

How to scan DVWA with Kali Sparta

Introduction

The motivation for this paper is to show the user how to quickly get Sparta operational and scanning the DVWA running on a local instance. "SPARTA is a python GUI application that simplifies network infrastructure penetration testing by aiding the penetration tester in the scanning and enumeration phase. It allows the tester to save time by having point-and-click access to their toolkit and by displaying all tool output in a convenient way. If less time is spent setting up commands and tools, more time can be spent focusing on analysing results." -- tools.kali.org/information-gathering/sparta

Requirements

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, i.e. the Ubuntu user. 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
```

VirtualBox

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

The author is running on Ubuntu 17.04, 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.

Clean VirtualBox Networking

Run these two commands from a Terminal:

```
VBoxManage list natnetworks
```

```
VBoxManage list dhcpservers
```

Output :

```
NetworkName:    192.168.139-NAT
IP:             192.168.139.1
Network:        192.168.139.0/24
IPv6 Enabled:   No
IPv6 Prefix:    fd17:625c:f037:a88b::/64
DHCP Enabled:   Yes
Enabled:        Yes
loopback mappings (ipv4)
```

```
127.0.0.1=2
```

```
NetworkName: 192.168.139-NAT
IP:          192.168.139.3
NetworkMask: 255.255.255.0
lowerIPAddress: 192.168.139.101
upperIPAddress: 192.168.139.254
Enabled:     Yes
```

```
NetworkName: HostInterfaceNetworking-vboxnet0
IP:          172.20.0.3
NetworkMask: 255.255.255.0
lowerIPAddress: 172.20.0.101
upperIPAddress: 172.20.0.254
Enabled:     Yes
```

```
NetworkName: HostInterfaceNetworking-vboxnet1
IP:          0.0.0.0
NetworkMask: 0.0.0.0
lowerIPAddress: 0.0.0.0
upperIPAddress: 0.0.0.0
Enabled:     No
```

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.
```

```
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.
```

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
```

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.

Kali Linux

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/
```

Inside of the kali-linux-vm directory, populate a new file with the exact name, "Vagrantfile". Case matters, uppercase the "V".

Vagrantfile:

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

# Vagrantfile API/syntax version.
VAGRANTFILE_API_VERSION = "2"

Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
  config.vm.box = "Sliim/kali-2017.2-amd64"
  config.vm.box_version = "1"

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

  # For macbook/OSx systems, uncomment the line:
  #config.vm.network "public_network", bridge: "en0: Wi-Fi (AirPort)", auto_config: true

  config.vm.hostname = "kali-linux-vagrant"

  config.vm.provider "virtualbox" do |vb|
    vb.memory = "4096"
    vb.cpus = "3"
    vb.gui = true
    vb.customize ["modifyvm", :id, "--cpuexecutioncap", "95"]
    vb.customize ["modifyvm", :id, "--vram", "32"]
    vb.customize ["modifyvm", :id, "--accelerate3d", "on"]
    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
end
```

Save and write this file.

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 image and start the virtual machine.

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
```

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.

Damn Vulnerable Web Application (DVWA)

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

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

Inside of the dvwa-linux-vm directory, populate a new file with the exact name, “Vagrantfile”. Case matters, uppercase the “V”.

Vagrantfile:

```
#
# setup local instance of Damn Vulnerable Web Application (DVWA):
#

# Vagrantfile API/syntax version. Don't touch unless you know what you're doing!
VAGRANTFILE_API_VERSION = "2"

Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|

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

  # For macbook/OSx systems, uncomment the line:
  #config.vm.network "public_network", bridge: "en0: Wi-Fi (AirPort)", auto_config: true
```

```

# uncomment the next line for Macbook/OSx systems, wireless :
# config.vm.network "public_network", bridge: "en0: Wi-Fi (AirPort)", auto_config: true

config.vm.provision :shell, path: "bootstrap.sh"
config.vm.hostname = "dvwa"

config.vm.provider "virtualbox" do |vb|
  vb.memory = "1024"
  vb.cpus = "2"
  vb.gui = false
  vb.customize ["modifyvm", :id, "--cpuexecutioncap", "95"]
  vb.customize ["modifyvm", :id, "--vram", "32"]
  vb.customize ["modifyvm", :id, "--accelerate3d", "on"]
  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
end

```

Save and write this file.

Inside of the dvwa-linux-vm directory, populate a new file with the exact name, "bootstrap.sh". Case matters, all lowercase.

bootstrap.sh (include the shebang in your file, the `#!/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_root'
MYSQL_dvwa_password='sunshine'
DVWA_admin_password='admin'
recaptcha_public_key='u8392ihj32k18hujalkshuil32'
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}';"

service mysql restart

```

```

}

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_dwva(){
    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.

```

From a Terminal, change directory to:

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

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

```
$ vagrant up
```

You will need the IP address from the new DVWA virtual machine.

Login with:

```
$ vagrant ssh
```

Then run:

```
$ ip a
```

Choose the second network adapter, it should look like:

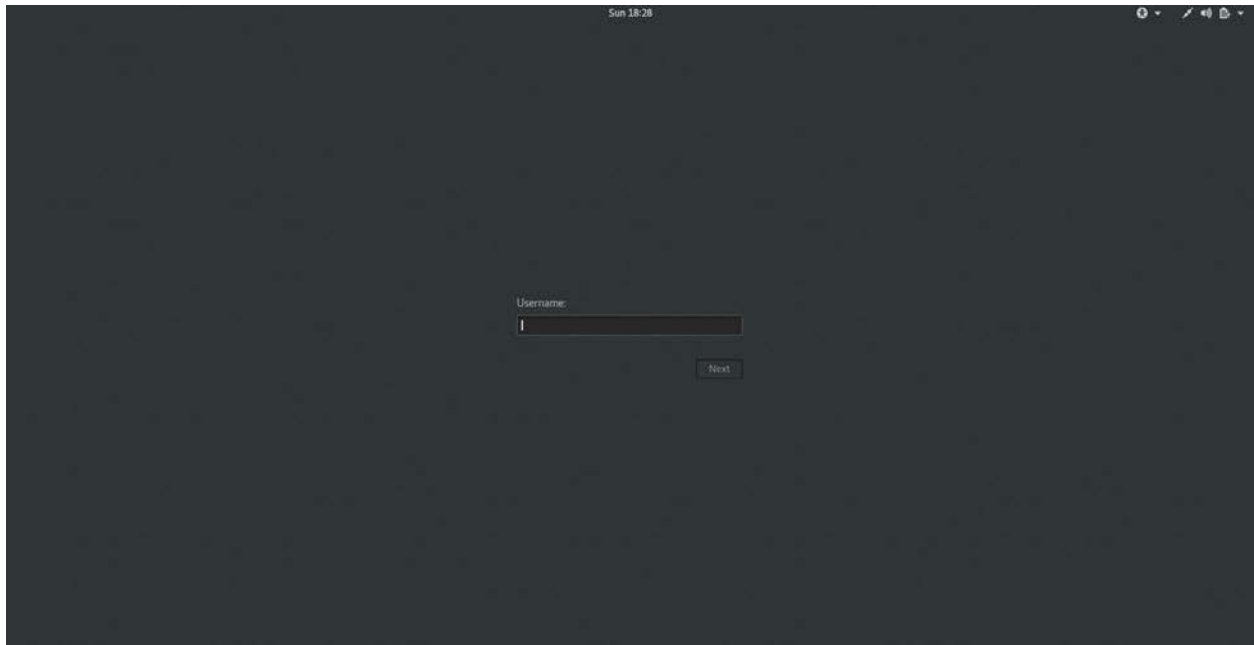
```
ubuntu@dvwa:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft 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 author's home wireless network uses 172.20.156.0/24 as the network range. 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.

Sparta (Kali Linux version, Beta)

First launch both Vagrant boxes for Kali-Linux and DVWA.

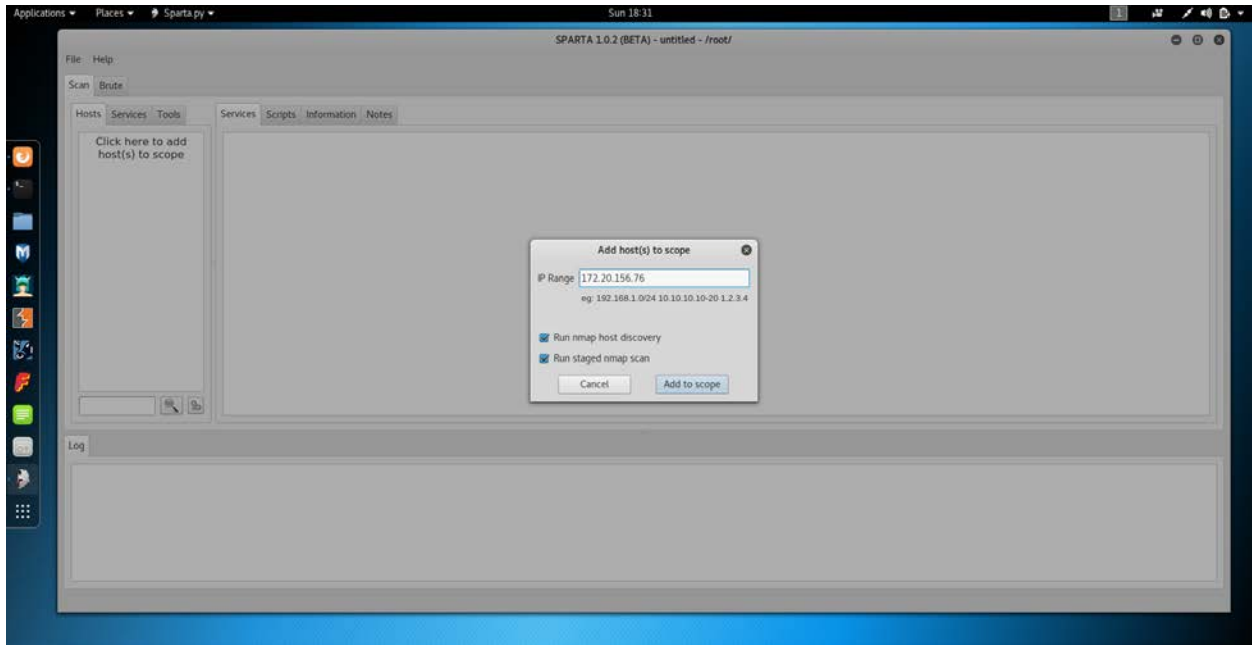
Then log into Kali-Linux with username: root and password:toor.



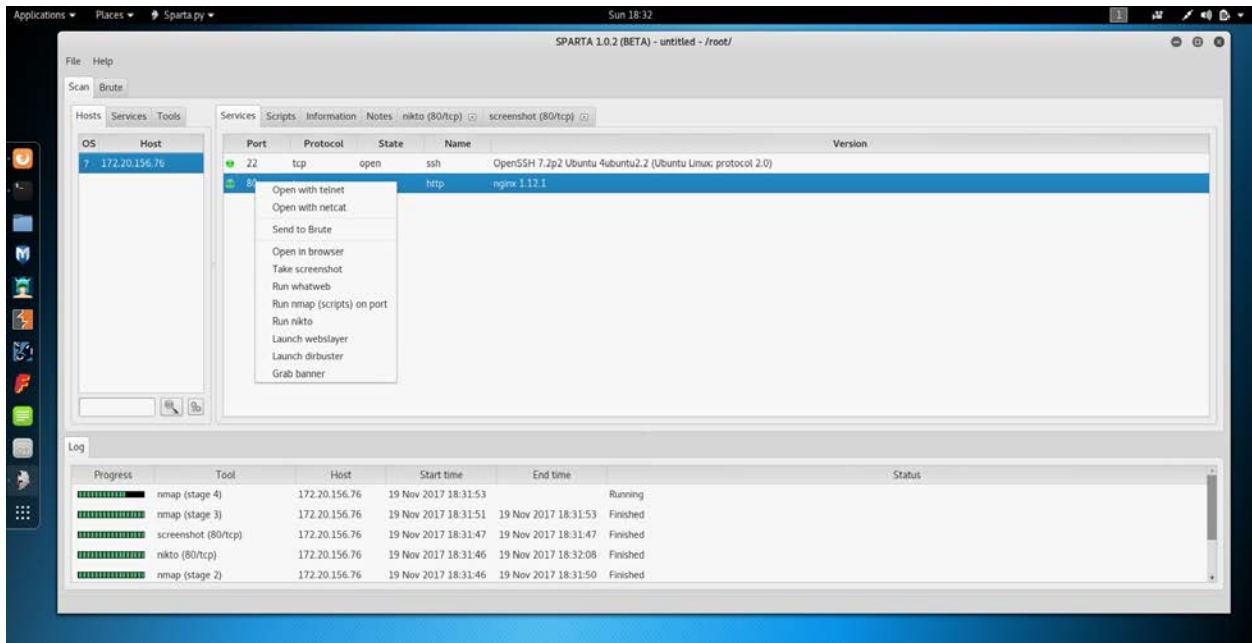
Next, from the upper menu, Application, open and go to 01 Information Gathering, and select Sparta.



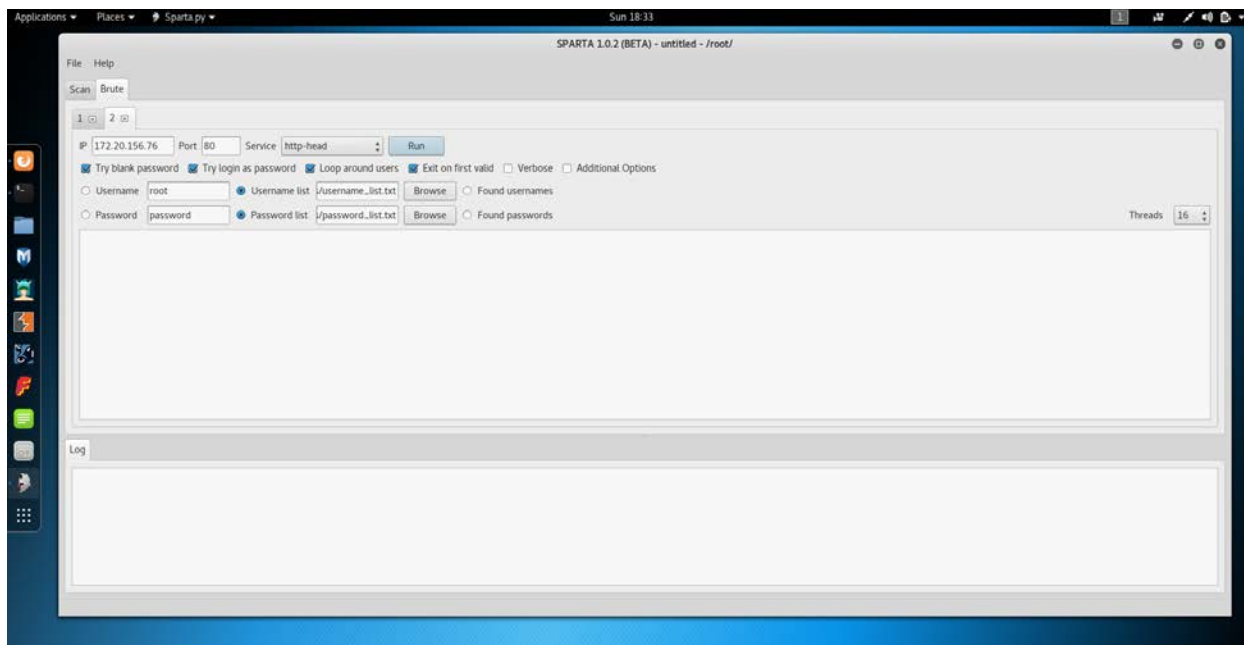
Some



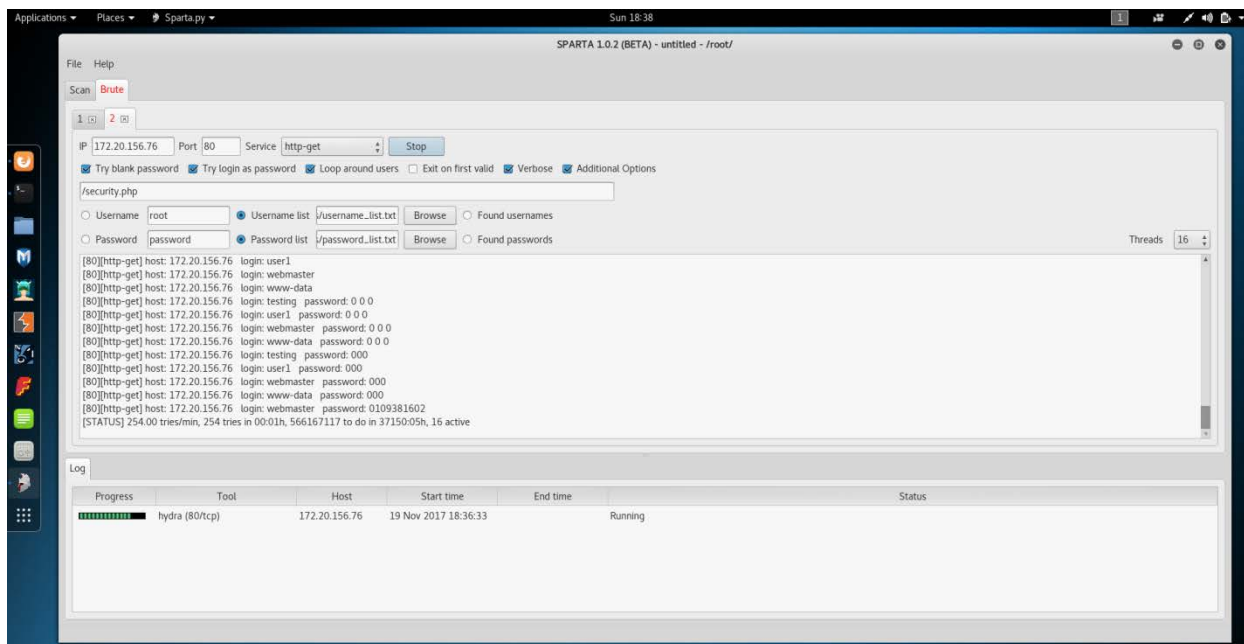
Above, in Hosts, click where it says, Click here..., which will bring up the above box, type in your target IP or range. Here we want to target single instance. Then click on Add to Scope button.



Above, we see TCP ports 22 and 80 open. Right-Click on the line with port 80, then select Send to Brute.



Uncheck the box with Exit on first valid, then change the method from http-head to http-get. Now change the radio buttons to Username list and Password list. The author recommends using the same lists as from the previous lab utilizing Hydra with Kali Linux. Now click on Run.



Above, we see the scan in progress with results of username and password combinations that worked. On an older laptop, this will take a very long time to finish the brute force attack. However, on a newer MacBook, model a1398, the same exact scan took less than 2 minutes to complete, using the same password_list.txt file. You can right click in the log on any line and instruct the command to be killed.

Conclusion

By following this guide, the user has installed VirtualBox, Vagrant, Kali-Linux, DVWA and ran a successful attack with Sparta against DVWA. All isolated on the local system.

What's beautiful about Sparta is that it is a wrapper for nmap, hydra, nikto and many other tools.

Features (from their homepage):

- Run nmap from SPARTA or import nmap XML output.
- Transparent staged nmap: get results quickly and achieve thorough coverage.
- Configurable context menu for each service. You can configure what to run on discovered services. Any tool that can be run from a terminal, can be run from SPARTA.
- You can run any script or tool on a service across all the hosts in scope, just with a click of the mouse.
- Define automated tasks for services (ie. Run nikto on every HTTP service, or sslscan on every ssl service).
- Default credentials check for most common services. Of course, this can also be configured to run automatically.
- Identify password reuse on the tested infrastructure. If any usernames/passwords are found by Hydra they are stored in internal wordlists which can then be used on other targets in the same network (breaking news: sysadmins reuse passwords).
- Ability to mark hosts that you have already worked on so that you don't waste time looking at them again.
- Website screenshot taker so that you don't waste time on less interesting web servers.

Sparta goes through and simplifies the coordination between these various tools and makes it easy to get to the later stage of having working credentials to get into the remote target. Make sure you use this tool with complete knowledge of the target you are attacking and the written legal permission to use this tool on any networks other than isolated on your laptop. You've been warned.