How to securely isolate and execute Hosthunter from Kali Linux

Version 0.1, Last Updated: 2024-06-22

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

Table of Contents

Chapter 1. Introduction

This paper introduces HostHunter, a Python based security tool that helps identify and gather hostnames (the names assigned to devices on a network) from a text file with a large list of IP addresses. It uses simple methods to collect public information and actively scan to connect IP addresses with their corresponding hostnames. This is important because it shows the full range of potential vulnerabilities an organization might have. HostHunter makes it easy to review the results by allowing you to export the information in different formats, like CSV, TXT, or Nessus files for inclusion into other applications.

As the developer explains:

"A tool to efficiently discover and extract hostnames providing a large set of target IPv4 or IPv6 addresses. HostHunter utilises simple OSINT [Open-Source Intelligence] and active reconaissance techniques to map IP targets with virtual hostnames. This is especially useful for discovering the true attack surface of your organisation. Output can be generated in multiple formats including CSV, TXT or Nessus file formats." [Andreas_Georgiou]

This paper will explore how HostHunter works, its practical applications, and how it enhances security against cyber threats. Let us begin.

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 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.

Chapter 3. HostHunter

3.1. How does it work

HostHunter operates by combining Open-Source Intelligence (OSINT) with active reconnaissance techniques to identify and map hostnames associated with IP addresses. Its core functionality is designed to streamline the process of discovering the actual attack surface of a target network, which often extends beyond just IP addresses to include virtual hosts and subdomains. Below is a formal explanation of how HostHunter works:

3.1.1. Input Handling

HostHunter begins by taking a list of target IP addresses as input, typically provided in a text file. This list can contain both IPv4 and IPv6 addresses, representing the devices and services that may be publicly or internally exposed by the organization.

3.1.2. OSINT-based Hostname Discovery

HostHunter uses OSINT techniques to gather publicly available information about the given IP addresses. It searches for domain names, subdomains, and virtual hosts that may be associated with these IPs, pulling data from public sources such as DNS records, certificate transparency logs, or web crawlers.

3.1.3. Active Reconnaissance

In addition to OSINT, HostHunter performs active scanning of the IP addresses to identify any running services and virtual hosts that may not be

easily discoverable via passive methods. This involves probing the IP addresses using tools such as Nmap, DNS enumeration, or banner grabbing to detect services and match them to virtual hostnames.

3.1.4. Hostname Mapping

Once discovered, HostHunter maps the gathered hostnames to their corresponding IP addresses. This process creates a clear relationship between the IP address and any virtual hosts that might be running on it, providing a comprehensive view of the network infrastructure.

3.1.5. Output Generation

After completing the mapping process, HostHunter allows users to export the results in various formats, such as CSV, TXT, or Nessus file formats. These formats are chosen for their compatibility with other security tools, such as vulnerability scanners, making it easier to integrate the findings into broader security assessments.

3.1.6. Usage and Integration

HostHunter's simplicity and format flexibility make it easy to integrate into existing security workflows. It's particularly valuable for identifying hidden or misconfigured services that may have been overlooked, giving security teams a broader view of their attack surface.

By automating much of the reconnaissance and discovery process, HostHunter helps organizations gain a deeper understanding of their network exposure, thereby allowing them to mitigate risks more effectively.

3.2. Practical Application

Open a terminal in Kali Linux and run the following:

```
cd ${HOME}

git clone https://github.com/SpiderLabs/HostHunter.git

cd HostHunter
```

Before we continue, verify your version of python is at least 3.10 or above with the command: `python --version`.

Back in our terminal we need to install the dependencies:

```
python -m pip install -r requirements.txt
```

The help section for our tooling:

```
┌──(root㉿kali-linux-vagrant)-[~/HostHunter]
└─# python hosthunter.py -h
usage: hosthunter.py [-h] [-f FORMAT] [-o OUTPUT] [-t TARGET] [-g GRAB] [-v] [-
V] [-d] [targets]

[?] HostHunter v2.0 - Help Page

positional arguments:
  targets               Sets the path of the target IPs file.

options:
  -h, --help            show this help message and exit
  -f FORMAT, --format FORMAT
                        Choose between .CSV, .TXT, Nessus output file formats.
  -o OUTPUT, --output OUTPUT
                        Sets the path of the output file.
  -t TARGET, --target TARGET
                        Hunt a Single IP.
  -g GRAB, --grab GRAB  Choose which SSL ports to actively scan. Default ports:
21/tcp, 25/tcp, 443/tcp, 993/tcp, 8443/tcp
  -v, --verify          Attempts to resolve IP Address
  -V, --version         Displays the current version.
```

```
  -d, --debug           Displays additional output and debugging information.

Author: Andreas Georgiou (@superhedgy)
```

To scan a single target, run [the IP address of the author's running DVWA]:

```
python hosthunter.py -f csv -o my_dvwa.csv -t 192.168.99.19
```

Output of scan:

```
....
 HostHunter:  v2.0
 Author :  Andreas Georgiou (@superhedgy)

|---------------------------------------------------------------------------
---------------------|

[+] IPv6 Hunting is Enabled

[+] Target: 192.168.99.19
[-] Hostnames: no results

|---------------------------------------------------------------------------
---------------------|

  Hunting Completed!

  Searched against 1 targets

  0 hostname was discovered in 0.82 sec

|---------------------------------------------------------------------------
---------------------|
```

File output:

```
-rw-r--r-- 1 root root     0 Jun 21 10:37 my_dvwa.csv.nessus
-rw-r--r-- 1 root root     0 Jun 21 10:37 my_dvwa.csv.vhosts
-rw-r--r-- 1 root root    79 Jun 21 10:37 my_dvwa.csv.vhosts.csv
-rw-r--r-- 1 root root     0 Jun 21 10:37 my_dvwa.csv.webapps
```

Check the vhosts file: `cat my_dvwa.csv.vhosts.csv`

```
"IP Address","Port/Protocol","Domains","Operating System","OS Version","Notes"
```

Not very helpful!

Create our list of reconaissance systems [that we wish to scan]:

```
printf '%s\n' 172.20.156.{3..255} > ./my_targets.txt
```

Finally to launch a scan of our target systems, we will run:

```
python hosthunter.py -f csv -o my_recon.csv -v ./my_targets.txt
```

Output:

```
....
 HostHunter:  v2.0
 Author :  Andreas Georgiou (@superhedgy)

|-------------------------------------------------------------------------------
---------------------|

[!] IPv6 Hunting is Disabled

[+] Target: 172.20.156.3
[-] Hostnames: no results

[+] Target: 172.20.156.4
time.fortress.lan

[+] Target: 172.20.156.5
ns5.fortress.lan

[+] Target: 172.20.156.6
ns6.fortress.lan

[+] Target: 172.20.156.7
ns7.fortress.local
```

```
[+] Target: 172.20.156.8
ns8.fortress.local

[+] Target: 172.20.156.9
[-] Hostnames: no results

....

[+] Target: 172.20.156.120
master01.fortress.lan

[+] Target: 172.20.156.121
master02.fortress.lan

[+] Target: 172.20.156.122
master03.fortress.lan

[+] Target: 172.20.156.123
[-] Hostnames: no results

....

[+] Target: 172.20.156.250
[-] Hostnames: no results

[+] Target: 172.20.156.251
[-] Hostnames: no results

[+] Target: 172.20.156.252
[-] Hostnames: no results

[+] Target: 172.20.156.253
[-] Hostnames: no results

[+] Target: 172.20.156.254
[-] Hostnames: no results

[+] Target: 172.20.156.255
[-] Hostnames: no results

|-------------------------------------------------------------------------------
---------------------|

  Hunting Completed!

  Searched against 253 targets

  1 hostnames were discovered in 3605.99 sec

|-------------------------------------------------------------------------------
---------------------|
```

Output of CSV: `cat my_recon.csv.vhosts.csv`

```
"IP Address","Port/Protocol","Domains","Operating System","OS Version","Notes"
"172.20.156.202","443/tcp","talos-dn20.fortress.lan","","",""
```

The author is a little bit disappointed on the CSV output. He was hoping for it to match the console output above.

The primary challenge with the author's results, as presented in the "Practical Application" section, is that the scope of his Homelab is insufficient to support the scale required for testing the thousands of services and applications discussed in the paper. This limitation raises ethical concerns, particularly around the violation of Non-Disclosure Agreements (NDAs) and the unwillingness of companies being scanned to have their results made public. These constraints restrict the ability to fully demonstrate the tool's capabilities in a real-world, large-scale environment.

The goal of these papers is to guide the audience in setting up Damn Vulnerable Web Application (DVWA), providing a target rich in vulnerabilities to demonstrate the use of Kali Linux tools. However, larger-scale environments would provide a more realistic context to sample a wider range of assets and better observe how the tools perform, including identifying any potential side effects. Such large-scale testing is crucial for assessing the full impact and behavior of these security tools, especially when considering deployment in real-world environments with significant target audiences.

3.3. How HostHunter enhances security against cybersecurity threats

HostHunter enhances security against cybersecurity threats by providing

organizations with critical insights into their actual attack surface. It does this by systematically discovering and mapping hidden or overlooked hostnames and virtual hosts associated with known IP addresses, which could otherwise remain undetected in conventional security assessments. Below is a formal description of how HostHunter improves an organization's defense posture against cyber threats:

3.3.1. Comprehensive Attack Surface Mapping

HostHunter goes beyond basic IP address scanning by identifying and mapping virtual hosts, subdomains, and associated services that might be running on the same IP address. For example, an organization may have a primary domain like example.com tied to a specific IP address, but HostHunter could also reveal subdomains such as mail.example.com (hosting an email server), dev.example.com (used for development and potentially running vulnerable or outdated software), and blog.example.com (hosting a content management system like WordPress). Each of these subdomains could expose different services—such as an outdated email server, an unpatched development environment, or an insecure web application—that would expand the organization's attack surface and require attention. By uncovering these hidden assets, security teams can gain a complete picture of the organization's true attack surface, which includes potentially vulnerable systems that were not initially identified.

3.3.2. Exposure of Misconfigurations and Shadow IT

Organizations often have services running under virtual hosts or subdomains that are not well-documented, leading to "shadow IT." For example, HostHunter might uncover an old subdomain like legacy.example.com that is still publicly accessible but no longer maintained.

This subdomain could be running outdated software, such as an unpatched version of Apache or a legacy web application with known vulnerabilities. HostHunter could also reveal that this subdomain is using insecure protocols, such as HTTP instead of HTTPS, or that it lacks proper authentication mechanisms, allowing unauthorized access. By identifying these vulnerable assets, the tool enables security teams to take corrective actions, such as updating software, securing protocols, or decommissioning the legacy service, before attackers can exploit these weaknesses.

3.3.3. Proactive Risk Identification

The tool's ability to link IP addresses with virtual hostnames provides valuable insight into potential vulnerabilities, allowing organizations to anticipate and address risks before they become security incidents. This proactive approach significantly reduces the likelihood of successful cyberattacks by identifying potential threats early in the security assessment process.

3.3.4. Enhanced Vulnerability Management

By exporting results in formats compatible with popular vulnerability scanning tools (such as Nessus), HostHunter integrates seamlessly into existing security workflows. This allows security teams to quickly follow up with more detailed vulnerability scans or penetration testing on the discovered assets, improving the efficiency and thoroughness of vulnerability management programs.

3.3.5. Identification of Weaknesses in Web Infrastructure

Since many cyberattacks exploit weaknesses in web-facing services (such as

outdated web servers, insecure web applications, or unpatched software), HostHunter's ability to identify virtual hosts and web-related assets strengthens the organization's web security. It highlights weak points in the infrastructure that are susceptible to common attacks like cross-site scripting (XSS), SQL injection, or remote code execution.

3.3.6. Improved Incident Response and Threat Intelligence

HostHunter's insights into the network and its assets enhance an organization's incident response capabilities by providing detailed information on the infrastructure. This information aids in threat intelligence efforts, helping to correlate attack patterns or indicators of compromise (IOCs) with specific hosts or services, allowing teams to respond quickly and effectively to ongoing threats.

3.3.7. Support for Regulatory Compliance

By offering detailed visibility into all potential exposure points, HostHunter aids in ensuring compliance with security standards such as the General Data Protection Regulation (GDPR), PCI-DSS, or ISO/IEC 27001. Many regulatory frameworks require organizations to have a clear understanding of their attack surface and to secure all publicly accessible systems. HostHunter helps meet these requirements by ensuring that no asset is overlooked.

3.3.8. Mitigation of External Threats

Attackers often use automated tools to scan for unprotected or under-secured services, particularly those associated with poorly managed subdomains or virtual hosts. By identifying these services, HostHunter helps organizations

mitigate risks from external threat actors, reducing the chance that overlooked infrastructure will be exploited by malicious actors, including cybercriminals or state-sponsored attackers.

3.3.9. Why this matters

By uncovering hidden assets, improving risk awareness, and integrating with existing security tools and processes, HostHunter strengthens an organization's cybersecurity defenses. Its comprehensive approach to mapping the attack surface enables proactive identification and mitigation of vulnerabilities, ensuring that organizations are better prepared to defend against evolving cyber threats.

3.3.10. Value to the organization

HostHunter provides significant value to an organization's cybersecurity efforts by enhancing proactive risk identification. One of the key strengths of HostHunter lies in its ability to link IP addresses with virtual hostnames, revealing potential vulnerabilities in systems that may not have been documented or properly monitored.

For example, many organizations use a mix of internal and third-party services on the same IP infrastructure, but may not actively track all subdomains and associated services, particularly those related to testing or temporary deployments [the Author is guilty of this on more than one occasion]. HostHunter identifies these shadow IT assets—such as development and test servers, misconfigured databases, or outdated web applications — before they become potential targets for attackers. This proactive identification allows security teams to address risks early, reducing the chances of successful attacks by mitigating vulnerabilities long before they can be exploited in the wild. In this way, HostHunter helps

reduce the overall attack surface by uncovering both legacy and shadow IT that would otherwise go unnoticed and undocumented [simply check the results into git or CVS].

In addition to improving risk identification, HostHunter enhances vulnerability management, offering seamless integration with popular tools like Nessus. The tool's export capabilities allow organizations to quickly perform vulnerability scans or penetration tests on the identified assets, ensuring no gaps are left untested. For example, once HostHunter discovers vulnerable subdomains or outdated software services, the data can be directly imported into scanning tools [e.g. Nessus], enabling a more focused and efficient assessment process. This direct integration streamlines the workflow, making it easier for security teams to follow up on discovered risks. HostHunter not only helps identify these risks but also provides the infrastructure for security teams to act on them immediately, leading to faster resolution of security gaps. As a result, organizations can improve their vulnerability management programs, which helps prevent exploitation and reinforces overall security.

3.4. Tools similar to hosthunter

There are several tools that provide similar functionality to HostHunter for gathering information about hosts, domain names, or subdomains. Some of these alternatives include:

3.4.1. Sublist3r:

A tool designed to enumerate subdomains using various search engines like Google, Yahoo, Bing, and others. It's commonly used for subdomain enumeration.

3.4.2. Amass:

An advanced open-source tool for network mapping of attack surfaces, including subdomain enumeration, DNS resolution, and other reconnaissance tasks.

3.4.3. Recon-ng:

A powerful reconnaissance tool that provides various modules to gather information about domains, including host enumeration, subdomain discovery, and other forms of reconnaissance.

3.4.4. theHarvester:

A tool for gathering emails, subdomains, hosts, and more from public sources such as search engines and PGP key servers. It's useful for gathering open-source intelligence (OSINT).

3.4.5. Assetfinder:

A small and simple tool to find related domains and subdomains through a variety of sources, including search engines and APIs.

3.4.6. DNSDumpster:

An online tool, but it can be scripted into Kali Linux, used for performing DNS reconnaissance and discovering hosts related to a domain.

3.4.7. Aquatone:

A tool for visual reconnaissance of websites across a domain, which also

includes subdomain discovery.

3.4.8. MassDNS:

A high-performance DNS resolver that is useful in enumerating a large number of subdomains and resolving them efficiently.

These tools complement each other and provide various ways to gather host and domain-related information during reconnaissance tasks in penetration testing or ethical hacking.

Chapter 4. Conclusion

HostHunter is a robust and invaluable tool for internal use within organizations, providing comprehensive visibility into an often-underestimated aspect of cybersecurity: the true attack surface. By combining OSINT techniques with active reconnaissance, HostHunter uncovers not only the IP addresses associated with an organization's network but also the virtual hosts, subdomains, and services that could be targets of cyberattacks. This expanded visibility allows security teams to identify and mitigate risks that may otherwise go unnoticed in conventional security assessments, such as shadow IT, misconfigurations, or insecure web services. The ability to map these assets in real-time and export the findings in various formats further enhances integration with vulnerability management and security operations, ensuring a thorough and proactive defense strategy.

Additionally, HostHunter's ability to automate the discovery process, uncover hidden vulnerabilities, and support seamless integration with popular scanning and reporting tools makes it an essential resource for organizations aiming to stay ahead of cyber threats. It not only enhances risk identification and incident response but also improves regulatory compliance by ensuring that no potential security risks are left unaddressed. By leveraging HostHunter, organizations can better protect themselves from both internal misconfigurations and external threats, significantly strengthening their overall cybersecurity posture.

Thank you for taking the time to read this white paper. As we conclude, we ask that you prepare for the end by metaphorically returning to your seat and securing any loose thoughts. Please fasten your mental seatbelt, ensuring your focus is in an upright and attentive position.

As you close this paper, we hope you retain the insights shared. The author sincerely appreciates your time and attention, knowing that you have countless options for reading material online. Thank you once again for choosing this white paper.

Chapter 5. Appendix

References

1. https://github.com/SpiderLabs/HostHunter
2. https://www.kali.org/tools/hosthunter/

Demo of the tool: https://asciinema.org/a/jp9B0IB6BzRAgbH3iNp7cCTpt