the root user’s password, which is highly recommended.

For the DVWA instance, I would first run ‘vagrant status’ to capture the name that vagrant is using for the running instance.

```
# vagrant status
```

Choose

```
Current machine states:
```

```
kali-linux-vagrant running (virtualbox)
dvwa-vagrant running (virtualbox)
```

This environment represents multiple VMs. The VMs are all listed above with their current state. For more information about a specific VM, run `vagrant status NAME`.

From there, log into the DVWA instance with:

```
$ vagrant ssh dvwa-vagrant
```

And then get the current IP address.

```
$ ip a
```

Choose the second network adapter, it should look like:

```
ubuntu@dvwa:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen
1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group
default qlen 1000
    link/ether 02:53:17:3c:de:80 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global enp0s3
        valid_lft forever preferred_lft forever
    inet6 fe80::53:17ff:fe3c:de80/64 scope link
        valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group
default qlen 1000
    link/ether 08:00:27:f0:77:2d brd ff:ff:ff:ff:ff:ff
    inet 172.20.156.76/24 brd 172.20.156.255 scope global enp0s8
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fef0:772d/64 scope link
        valid_lft forever preferred_lft forever
```

The test network used for this paper uses 172.20.156.0/24 as the network range [shown here in section 3]. Therefore, the adapter, enp0s8 is what he is looking for. The IP to use as a target is 172.20.156.76. Write down your value.

Chapter 3. Netcat

A simple Linux tool that has been around for two decades. Netcat reads and writes data over network connections using either the TCP or UDP protocols. It is meant to act as a dependable tool that can conveniently be used by other programs and scripts in addition to being used directly from the command line. Due to its ability to create almost any type of connection you would need and its wealth of integrated functionality, it also serves as a feature-rich network debugging and forensics tool. Hence the very appropriate knickname: "Netcat, the Swiss Army Knife of network tools."

In the following examples, the IPs:

- 192.168.99.139 is a Kali Linux Virtual Machine
- 192.168.99.19 is a DVWA Virtual Machine

Let's get started.

3.1. Netcat Syntax

Let us take a gander at the man(ual) page.

```
┌──(root㉿kali-linux-vagrant)-[/tmp]
└─# man nc
NC(1)                                                       General Commands Manual
NC(1)

NAME
       nc - TCP/IP swiss army knife

SYNOPSIS
       nc [-options] hostname port[s] [ports] ...

       nc -l -p port [-options] [hostname] [port]

DESCRIPTION
       netcat is a simple unix utility which reads and writes data across network
connections, using TCP or UDP protocol. It is designed to be a reliable "backend" tool
that can be used directly or easily driven by other programs and scripts. At the same
time, it is a feature-rich network debugging and exploration tool, since it can create
almost any kind of connection you would need and has several interesting built-in
capabilities.
       Netcat, or "nc" as the actual program is named, should have been supplied long
ago as another one of those cryptic but standard Unix tools.

       In the simplest usage, "nc host port" creates a TCP connection to the given
port on the given target host. Your standard input is then sent to the host, and
anything that comes back across the connection is sent to your standard output. This
continues indefinitely, until the network side of the connection shuts down.
       Note that this behavior is different from most other applications which shut
everything down and exit after an end-of-file (EOF) on the standard input.
```

```
       Netcat can also function as a server, by listening for inbound connections on
arbitrary ports and then doing the same reading and writing. With minor limitations,
netcat doesn't really care if it runs in "client" or "server" mode  it still shovels
data back and forth until there isn't any more left. In either mode, shutdown can be
forced after a configurable time of inactivity on the network side.

       And it can do this via UDP too, so netcat is possibly the "udp telnet-like"
application you always wanted for testing your UDP-mode servers. UDP, as the "U"
implies, gives less reliable data transmission than TCP connections and some systems
may have trouble sending large amounts of data that way, but it's still a useful
capability to have.

       You may be asking "why not just use telnet to connect to arbitrary ports?"
Valid question, and here are some reasons. Telnet has the "standard input EOF"
problem, so one must introduce calculated delays in driving scripts to allow network
output to finish. This is the main reason netcat stays running until the *network*
side closes. Telnet also will not transfer arbitrary binary data, because certain
characters are interpreted as telnet options and are thus removed from the data
stream. Telnet also emits some of its diagnostic messages to standard output, where
netcat keeps such things religiously separated from its *output* and will never modify
any of the real data in transit unless you  *really* want it to. And of course telnet
is incapable of listening for inbound connections, or using UDP instead. Netcat
doesn't have any of these limitations, is much smaller and faster than telnet, and has
many other advantages.

OPTIONS
       -c string    specify shell commands to exec after connect (use with caution).
                    The string is passed to '/bin/sh -c' for execution.
                    See the  '-e' option if you don't have a working /bin/sh
                    (Note that POSIX-conformant system must have one).

       -e filename  specify filename to exec after connect (use with caution).
                    See the -c option for enhanced functionality.

       -g gateway   source-routing hop point[s], up to 8

       -G num       source-routing pointer: 4, 8, 12, ...

       -h           display help

       -i secs      delay interval for lines sent, ports scanned

       -l           listen mode, for inbound connects

       -n           numeric-only IP addresses, no DNS

       -o file      hex dump of traffic

       -p port      local port number (port numbers can be individual or ranges: lo-hi
[inclusive])
```

```
       -q seconds   after EOF on stdin, wait the specified number of seconds and then
quit. If seconds is negative, wait forever.

       -b           allow UDP broadcasts

       -r           randomize local and remote ports

       -s addr      local source address

       -t           enable telnet negotiation

       -u           UDP mode

       -v           verbose [use twice to be more verbose]

       -w secs      timeout for connects and final net reads

       -C           Send CRLF as line-ending

       -z           zero-I/O mode [used for scanning]

       -T type      set TOS flag (type may be one of "Minimize-Delay",
"Maximize-Throughput", "Maximize-Reliability", or "Minimize-Cost".)

COPYRIGHT
       Netcat is entirely my own creation, although plenty of other code was used as
examples. It is freely given away to the Internet community in the hope that it will
be useful, with no restrictions except giving credit where it is due. No GPLs,
Berkeley copyrights or any of that non-sense. The author assumes NO responsibility for
how anyone uses it. If netcat makes you rich somehow and you're feeling generous, mail
me a check. If you are affiliated in any way with Microsoft Network, get a life.
Always ski in control. Comments, questions, and patches to hobbit@avian.org.

NOTES
       Some port names in /etc/services contain hyphens -- netcat currently will not
correctly parse those unless you escape the hyphens with back-slashes (e.g. "netcat
localhost -ftp\-data-").

BUGS
       Efforts have been made to have netcat "do the right thing" in all its various
modes. If you believe that it is doing the wrong thing under whatever circumstances,
please notify me and tell me how you think it should behave. If netcat is not able to
do some task you think up, minor tweaks to the code will probably fix that. It
provides a basic and easily-modified template for writing other network applications,
and I certainly encourage people to make custom mods and send in any improvements they
make to it. Continued feedback from the Internet community is always welcome!

EXAMPLES
       For several netcat recipes:
       - /usr/share/doc/netcat/README.gz
```

```
       - /usr/share/doc/netcat/README.Debian.gz

AUTHOR
       Netcat was written by a guy we know as the Hobbit <hobbit@avian.org>.
```

Netcat help shows more concise syntax options available:

```
┌──(root㉿kali-linux-vagrant)-[/tmp]
└─# nc -h
[v1.10-47]
connect to somewhere:   nc [-options] hostname port[s] [ports] ...
listen for inbound: nc -l -p port [-options] [hostname] [port]
options:
    -c shell commands   as `-e'; use /bin/sh to exec [dangerous!!]
    -e filename     program to exec after connect [dangerous!!]
    -b          allow broadcasts
    -g gateway      source-routing hop point[s], up to 8
    -G num          source-routing pointer: 4, 8, 12, ...
    -h          this cruft
    -i secs         delay interval for lines sent, ports scanned
        -k                      set keepalive option on socket
    -l          listen mode, for inbound connects
    -n          numeric-only IP addresses, no DNS
    -o file         hex dump of traffic
    -p port         local port number
    -r          randomize local and remote ports
    -q secs         quit after EOF on stdin and delay of secs
    -s addr         local source address
    -T tos          set Type Of Service
    -t          answer TELNET negotiation
    -u          UDP mode
    -v          verbose [use twice to be more verbose]
    -w secs     timeout for connects and final net reads
    -C          Send CRLF as line-ending
    -z          zero-I/O mode [used for scanning]

       port numbers can be individual or ranges: lo-hi [inclusive];
       hyphens in port names must be backslash escaped (e.g. 'ftp\-data').
```

3.2. Netcat port scan

Referencing above, we can see that:

- option `-z` is used for scanning
- option `-v` is to increase the output verbosity
- option `-w` is the timeout for connects and final net reads

```
┌──(root㉿kali-linux-vagrant)-[/tmp]
└─# nc -z -v -w 1 192.168.99.19 80-10000
192.168.99.19: inverse host lookup failed: Unknown host
(UNKNOWN) [192.168.99.19] 111 (sunrpc) open
(UNKNOWN) [192.168.99.19] 80 (http) open
```

That was super interesting. If you don't have nmap available, netcat can be used for a port scanner.

3.3. Netcat client and server testing connectivity

Server Side, we need to setup using the Server's IP address.

- option `-l` is for listen mode, for inbound connects
- option `-v` is to increase the output verbosity
- option `-n` is numeric-only IP addresses, no DNS lookups
- option `-p` is for the local port number

Again, this is server side

```
root@debian-dvwa:~# nc -lvnp 87 192.168.99.19
```

Client Side, we need to then target the IP and the port.

```
nc -nv 192.168.99.19 87
```

- option `-n` is numeric-only IP addresses, no DNS lookups
- option `-v` is to increase the output verbosity

Outputs: - Client side:

```
┌──(root㉿kali-linux-vagrant)-[/tmp]
└─# nc -nv 192.168.99.19 87
(UNKNOWN) [192.168.99.19] 87 (?) open
```

- Server side:

```
root@debian-dvwa:~# nc -lvnp 87 192.168.99.19
listening on [any] 87 ...
invalid connection to [192.168.99.19] from (UNKNOWN) [192.168.99.139] 51648
```

That may be useful in the future for simple tests.

3.4. Transfer a file with Netcat

Server or system you want to write the copied file to:

```
nc -l -p 87 > /tmp/foo.iso
```

Client or system you want to read the file from and send to remote target:

```
# generate the temp file for testing
fallocate -l 4G /tmp/foo.iso

# copy the file to the remote target. Using redirect to read the file in here
nc -q 1 -n 192.168.99.19 87 < /tmp/foo.iso
```

The option for `-q` will close the connection 1 second after the End of File is sent from the client, both on the client and the server side.

3.5. Reverse Shell

source: https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Reverse%20Shell%20Cheatsheet.md#netcat-traditional

Server/target Opening the reverse shell:

```
nc -l -p 87 -c /bin/sh
```

Client connecting to the server/target IP address with the listener and remote shell:

```
nc -n 192.168.99.19 87
```

The netcat command will do nothing more, on the client side, until you type.

So enter in a few basic commands to know that you are remotely connected.

```
┌──(root㉿kali-linux-vagrant)-[/tmp]
└─# nc -n 192.168.99.19 87
whoami
root
pwd
/tmp
uptime
 08:42:10 up 54 min,  1 user,  load average: 0.00, 0.08, 0.06
```

Chapter 4. Conclusion

Netcat has been around for decades and is still an amazing tool for testing and down and dirty file transfers. Typically, I prefer to use Secure Copy (SCP) for my file transfers. But if you couldn't install Secure Shell (SSH) or Secure FTP (SFTP) then netcat could be your tool if available, for a quick file transfer if encryption was not necessary. If someone wanted to test un-encrypted network transfers, then this approach would be a solid choice. As shown above, it allows for quick network port scans [as an alternative to `nmap` scans].

Now, having stated the positive uses of Netcat, there is a darker side which needs to be stated for our readers that don't know about such nefarious matters. That is the fact that Netcat can be used for unsanctioned data egress, as well as reverse tunnels back into secure networks. Most organizations typically reject the use of Netcat within their network perimeters. Several Organizations that I am aware of normally have their antivirus and/or Host Based Security System (HBSS) configred to deny execution, as well as alert on the use of Netcat on any host within their control. For these Organizations, this is a good approach to help prevent arbitrary execution of software programs, malware and Remote Access Trojans (RAT).

That stated, before launching Netcat on a customer site, get written approval that you can install and launch the tool [typically an email will suffice from the approving authority]. For Homelab execution, this is an amazing tool and should be in the back of your brain if you need a quick TCP or UDP listener to verify remote connectivity inside of your network boundary [Homelab].

As always, keep the System Secure and Live a good life.

Chapter 5. Appendix

References

https://www.kali.org/tools/netcat/

source code: https://salsa.debian.org/debian/netcat

https://github.com/swisskyrepo/PayloadsAllTheThings