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

How to securely isolate and execute NetSED from Kali Linux

Version 0.1, Last Updated: 5th July, 2023

Table of Contents

Chapter 1. Introduction

The motivation behind this paper is to explore using the tool NetSED 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. NetSED

NetSED is a tool that allows users to alter the contents of TCP and UDP data packets on their local network. You can directly alter unencrypted data streams with the NetSED command-line tool, enabling data integrity checks and black-box protocol verification. Only machines that route packages (such as routers and firewalls) can use the NetSED service to intercept and transform traffic. The software acts as a proxy by sniffing the data stream from a specified connection and then manipulating it in real time in accordance with your established rules. Both manually starting the application and including it in a shell script are valid options. Normal user privileges are enough if you utilize local port numbers higher than 1023 [e.g. ports 1024-65535]; otherwise, root privileges are required for ports less than port 1024.

3.1. NetSED install

On the version of Kali I am using to test this, `netsed` is already installed by default.

If your version of Kali Linux is missing the tool, as the `root` user, simply run in a terminal:

```
apt install -y netsed
```

3.2. NetSED help

```
┌──(root㉿kali-linux-vagrant)-[~]
└─# netsed --help
netsed 1.2 by Julien VdG <julien@silicone.homelinux.org>
      based on 0.01c from Michal Zalewski <lcamtuf@ids.pl>
Usage: netsed [option] proto lport rhost rport rule1 [ rule2 ... ]

  options - can be --ipv4 or -4 to force address resolution in IPv4,
            --ipv6 or -6 to force address resolution in IPv6,
            --ipany to resolve the address in either IPv4 or IPv6.
          - --help or -h to display this usage information.
  proto   - protocol specification (tcp or udp)
  lport   - local port to listen on (see README for transparent
            traffic intercepting on some systems)
  rhost   - where connection should be forwarded (0 = use destination
            address of incoming connection, see README)
  rport   - destination port (0 = dst port of incoming connection)
  ruleN   - replacement rules (see below)

General syntax of replacement rules: s/pat1/pat2[/expire]

This will replace all occurrences of pat1 with pat2 in any matching packet.
An additional parameter, 'expire' of the form [CHAR][NUM], can be used to
expire a rule after NUM successful substitutions during a given connection.
The character CHAR is one of "iIoO", with the effect of restricting the rule
to apply to incoming ("iI") or to outgoing ("oO") packets only, as seen from
```

```
the client's perspective. Both of CHAR and NUM are optional.

Eight-bit characters, including NULL and '/', can be applied using HTTP-like
hex escape sequences (e.g. CRLF as %0a%0d).
A match on '%' can be achieved by specifying '%%'.

Examples:
  's/andrew/mike/1'     - replace 'andrew' with 'mike' (only first time)
  's/andrew/mike'       - replace all occurrences of 'andrew' with 'mike'
  's/andrew/mike%00%00' - replace 'andrew' with 'mike\x00\x00'
                          (manually padding to keep original size)
  's/%%/%2f/20'         - replace the 20 first occurrence of '%' with '/'
  's/andrew/mike/o'     - the server will always see 'mike', never 'andrew'

  's/Rilke/Proust/o s/Proust/Rilke/i'
                        - let Rilke travel incognito as Proust

Rules are not active across packet boundaries, and they are evaluated
from first to last, not yet expired rule, as stated on the command line.
```

3.3. NetSED Syntax

```
Usage: netsed [option] proto lport rhost rport rule1 [ rule2 ... ]
```

Supported protocols are either tcp or udp, and we can address the `rhost` [remote target system] by either its hostname or its IP address.

The Stream Editor (SED) part of the tool supports patterns in the form of:

```
's/<Search>/<Replace>/'
```

If you leave the field <Replace> blank, the tool will follow your pattern and delete the specified search term from the data stream.

The software can also process multiple statements at the same time. By adding a numeric parameter to the end of the search/replace pattern, we can restrict the number of change actions per stream. If you want the program to perform a change exactly once, you would use:

```
's/<old>/<new>/1'
```

In the above example, `netsed` will only replace the first occance of the `<old>` pattern matched.

3.4. Using NetSED

```
netsed tcp 9876 192.168.99.19 80 's/test/evaluation/'
```

The above IP address ending with the octet `.19` is my DVWA instance.

Then in a browser, I want go to the proxy IP address for the Kali Linux server:

http://192.168.99.139:9876/test

The output from the proxy command [netsed] is:

```
 ┌──(root㉿kali-linux-vagrant)-[~]
 └─# netsed tcp 9876 192.168.99.19 80 's/test/evaluation/'
netsed 1.2 by Julien VdG <julien@silicone.homelinux.org>
      based on 0.01c from Michal Zalewski <lcamtuf@ids.pl>
[*] Parsing rule s/test/evaluation/...
[+] Loaded 1 rule...
[+] Using fixed forwarding to 192.168.99.19,80.
[+] Listening on port 9876/tcp.
[+] Got incoming connection from 192.168.99.188,36434 to 192.168.99.188,36434
[*] Forwarding connection to 192.168.99.19,80
[+] Caught client -> server packet.
[*] Forwarding untouched packet of size 391.
[+] Caught server -> client packet.
[*] Forwarding untouched packet of size 3380.
[+] Caught client -> server packet.
[*] Forwarding untouched packet of size 388.
[+] Caught server -> client packet.
[*] Forwarding untouched packet of size 6040.
[+] Caught client -> server packet.
[*] Forwarding untouched packet of size 378.
[+] Caught server -> client packet.
[*] Forwarding untouched packet of size 507.
[+] Got incoming connection from 192.168.99.188,36446 to 192.168.99.188,36446
[*] Forwarding connection to 192.168.99.19,80
[+] Caught client -> server packet.
    Applying rule s/test/evaluation...
[*] Done 1 replacements, forwarding packet of size 401 (orig 395).
[+] Caught server -> client packet.
[*] Forwarding untouched packet of size 507.
```

The web page of course returns an error in the form of:

```
Not Found
The requested URL /evaluation was not found on this server.

Apache/2.4.10 (Debian) Server at 192.168.99.139 Port 9876
```

My initial reaction to this is very good. I have been a fan of the original `sed` tool for roughly 25 years at the time of this writing. It gets the job done and is extremely effecient. My head is exploding with ideas for URL and Header manipulation leveraging this tool.

Let’s start tinkering with this more.

```
netsed tcp 9876 192.168.99.19 80 's/apache/Awesomeness/'
```

Output:

```
 ┌──(root㉿kali-linux-vagrant)-[~]
 └─# netsed tcp 9876 192.168.99.19 80 's/apache/Awesomeness/'
netsed 1.2 by Julien VdG <julien@silicone.homelinux.org>
      based on 0.01c from Michal Zalewski <lcamtuf@ids.pl>
[*] Parsing rule s/apache/Awesomeness/...
[+] Loaded 1 rule...
[+] Using fixed forwarding to 192.168.99.19,80.
[+] Listening on port 9876/tcp.
[+] Got incoming connection from 192.168.99.188,48946 to 192.168.99.188,48946
[*] Forwarding connection to 192.168.99.19,80
[+] Caught client -> server packet.
[*] Forwarding untouched packet of size 509.
[+] Caught server -> client packet.
[*] Forwarding untouched packet of size 3380.
[+] Caught client -> server packet.
[*] Forwarding untouched packet of size 509.
[+] Caught server -> client packet.
[*] Forwarding untouched packet of size 3379.
[+] Got incoming connection from 192.168.99.188,44568 to 192.168.99.188,44568
[*] Forwarding connection to 192.168.99.19,80
[+] Caught client -> server packet.
[*] Forwarding untouched packet of size 83.
[+] Caught server -> client packet.
    Applying rule s/apache/Awesomeness...
    Applying rule s/apache/Awesomeness...
    Applying rule s/apache/Awesomeness...
    Applying rule s/apache/Awesomeness...
    Applying rule s/apache/Awesomeness...
    Applying rule s/apache/Awesomeness...
    Applying rule s/apache/Awesomeness...
    Applying rule s/apache/Awesomeness...
    Applying rule s/apache/Awesomeness...
    Applying rule s/apache/Awesomeness...
    Applying rule s/apache/Awesomeness...
    Applying rule s/apache/Awesomeness...
    Applying rule s/apache/Awesomeness...
    Applying rule s/apache/Awesomeness...
[*] Done 14 replacements, forwarding packet of size 11026 (orig 10956).
```

By the way, I switched testing methods from the browser to the `curl` command to make it easier to show the commands executed.

Output from the `curl` command from testing:

```
curl http://192.168.99.139:9876/

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!-- part of the output has been stripped for brevity -->
  <body>
  <!-- a wee bit more of the output has been stripped for brevity -->

/etc/Awesomeness2/
|-- Awesomeness2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
          </pre>
          <ul>
                        <li>
                           <tt>Awesomeness2.conf</tt> is the main configuration
                           file. It puts the pieces together by including all
remaining configuration

                           files when starting up the web server.
                        </li>

                        <li>
                           <tt>ports.conf</tt> is always included from the
                           main configuration file. It is used to determine the
listening ports for

                           incoming connections, and this file can be customized
anytime.
                        </li>

                        <li>
                           Configuration files in the <tt>mods-enabled/</tt>,
                           <tt>conf-enabled/</tt> and <tt>sites-enabled/</tt>
directories contain

                           particular configuration snippets which manage modules,
global configuration

                           fragments, or virtual host configurations, respectively.
                        </li>

                        <li>
```

```
                           They are activated by symlinking available
                           configuration files from their respective
                           *-available/ counterparts. These should be managed
                           by using our helpers
                           <tt>
                                a2enmod,
                                a2dismod,
                           </tt>
                           <tt>
                                a2ensite,
                                a2dissite,
                            </tt>
                                and
                           <tt>
                                a2enconf,
                                a2disconf
                           </tt>. See their respective man pages for detailed
information.
                        </li>

                        <li>
                           The binary is called Awesomeness2. Due to the use of
                           environment variables, in the default configuration,
Awesomeness2 needs to be
                           started/stopped with <tt>/etc/init.d/Awesomeness2</tt> or
<tt>Awesomeness2ctl</tt>.
                           <b>Calling <tt>/usr/bin/Awesomeness2</tt> directly will not
work</b> with the
                           default configuration.
                        </li>
          </ul>
        </div>

        <div class="section_header">
            <div id="docroot"></div>
                Document Roots
        </div>

        <div class="content_section_text">
            <p>
                By default, Debian does not allow access through the web browser to
                <em>any</em> file apart of those located in <tt>/var/www</tt>,
                <a href="http://httpd.Awesomeness.org/docs/2.4/mod/mod_userdir.html"
rel="nofollow">public_html</a>
                directories (when enabled) and <tt>/usr/share</tt> (for web
                applications). If your site is using a web document root
                located elsewhere (such as in <tt>/srv</tt>) you may need to whitelist
your
                document root directory in
<tt>/etc/Awesomeness2/Awesomeness2.conf</tt>.
            </p>
            <p>
                The default Debian document root is <tt>/var/www/html</tt>. You
                can make your own virtual hosts under /var/www. This is different
                to previous releases which provides better security out of the box.
            </p>
        </div>

        <div class="section_header">
          <div id="bugs"></div>
                Reporting Problems
        </div>
        <div class="content_section_text">
          <p>
                Please use the <tt>reportbug</tt> tool to report bugs in the
                Apache2 package with Debian. However, check <a
                href="http://bugs.debian.org/cgi-
bin/pkgreport.cgi?ordering=normal;archive=0;src=Awesomeness2;repeatmerged=0"
                rel="nofollow">existing bug reports</a> before reporting a new bug.
          </p>
          <p>
                Please report bugs specific to modules (such as PHP and others)
                to respective packages, not to the web server itself.
          </p>
        </div>
```

I am super excited with these results. I know, I know! I am easy to impress when it comes to simple tools that do one thing really, really well.

Final test, I want to change multiple values in the stream.

Test 3:

```
┌──(root㉿kali-linux-vagrant)-[~]
└─# netsed tcp 9876 192.168.99.19 80 's/apache/Awesomeness/' 's/Debian/Linux/'
netsed 1.2 by Julien VdG <julien@silicone.homelinux.org>
      based on 0.01c from Michal Zalewski <lcamtuf@ids.pl>
[*] Parsing rule s/apache/Awesomeness/...
[*] Parsing rule s/Debian/Linux/...
[+] Loaded 2 rules...
[+] Using fixed forwarding to 192.168.99.19,80.
[+] Listening on port 9876/tcp.
[+] Got incoming connection from 192.168.99.188,37778 to 192.168.99.188,37778
[*] Forwarding connection to 192.168.99.19,80
[+] Caught client -> server packet.
[*] Forwarding untouched packet of size 83.
[+] Caught server -> client packet.
    Applying rule s/Debian/Linux...
    Applying rule s/Debian/Linux...
    Applying rule s/Debian/Linux...
    Applying rule s/Debian/Linux...
    Applying rule s/Debian/Linux...
    Applying rule s/Debian/Linux...
    Applying rule s/Debian/Linux...
    Applying rule s/apache/Awesomeness...
    Applying rule s/Debian/Linux...
    Applying rule s/apache/Awesomeness...
    Applying rule s/Debian/Linux...
    Applying rule s/apache/Awesomeness...
    Applying rule s/apache/Awesomeness...
    Applying rule s/apache/Awesomeness...
    Applying rule s/apache/Awesomeness...
    Applying rule s/apache/Awesomeness...
    Applying rule s/apache/Awesomeness...
    Applying rule s/apache/Awesomeness...
    Applying rule s/apache/Awesomeness...
    Applying rule s/Debian/Linux...
    Applying rule s/apache/Awesomeness...
    Applying rule s/apache/Awesomeness...
    Applying rule s/apache/Awesomeness...
    Applying rule s/Debian/Linux...
    Applying rule s/Debian/Linux...
    Applying rule s/apache/Awesomeness...
[*] Done 26 replacements, forwarding packet of size 11014 (orig 10956).
```

And the output from the curl command:

```
curl http://192.168.99.139:9876/
0ec7aa  main

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- part of the output has been stripped for brevity -->
  <body>
  <!-- a wee bit more of the output has been stripped for brevity -->

/etc/Awesomeness2/
|-- Awesomeness2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
          </pre>
          <ul>
                        <li>
                           <tt>Awesomeness2.conf</tt> is the main configuration
                           file. It puts the pieces together by including all
```

```
remaining configuration
                           files when starting up the web server.
                        </li>

                        <li>
                           <tt>ports.conf</tt> is always included from the
                           main configuration file. It is used to determine the
listening ports for
                           incoming connections, and this file can be customized
anytime.
                        </li>

                        <li>
                           Configuration files in the <tt>mods-enabled/</tt>,
                           <tt>conf-enabled/</tt> and <tt>sites-enabled/</tt>
directories contain
                           particular configuration snippets which manage modules,
global configuration
                           fragments, or virtual host configurations, respectively.
                        </li>

                        <li>
                           They are activated by symlinking available
                           configuration files from their respective
                           *-available/ counterparts. These should be managed
                           by using our helpers
                           <tt>
                                a2enmod,
                                a2dismod,
                           </tt>
                           <tt>
                                a2ensite,
                                a2dissite,
                            </tt>
                                and
                           <tt>
                                a2enconf,
                                a2disconf
                           </tt>. See their respective man pages for detailed
information.
                        </li>

                        <li>
                           The binary is called Awesomeness2. Due to the use of
                           environment variables, in the default configuration,
Awesomeness2 needs to be
                           started/stopped with <tt>/etc/init.d/Awesomeness2</tt> or
<tt>Awesomeness2ctl</tt>.
                           <b>Calling <tt>/usr/bin/Awesomeness2</tt> directly will not
work</b> with the
                           default configuration.
                        </li>
          </ul>
        </div>

        <div class="section_header">
            <div id="docroot"></div>
                Document Roots
        </div>

        <div class="content_section_text">
            <p>
                By default, Linux does not allow access through the web browser to
                <em>any</em> file apart of those located in <tt>/var/www</tt>,
                <a href="http://httpd.Awesomeness.org/docs/2.4/mod/mod_userdir.html"
rel="nofollow">public_html</a>
                directories (when enabled) and <tt>/usr/share</tt> (for web
                applications). If your site is using a web document root
                located elsewhere (such as in <tt>/srv</tt>) you may need to whitelist
your
                document root directory in
<tt>/etc/Awesomeness2/Awesomeness2.conf</tt>.
            </p>
            <p>
                The default Linux document root is <tt>/var/www/html</tt>. You
                can make your own virtual hosts under /var/www. This is different
                to previous releases which provides better security out of the box.
            </p>
        </div>

        <div class="section_header">
          <div id="bugs"></div>
                Reporting Problems
        </div>
        <div class="content_section_text">
          <p>
                Please use the <tt>reportbug</tt> tool to report bugs in the
                Apache2 package with Linux. However, check <a
                href="http://bugs.debian.org/cgi-
bin/pkgreport.cgi?ordering=normal;archive=0;src=Awesomeness2;repeatmerged=0"
                rel="nofollow">existing bug reports</a> before reporting a new bug.
          </p>
          <p>
                Please report bugs specific to modules (such as PHP and others)
                to respective packages, not to the web server itself.
          </p>
        </div>




      </div>
```

```
    </div>
```

We can see that both terms, "apache|Debian" was edited to "Awesomeness|Linux", respectively. With this knowledge, you can chain as many terms as the command line will accept for a single line of input, depending on your Shell. For readability I would recommend a format like:

```
netsed tcp 9876 192.168.99.19 80 \
    's/Fii/term1/' \
    's/Fo/term2/' \
    's/Fum/term3/'
```

Chapter 4. Conclusion

"NetSED is small and handful utility designed to alter the contents of packets forwarded thru your network in real time. It is really useful for network hackers in following applications:

- black-box protocol auditing - whenever there are two or more propertiary boxes communicating over undocumented protocol (by enforcing changes in ongoing transmissions, you will be able to test if tested application is secure),
- fuzz-alike experiments, integrity tests - whenever you want to test stability of the application and see how it ensures data integrity,
- other common applications - fooling other people, content filtering, etc etc - choose whatever you want to." — NetSED docs

I am restating what the Developer originally wanted the community to know about his tool. The tests that I have performed are very simple and show us how easy the tool is to add into our workflows to modify data in either a TCP or UDP data stream across the network.

"It perfectly fits netgrep, netcat and tcpdump tools suite" — NetSED docs

I wholeheartedly agree with that statement. If you can leverage another tool to pull the data, NetSED is a great choice to perform Transformations on the data stream coming across the network.

This is of course another tool to add to your toolbelt, but worth keeping in the back of your active mind so that you have a clever trick that not only helps you solve the problem, but sets you apart from your colleagues as a person that gets stuff done [no matter how daunting the task at hand].

Keep securing the network. It was a pleasure to learn NetSED. Thank you Michal Zalewski for writing this tool.

Chapter 5. Appendix

References

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

http://silicone.homelinux.org/projects/netsed/

https://github.com/xlab/netsed