

# How to securely isolate and execute sfuzz from Kali Linux

Version 0.1, Last Updated: 26 Sep 2020

---



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

## 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 .....	3
2.3. Vagrant .....	4
2.4. Kali Linux and Damn Vulnerable Web Application (DVWA) .....	4
2.4.1. Vagrantfile .....	4
2.4.2. bootstrap.sh .....	7
3. sfuzz .....	11
4. Conclusion .....	15
5. Appendix .....	16

---

## 1. Introduction

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

*What is a Fuzzer/Fuzzing:*

"Fuzzing or fuzz testing is an automated software testing technique that involves providing invalid, unexpected, or random data as inputs to a computer program. The program is then monitored for exceptions such as crashes, failing built-in code assertions, or potential memory leaks. Typically, fuzzers are used to test programs that take structured inputs. This structure is specified, e.g., in a file format or protocol and distinguishes valid from invalid input. An effective fuzzer generates semi-valid inputs that are "valid enough" in that they are not directly rejected by the parser, but do create unexpected behaviors deeper in the program and are "invalid enough" to expose corner cases that have not been properly dealt with.

For the purpose of security, input that crosses a trust boundary is often the most interesting. For example, it is more important to fuzz code that handles the upload of a file by any user than it is to fuzz the code that parses a configuration file that is accessible only to a privileged user." **source:** [Fuzzing](#)

*What is this tool:*

" simple fuzz is exactly what it sounds like – a simple fuzzer. don't mistake simple with a lack of fuzz capability. this fuzzer has two network modes of operation, an output mode for developing command line fuzzing scripts, as well as taking fuzzing strings from literals and building strings from sequences.

simple fuzz is built to fill a need – the need for a quickly configurable black box testing utility that doesn't require intimate knowledge of the inner workings of C or require specialized software rigs. the aim is to just provide a simple interface, clear inputs/outputs, and reusability.

Features: - simple script language for creating test cases - support for repeating strings as well as fixed strings ('sequences' vs. 'literals') - variables within test cases (ex: strings to be replaced with different strings) - tcp and udp payload transport (icmp support tbd) - binary substitution support (see basic.a11 for more information) - plugin support (NEW!) see plugin.txt for more information. - previous packet contents inclusion " **source:** <https://tools.kali.org/vulnerability-analysis/sfuzz>[Kali Linux]

I am excited to use this tool.

---

## 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](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.

## 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 "dwaa-vagrant" do |conf|

    conf.vm.box = "ubuntu/xenial64"

    conf.vm.hostname = "dwaa-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 prohibit-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.

---

### 3. sfuzz

On your Host system, running VirtualBox and Vagrant, open a terminal and run:

```
$ vagrant status
```

Then SSH into the Kali Linux namespace

```
$ vagrant ssh kali-linux-vagrant
```

Elevate to the root user (inside the Virtual Machine for Kali Linux)

```
$ sudo su -
```

Install sfuzz (inside the Virtual Machine for Kali Linux)

```
# apt-get install -y sfuzz
```

To launch an attack against our DVWA, we will need to use this format:

```
# sfuzz -S <IP_Address> -p <port_number> -T -f /usr/share/sfuzz-db/basic.http
```

In case my IP address Changed for DVWA, I am going to run this command from my Host

```
$ vagrant ssh dvwa-vagrant -c "ip a"
```

I am going to run **sfuzz** with this format to evaluate the security posture of DVWA

```
# ( time sfuzz -S 172.20.156.142 -p 80 -T -f /usr/share/sfuzz-db/basic.http & ) 1> /tmp/sfuzz.out 2>&1
```

And then immediately tail the output file

```
# tail -f /tmp/sfuzz.out
```

Parse through the output looking for vulnerabilities. For me, I am not seeing anything of value.

To see all of the attacks that have been prepackaged with the software, change directory to this directory

```
# cd /usr/share/sfuzz-db
```

What is available

```

root@kali-linux-vagrant:/usr/share/sfuzz-db# ls -l
total 120
-rwxr-xr-x 1 root root 4683 Nov 24 2015 bad-nums.blocks.inc
-rwxr-xr-x 1 root root 3222 Nov 24 2015 basic.all
-rwxr-xr-x 1 root root 195 Nov 24 2015 basic.cmd
-rwxr-xr-x 1 root root 1249 Nov 24 2015 basic.cvs
-rwxr-xr-x 1 root root 1391 Nov 24 2015 basic-fuzz-strings.list
-rwxr-xr-x 1 root root 1890 Nov 24 2015 basic.http
-rwxr-xr-x 1 root root 847 Nov 24 2015 basic.http.blocks
-rwxr-xr-x 1 root root 743 Nov 24 2015 basic.nuke
-rwxr-xr-x 1 root root 946 Nov 24 2015 basic.pop3
-rwxr-xr-x 1 root root 1183 Nov 24 2015 basic.rtsp
-rwxr-xr-x 1 root root 1000 Nov 24 2015 basic.smtp
-rwxr-xr-x 1 root root 52 Nov 24 2015 basic.unknown
-rwxr-xr-x 1 root root 4589 Nov 24 2015 big-ant.0day
-rwxr-xr-x 1 root root 1218 Nov 24 2015 http-etc-enumeration.list
-rwxr-xr-x 1 root root 72 Nov 24 2015 http-nuke-enumeration.list
-rwxr-xr-x 1 root root 267 Nov 24 2015 php-vuln.test
-rwxr-xr-x 1 root root 12086 Nov 24 2015 server.basic.http
-rwxr-xr-x 1 root root 2191 Nov 24 2015 server.browser-fuzz.blocks
-rw-r--r-- 1 root root 5848 Nov 24 2015 sfuzz-plugin-example.so
-rw-r--r-- 1 root root 8144 Nov 24 2015 sfuzz-server-plugin.so
-rwxr-xr-x 1 root root 2300 Nov 24 2015 smb.0day
-rwxr-xr-x 1 root root 2421 Nov 24 2015 std-cmdline-exploits.list
-rwxr-xr-x 1 root root 1186 Nov 24 2015 twitter.cfg
-rwxr-xr-x 1 root root 2403 Nov 24 2015 vulnerable-echo-server.c

```

## Truncated attack sequence

```

root@kali-linux-vagrant:/tmp# cat sfuzz.out
[17:17:30] dumping options:
  filename: </usr/share/sfuzz-db/basic.http>
  state: ✘
  lineno:
  literals: [74]
  sequences: [34]
  symbols: [0]
  req_del:
  mseq_len:
  plugin: <none>
  s_syms: ✘
  literal[1] = [AREALLYBADSTRING]
  literal[2] = [oaiwrkkgaoiul;234987 103984a;lk-814 1]
  literal[3] = [0]
  literal[4] = [0]
  literal[5] = [0]
  literal[6] = [%n]
  literal[7] = [%#123456x]
  literal[8] = [%s]
  literal[9] = [%%s]
  literal[10] = [%20s]
  literal[11] = [%%20s]
  literal[12] = [%20x]
  literal[13] = [%%20x]
  literal[14] = [%n%n%n%n%n]
  literal[15] = [%p%p%p%p%p]
  literal[16] = [%x%x%x%x%x]
  literal[17] = [%d%d%d%d%d]
  literal[18] = [%s%s%s%s%s]
  literal[19] = [%s%p%x%d]
  literal[20] = [%.1024d]
  literal[21] = [%.1025d]
  literal[22] = [%.2048d]
  literal[23] = [%.2049d]
  literal[24] = [%.4096d]
  literal[25] = [%.4097d]
  literal[26] = [%999999999999s]
  literal[27] = [%08x]
  literal[28] = [%%20d]
  literal[29] = [%%20n]
  literal[30] = [%%20x]
  literal[31] = [%%20s]

```

```

literal[32] = [%#0123456x%08x%x%s%p%d%n%o%u%c%h%l%q%j%z%Z%t%i%e%g%f%a%C%S%08x%%]
literal[33] = [../etc]
literal[34] = [../../../../etc]
literal[35] = [../../../../etc]
literal[36] = [../../../../etc]
literal[37] = [../../../../etc]
literal[38] = [../../../../etc]
literal[39] = [../../../../etc]
literal[40] = [../etc]
literal[41] = [../../../../etc]
literal[42] = [../../../../etc]
literal[43] = [../../../../etc]
literal[44] = [../../../../etc]
literal[45] = [../../../../etc]
literal[46] = [../../../../etc]
literal[47] = [../etc/]
literal[48] = [../../../../etc/]
literal[49] = [../../../../etc/]
literal[50] = [../../../../etc/]
literal[51] = [../../../../etc/]
literal[52] = [../../../../etc/]
literal[53] = [../../../../etc/]
literal[54] = [../etc/]
literal[55] = [../../../../etc/]
literal[56] = [../../../../etc/]
literal[57] = [../../../../etc/]
literal[58] = [../../../../etc/]
literal[59] = [../../../../etc/]
literal[60] = [../../../../etc/]
literal[61] = [../etc/passwd]
literal[62] = [../../../../etc/passwd]
literal[63] = [../../../../etc/passwd]
literal[64] = [../../../../etc/passwd]
literal[65] = [../../../../etc/passwd]
literal[66] = [../../../../etc/passwd]
literal[67] = [../../../../etc/passwd]
literal[68] = [../etc/passwd]
literal[69] = [../../../../etc/passwd]
literal[70] = [../../../../etc/passwd]
literal[71] = [../../../../etc/passwd]
literal[72] = [../../../../etc/passwd]
literal[73] = [../../../../etc/passwd]
literal[74] = [../../../../etc/passwd]
sequence[1] = [%n]
sequence[2] = [%f]
sequence[3] = [%%n]
sequence[4] = [A]
sequence[5] = [Ø]
sequence[6] = [Ø]
sequence[7] = [Ø]
sequence[8] = [%n]
sequence[9] = [%#123456x]
sequence[10] = [%s]
sequence[11] = [%%s]
sequence[12] = [%20s]
sequence[13] = [%%20s]
sequence[14] = [%20x]
sequence[15] = [%%20x]
sequence[16] = [%n%n%n%n%n]
sequence[17] = [%p%p%p%p%p]
sequence[18] = [%x%x%x%x%x]
sequence[19] = [%d%d%d%d%d]
sequence[20] = [%s%s%s%s%s]
sequence[21] = [%s%p%x%d]
sequence[22] = [%.1024d]
sequence[23] = [%.1025d]
sequence[24] = [%.2048d]
sequence[25] = [%.2049d]
sequence[26] = [%.4096d]
sequence[27] = [%.4097d]
sequence[28] = [%999999999999s]
sequence[29] = [%08x]
sequence[30] = [%%20d]
sequence[31] = [%%20n]
sequence[32] = [%%20x]

```

```

sequence[33] = [%%20s]
sequence[34] = [#0123456x%08x%x%sp%d%n%o%u%c%h%l%q%j%z%Z%t%i%e%g%f%a%C%S%08x%]
[17:17:30] info: beginning fuzz - method: tcp, config from: [/usr/share/sfuzz-db/basic.http], out:
[172.20.156.142:80]
[17:17:30] attempting fuzz - 1 (len: 18).
[17:17:30] info: tx fuzz - (18 bytes) - scanning for reply.
[17:17:31] read:
HTTP/1.1 302 Found
Server: nginx/1.16.1
Date: Sun, 13 Sep 2020 22:17:30 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
Set-Cookie: PHPSESSID=4hn65esk22gmbq27ag8q7ivqo1; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=4hn65esk22gmbq27ag8q7ivqo1; path=/; HttpOnly
Set-Cookie: security=impossible; HttpOnly
Location: login.php

```

```

=====
[17:17:31] attempting fuzz - 2 (len: 34).
[17:17:31] info: tx fuzz - (34 bytes) - scanning for reply.
[17:17:31] read:
HTTP/1.1 404 Not Found
Server: nginx/1.16.1
Date: Sun, 13 Sep 2020 22:17:31 GMT
Content-Type: text/html
Content-Length: 153
Connection: close

```

```

<html>
<head><title>404 Not Found</title></head>
<body>
<center><h1>404 Not Found</h1></center>
<hr><center>nginx/1.16.1</center>
</body>
</html>

```

```

=====
....

```

```

=====
[17:23:45] attempting fuzz - 1838 (len: 10059).
[17:23:45] info: tx fuzz - (10059 bytes) - scanning for reply.
[17:23:45] read:
HTTP/1.1 414 Request-URI Too Large
Server: nginx/1.16.1
Date: Sun, 13 Sep 2020 22:23:45 GMT
Content-Type: text/html
Content-Length: 177
Connection: close

```

```

<html>
<head><title>414 Request-URI Too Large</title></head>
<body>
<center><h1>414 Request-URI Too Large</h1></center>
<hr><center>nginx/1.16.1</center>
</body>
</html>

```

```

=====
[17:23:45] completed fuzzing.

```

```

real 6m14.481s
user 0m0.425s
sys 0m1.257s

```

This is good, but it holds nothing of value since NGINX is protecting the application, DVWA.



---

## 4. Conclusion

In this paper, I have shown how to run the tool `sfuzz` as well as where to find the config files for the attack sequences.

I am a long time advocate of Kali Linux for research purposes only on Private/Isolated networks. The `sfuzz` tool works according to the documentation. We can see from the output that it didn't do much for discovering vulnerabilities because of NGINX [our proxy server] sitting in front of the web server and preventing any kind of useful information gathering from `sfuzz`.

For fuzzing attacks, this tool right now is not something I will keep at the ready in my toolbelt of useful tools. I would recommend using DotDotPwn over this tool because in my 2020 Article 39, I have shown that `DotDotPwn` works effectively against the same target and found many vulnerabilities.

---

## 5. Appendix

### *References*

<https://tools.kali.org/vulnerability-analysis/sfuzz>

<https://gitlab.com/kalilinux/packages/sfuzz>