How to securely isolate and execute Nikto from Kali Linux

Introduction

The motivation behind this paper is to explore using the tool Nikto that comes with Kali Linux. Nikto is an open source web server scanning tool which performs remote tests against web servers for many items, including over 6000 potentially dangerous files/programs, checks for outdated versions of over 1200 servers, and version specific problems on over 250 servers. It also checks for server configuration items such as the presence of multiple index files, HTTP server options, and will attempt to identify installed web servers and software. Scan items and plugins are frequently updated and can be automatically updated. Nikto will test a web server in the quickest time possible, and is obvious in log files or to an IPS/ IDS. However, there is support for LibWhisker's anti-IDS methods in case you want to give it a try [or test your IDS system]. Not every check is a security related. There are some items that are "informational only" type checks that look for things that may not have a security flaw. These items are marked appropriately in the information printed.

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 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.

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 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/
```

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.

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

```
$os_update = <<SCRIPT
apt-get update
SCRIPT

$install_doona = <<SCRIPT
apt-get install -y doona
SCRIPT

# Vagrantfile API/syntax version.
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
  conf.vm.provision "shell", inline: $install_doona
  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, path: "bootstrap.sh"
  end
end
```

Save and write this file.

Inside of the kali-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 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_root'
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}"';"

    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_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.
#  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, you could use wget, i.e.
Make sure the bootstrap.sh file ends up in the same directory as the Vagrantfile.
I just googled for the string, "DVWA AND bootstrap.sh", case sensitive here with the AND.

`$ wget https://github.com/lookcrabs/DVWA-Vagrant/blob/master/bootstrap.sh`

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`

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 the second network adapter, it should look like:

```
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 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.

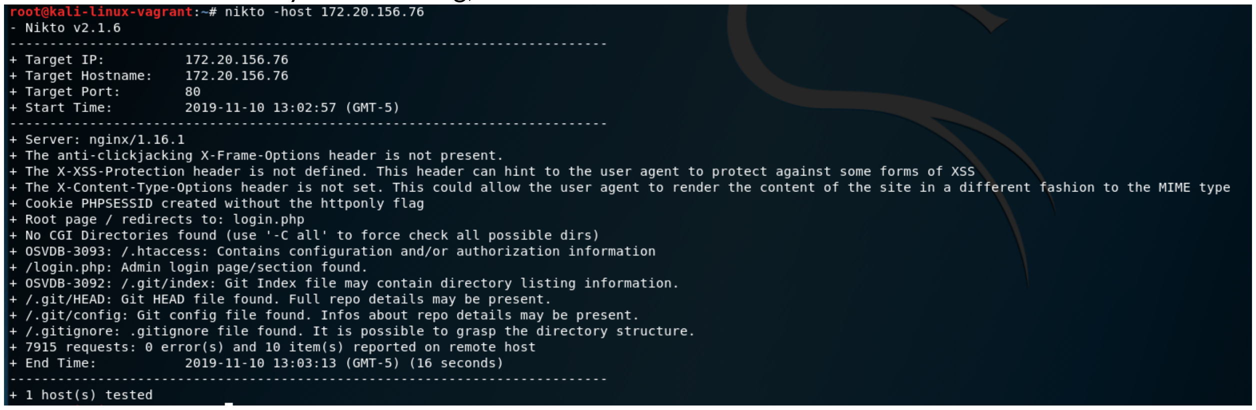
Nikto (Kali-Linux version)

Fire up both vagrant boxes of Kali-Linux and DVWA with vagrant up.
Login to kali linux. Username: root, Password: toor.

The author would highly recommend to either read the man page on Nikto or to run the base command with '-help' as a parameter, like:

```
# nikto -help
```

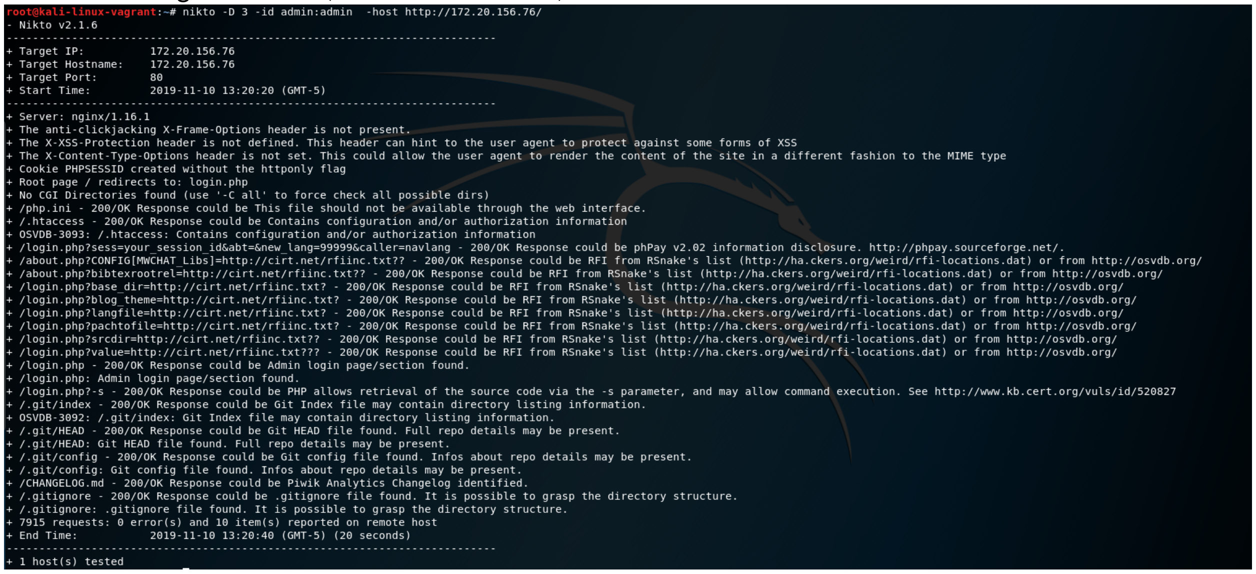
The most basic scan is by either adding, '-h <IP/hostname>' or '-host <IP/hostname>'.



```
root@kali-linux-vagrant:~# nikto -host 172.20.156.76
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          172.20.156.76
+ Target Hostname:    172.20.156.76
+ Target Port:        80
+ Start Time:         2019-11-10 13:02:57 (GMT-5)
---------------------------------------------------------------------------
+ Server: nginx/1.16.1
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ Cookie PHPSESSID created without the httponly flag
+ Root page / redirects to: login.php
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ OSVDB-3093: /.htaccess: Contains configuration and/or authorization information
+ /login.php: Admin login page/section found.
+ OSVDB-3092: /.git/index: Git Index file may contain directory listing information.
+ /.git/HEAD: Git HEAD file found. Full repo details may be present.
+ /.git/config: Git config file found. Infos about repo details may be present.
+ /.gitignore: .gitignore file found. It is possible to grasp the directory structure.
+ 7915 requests: 0 error(s) and 10 item(s) reported on remote host
+ End Time:           2019-11-10 13:03:13 (GMT-5) (16 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

Getting a little more advanced, I found that adding in the '-D' flag helps understand the output more. Initially I started with '-D v' and found the output was too much. I then modified this to '-D 3' to only see the HTTP 200 error codes, and this reduced the overall results to something much more digestible. Finally, there is an option for a username/password combo, as in: '-d username:password'.

In the following screenshot, we see the results, which took 20 seconds to scan.



```
root@kali-linux-vagrant:~# nikto -D 3 -id admin:admin  -host http://172.20.156.76/
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          172.20.156.76
+ Target Hostname:    172.20.156.76
+ Target Port:        80
+ Start Time:         2019-11-10 13:20:20 (GMT-5)
---------------------------------------------------------------------------
+ Server: nginx/1.16.1
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ Cookie PHPSESSID created without the httponly flag
+ Root page / redirects to: login.php
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ /php.ini - 200/OK Response could be This file should not be available through the web interface.
+ /.htaccess - 200/OK Response could be Contains configuration and/or authorization information
+ OSVDB-3093: /.htaccess: Contains configuration and/or authorization information
+ /login.php?sess=your_session_id&abt=&new_lang=99999&caller=navlang - 200/OK Response could be phPay v2.02 information disclosure. http://phpay.sourceforge.net/.
+ /about.php?CONFIG[MWCHAT_Libs]=http://cirt.net/rfiinc.txt?? - 200/OK Response could be RFI from RSnake's list (http://ha.ckers.org/weird/rfi-locations.dat) or from http://osvdb.org/
+ /about.php?bibtexrootrel=http://cirt.net/rfiinc.txt?? - 200/OK Response could be RFI from RSnake's list (http://ha.ckers.org/weird/rfi-locations.dat) or from http://osvdb.org/
+ /login.php?base_dir=http://cirt.net/rfiinc.txt? - 200/OK Response could be RFI from RSnake's list (http://ha.ckers.org/weird/rfi-locations.dat) or from http://osvdb.org/
+ /login.php?blog_theme=http://cirt.net/rfiinc.txt? - 200/OK Response could be RFI from RSnake's list (http://ha.ckers.org/weird/rfi-locations.dat) or from http://osvdb.org/
+ /login.php?langfile=http://cirt.net/rfiinc.txt? - 200/OK Response could be RFI from RSnake's list (http://ha.ckers.org/weird/rfi-locations.dat) or from http://osvdb.org/
+ /login.php?pachtofile=http://cirt.net/rfiinc.txt? - 200/OK Response could be RFI from RSnake's list (http://ha.ckers.org/weird/rfi-locations.dat) or from http://osvdb.org/
+ /login.php?srcdir=http://cirt.net/rfiinc.txt?? - 200/OK Response could be RFI from RSnake's list (http://ha.ckers.org/weird/rfi-locations.dat) or from http://osvdb.org/
+ /login.php?value=http://cirt.net/rfiinc.txt??? - 200/OK Response could be RFI from RSnake's list (http://ha.ckers.org/weird/rfi-locations.dat) or from http://osvdb.org/
+ /login.php - 200/OK Response could be Admin login page/section found.
+ /login.php: Admin login page/section found.
+ /login.php?-s - 200/OK Response could be PHP allows retrieval of the source code via the -s parameter, and may allow command execution. See http://www.kb.cert.org/vuls/id/520827
+ /.git/index - 200/OK Response could be Git Index file may contain directory listing information.
+ OSVDB-3092: /.git/index: Git Index file may contain directory listing information.
+ /.git/HEAD - 200/OK Response could be Git HEAD file found. Full repo details may be present.
+ /.git/HEAD: Git HEAD file found. Full repo details may be present.
+ /.git/config - 200/OK Response could be Git config file found. Infos about repo details may be present.
+ /.git/config: Git config file found. Infos about repo details may be present.
+ /CHANGELOG.md - 200/OK Response could be Piwik Analytics Changelog identified.
+ /.gitignore - 200/OK Response could be .gitignore file found. It is possible to grasp the directory structure.
+ /.gitignore: .gitignore file found. It is possible to grasp the directory structure.
+ 7915 requests: 0 error(s) and 10 item(s) reported on remote host
+ End Time:           2019-11-10 13:20:40 (GMT-5) (20 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

That's it, go forth and scan away. Try changing the '-D 3' flag back to verbose with '-D v' and run.
I would recommend with:

```
# nikto -D v id admin:admin -host http://<ip|hostname/ | tee -a /tmp/foo
```

Then review the output file with:

```
# less /tmp/foo
```

Conclusion

By following this paper, you have setup a secure lab that is flexible enough to test many different configurations/tests. Nikto comes as part of the base install of Kali Linux, so you don't have to worry about downloading it or installing. The commands might seem basic, but a lot is going on here with the security scan. In finality, this is an open source tool that should be a basic part of your toolkit to understand and verify the security posture of your web servers.