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

How to securely isolate and execute Netdiscover from Kali Linux

Version 0.1, Last Updated: 2nd Sep, 2023

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 . . . 4
    2.3. Vagrant . . . 4
    2.4. Kali Linux and Damn Vulnerable Web Application (DVWA) . . . 5
        2.4.1. Vagrantfile . . . 5
        2.4.2. bootstrap.sh . . . 7
3. Netdiscover . . . 13
    3.1. Default scan with a subnet range . . . 16
    3.2. Scan with subnet ranges in a file . . . 17
    3.3. Netdiscover Passive mode . . . 18
4. Conclusion . . . 19
5. Appendix . . . 20

Chapter 1. Introduction

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

Chapter 2. Requirements

2.1. Writing Conventions

If you see the following $ symbol on a command line to execute, what that means is that the command is executed as a regular user; meaning an account that does not have administrative privileges. Ignore the leading $ and execute the rest of the command.

```
$ command to execute as a regular user
```

If you see a command line lead with the # symbol, then that means that the command is executed as the root user. This implies you need to elevate to the root user before running the command, e.g. with: `sudo su  root`.

```
# command to execute as the root user
```

2.2. VirtualBox

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

The author is running on Ubuntu 18.04, so following to this URL: https://www.virtualbox.org/wiki/Linux_Downloads

For Ubuntu, double click on the .deb file, i.e. virtualbox-5.2_5.2.0-118431-Ubuntu-zesty_amd64.deb, and install VirtualBox on your local workstation.

2.2.1. Clean VirtualBox Networking

This section is here in case you already had virtualbox installed from before. The intent is to clean up the previous networking. If you do not need to do this, skip to Add VirtualBox Networking

Run these two commands from a Terminal:

```
$ VBoxManage list natnetworks
$ VBoxManage list dhcpservers
```

Output (example):

```
NetworkName:    192.168.139-NAT
IP:             192.168.139.1
Network:        192.168.139.0/24
IPv6 Enabled:   No
IPv6 Prefix:    fd17:625c:f037:2::/64
DHCP Enabled:   Yes
```

```
Enabled:        Yes
loopback mappings (ipv4)
        127.0.0.1=2


NetworkName:    192.168.139-NAT
Dhcpd IP:       192.168.139.3
LowerIPAddress: 192.168.139.101
UpperIPAddress: 192.168.139.254
NetworkMask:    255.255.255.0
Enabled:        Yes
Global Configuration:
    minLeaseTime:     default
    defaultLeaseTime: default
    maxLeaseTime:     default
    Forced options:   None
    Suppressed opts.: None
        1/legacy: 255.255.255.0
Groups:               None
Individual Configs:   None

NetworkName:    HostInterfaceNetworking-vboxnet0
Dhcpd IP:       172.20.0.3
LowerIPAddress: 172.20.0.101
UpperIPAddress: 172.20.0.254
NetworkMask:    255.255.255.0
Enabled:        Yes
Global Configuration:
    minLeaseTime:     default
    defaultLeaseTime: default
    maxLeaseTime:     default
    Forced options:   None
    Suppressed opts.: None
        1/legacy: 255.255.255.0
Groups:               None
Individual Configs:   None
```

Now, delete ALL of the pre-installed VirtualBox networks (one at a time following the syntax below):

```
VBoxManage natnetwork remove --netname <NetworkName_from_above>
VBoxManage natnetwork remove --netname 192.168.139-NAT
```

Repeat as many times as necessary to delete all of them.

Now, delete ALL of the pre-installed DHCP services:

```
VBoxManage dhcpserver remove --netname <DHCP_Server_NetworkName_from_above>
```

```
VBoxManage dhcpserver remove --netname 192.168.139-NAT
```

Repeat as many times as necessary to delete all of them.

2.2.2. Add VirtualBox Networking

Now, add the new VirtualBox networks so the Kali Linux guides work.

```
VBoxManage natnetwork add \
    --netname 192.168.139-NAT \
    --network "192.168.139.0/24" \
    --enable --dhcp on

VBoxManage dhcpserver add \
    --netname 192.168.139-NAT \
    --ip 192.168.139.3 \
    --lowerip 192.168.139.101 \
    --upperip 192.168.139.254 \
    --netmask 255.255.255.0 \
    --enable

VBoxManage hostonlyif create

VBoxManage hostonlyif ipconfig vboxnet0 \
    --ip 172.20.0.1 \
    --netmask 255.255.255.0

VBoxManage dhcpserver add \
    --ifname vboxnet0 \
    --ip 172.20.0.3 \
    --lowerip 172.20.0.101 \
    --upperip 172.20.0.254 \
    --netmask 255.255.255.0

VBoxManage dhcpserver modify \
    --ifname vboxnet0 \
    --enable
```

VirtualBox install complete.

2.3. Vagrant

Go to: https://www.vagrantup.com/downloads.html, follow the appropriate link to your OS and 32 or 64 bit version representing your local workstation. Download.

For Ubuntu, double click on the .deb file, i.e. vagrant_2.0.1_x86_64.deb, and install Vagrant on your local system.

NOTE Update vagrant vm: `vagrant box update`

2.4. Kali Linux and Damn Vulnerable Web Application (DVWA)

The author highly recommends to create a directory structure that is easy to navigate and find your code. As an example, you could use something similar to:

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

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

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

From a Terminal, change directory to:

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

2.4.1. Vagrantfile

Inside of the kali-linux-vm directory, populate a new file with the exact name, “Vagrantfile”. Case matters, uppercase the “V”. This file will contain both virtual machines for Kali Linux as well as setting up the DVWA virtual machine. Aggregating both virtual machines into one file has saved the author a lot of time. The coolness here is setting up the variables at the top of the Vagrantfile mimicing shell scripting inside of a virtual machine (passed in with provision: shell). I tested using: `apt-get update && apt-get upgrade -y`, but opted to take it out since it took over 45 minutes on my slower (old) hardware. See comment about downloading this file immediately preceding the code block.

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

$os_update = <<SCRIPT
apt-get update
SCRIPT

VAGRANTFILE_API_VERSION = "2"


Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
    config.vm.define "kali-linux-vagrant" do |conf|
        conf.vm.box = "kalilinux/rolling"

        # For Linux systems with the Wireless network, uncomment the line:
```

```
        conf.vm.network "public_network", bridge: "wlo1", auto_config: true

        # For macbook/OSx systems, uncomment the line and comment out the Linux
Wireless network:
        #conf.vm.network "public_network", bridge: "en0: Wi-Fi (AirPort)",
auto_config: true

        conf.vm.hostname = "kali-linux-vagrant"
        conf.vm.provider "virtualbox" do |vb|
            vb.gui = true
            vb.memory = "4096"
            vb.cpus = "2"
            vb.customize ["modifyvm", :id, "--vram", "32"]
            vb.customize ["modifyvm", :id, "--accelerate3d", "off"]
            vb.customize ["modifyvm", :id, "--ostype", "Debian_64"]
            vb.customize ["modifyvm", :id, "--boot1", "dvd"]
            vb.customize ["modifyvm", :id, "--boot2", "disk"]
            vb.customize ["modifyvm", :id, "--audio", "none"]
            vb.customize ["modifyvm", :id, "--clipboard", "hosttoguest"]
            vb.customize ["modifyvm", :id, "--draganddrop", "hosttoguest"]
            vb.customize ["modifyvm", :id, "--paravirtprovider", "kvm"]
        end
        conf.vm.provision "shell", inline: $os_update
    end

    config.vm.define "dvwa-vagrant" do |conf|

        conf.vm.box = "ubuntu/xenial64"

        conf.vm.hostname = "dvwa-vagrant"

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

        # For macbook/OSx systems, uncomment the line and comment out the Linux
Wireless network:
        #conf.vm.network "public_network", bridge: "en0: Wi-Fi (AirPort)",
auto_config: true

        config.vm.network "forwarded_port", guest: 80, host: 8080, auto_correct: true
        config.vm.network "forwarded_port", guest: 3306, host: 3306, auto_correct:
true

        conf.vm.provider "virtualbox" do |vb|
            vb.memory = "1024"
            vb.cpus = "2"
            vb.gui = false
            vb.customize ["modifyvm", :id, "--vram", "32"]
            vb.customize ["modifyvm", :id, "--accelerate3d", "off"]
            vb.customize ["modifyvm", :id, "--ostype", "Ubuntu_64"]
            vb.customize ["modifyvm", :id, "--boot1", "dvd"]
```

```
            vb.customize ["modifyvm", :id, "--boot2", "disk"]
            vb.customize ["modifyvm", :id, "--audio", "none"]
            vb.customize ["modifyvm", :id, "--clipboard", "hosttoguest"]
            vb.customize ["modifyvm", :id, "--draganddrop", "hosttoguest"]
            vb.customize ["modifyvm", :id, "--paravirtprovider", "kvm"]
        end
        conf.vm.provision "shell", inline: $os_update
        conf.vm.provision :shell, path: "bootstrap.sh"
    end
end
```

Save and write this file.

You can also download from:

```
$ curl -o Vagrantfile  http://securityhardening.com/files/Vagrantfile_20200928.txt
```

2.4.2. bootstrap.sh

Inside of the kali-linux-vm directory, populate a new file with the exact name, `bootstrap.sh`. Case matters, all lowercase. See comment about downloading this file immediately preceding the code block. `bootstrap.sh` (include the shebang in your file: the first line with `#!/usr/bin/env bash`):

```
#!/usr/bin/env bash
PHP_FPM_PATH_INI='/etc/php/7.0/fpm/php.ini'
PHP_FPM_POOL_CONF='/etc/php/7.0/fpm/pool.d/www.conf'
MYSQL_ROOT_PW='Assword12345'
MYSQL_dvwa_user='dvwa'
MYSQL_dvwa_password='sunshine'
DVWA_admin_password='admin'
recaptcha_public_key='u8392ihj32kl8hujalkshuil32'
recaptcha_private_key='89ry8932873832lih32ilj32'


install_base() {
    add-apt-repository -y ppa:nginx/stable
    sudo apt-get update
    sudo apt-get dist-upgrade -y
    sudo apt-get install -y \
        nginx \
        mariadb-server \
        mariadb-client \
        php \
        php-common \
        php-cgi \
        php-fpm \
        php-gd \
        php-cli \
```

```
        php-pear \
        php-mcrypt \
        php-mysql \
        php-gd \
        git \
        vim
}

config_mysql(){
    mysqladmin -u root password "${MYSQL_ROOT_PW}"
## Config the mysql config file for root so it doesn't prompt for password.
## Also sets pw in plain text for easy access.
## Don't forget to change the password here!!

cat <<EOF > /root/.my.cnf
[client]
user="root"
password="${MYSQL_ROOT_PW}"
EOF
    mysql -BNe "drop database if exists dvwa;"
    mysql -BNe "CREATE DATABASE dvwa;"
    mysql -BNe "GRANT ALL ON *.* TO '"${MYSQL_dvwa_user}"'@'localhost' IDENTIFIED BY
'"${MYSQL_dvwa_password}"';"

    systemctl enable mysql
    systemctl restart mysql
    sleep 2
}


config_php(){
    ## Config PHP FPM INI to disable some security settings:

    sed -i 's/^;cgi.fix_pathinfo.*$/cgi.fix_pathinfo = 0/g' ${PHP_FPM_PATH_INI}
    sed -i 's/allow_url_include = Off/allow_url_include = On/g' ${PHP_FPM_PATH_INI}
    sed -i 's/allow_url_fopen = Off/allow_url_fopen = On/g' ${PHP_FPM_PATH_INI}
    sed -i 's/safe_mode = On/safe_mode = Off/g' ${PHP_FPM_PATH_INI}
    echo "magic_quotes_gpc = Off" >> ${PHP_FPM_PATH_INI}
    sed -i 's/display_errors = Off/display_errors = On/g' ${PHP_FPM_PATH_INI}

    ## explicitly set pool options
    ## (these are defaults in ubuntu 16.04 so i'm commenting them out.
    ## If they are not defaults for you try uncommenting these)
    #sed -i 's/^;security.limit_extensions.*$/security.limit_extensions = \
    #.php .php3 .php4 .php5 .php7/g' /etc/php/7.0/fpm/pool.d/www.conf
    #sed -i 's/^listen.owner.*$/listen.owner = www-data/g'
/etc/php/7.0/fpm/pool.d/www.conf
    #sed -i 's/^listen.group.*$/listen.group = www-data/g'
/etc/php/7.0/fpm/pool.d/www.conf
    #sed -i 's/^;listen.mode.*$/listen.mode = 0660/g' /etc/php/7.0/fpm/pool.d/www.conf
```

```
    systemctl restart php7.0-fpm
}


config_nginx(){

cat << 'EOF' > /etc/nginx/sites-enabled/default
server
{
    listen  80;
    root /var/www/html;
    index index.php index.html index.htm;
    #server_name localhost
    location "/"
    {
        index index.php index.html index.htm;
        #try_files $uri $uri/ =404;
    }

    location ~ \.php$
    {
        include /etc/nginx/fastcgi_params;
        fastcgi_pass unix:/var/run/php/php7.0-fpm.sock;
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME $request_filename;
    }
}
EOF

    systemctl restart nginx
}


install_dvwa(){

    if [[ ! -d "/var/www/html" ]];
    then
          mkdir -p /var/www;
          ln -s /usr/share/nginx/html /var/www/html;
          chown -R www-data. /var/www/html;
    fi

    cd /var/www/html
    rm -rf /var/www/html/.[!.]*
    rm -rf /var/www/html/*
    git clone https://github.com/ethicalhack3r/DVWA.git ./
    chown -R www-data. ./
    cp config/config.inc.php.dist config/config.inc.php

    ### chmod uploads and log file to be writable by nobody
```

```
    chmod 777 ./hackable/uploads/
    chmod 777 ./external/phpids/0.6/lib/IDS/tmp/phpids_log.txt

    ## change the values in the config to match our setup (these are what you need to
update!
    sed -i '/db_user/ s/root/'${MYSQL_dvwa_user}'/'
/var/www/html/config/config.inc.php
    sed -i '/db_password/ s/p@ssw0rd/'${MYSQL_dvwa_password}'/'
/var/www/html/config/config.inc.php
    sed -i "/recaptcha_public_key/ s/''/'"${recaptcha_public_key}"'/"
/var/www/html/config/config.inc.php
    sed -i "/recaptcha_private_key/ s/''/'"${recaptcha_private_key}"'/"
/var/www/html/config/config.inc.php

}


update_mysql_user_pws(){
## The mysql passwords are set via /usr/share/nginx/html/dvwa/includes/DBMS/MySQL.php.
#  If you edit this every time they are reset it will reset to those.
#  Otherwise you can do a sql update statement to update them all (they are just md5's
of the string.
#  The issue is the users table doesn't get created until you click that button T_T to
init.

#mysql -BNe "UPDATE dvwa.users SET password = md5('YOUR_MYSQL_PW_HERE') WHERE user =
'admin';"
#mysql -BNe "UPDATE dvwa.users SET password = md5('YOUR_MYSQL_PW_HERE') WHERE user =
'gordonb';"
#mysql -BNe "UPDATE dvwa.users SET password = md5('YOUR_MYSQL_PW_HERE') WHERE user =
'1337';"
#mysql -BNe "UPDATE dvwa.users SET password = md5('YOUR_MYSQL_PW_HERE') WHERE user =
'pablo';"
#mysql -BNe "UPDATE dvwa.users SET password = md5('YOUR_MYSQL_PW_HERE') WHERE user =
'smithy';"

sed -i '/admin/ s/password/'${DVWA_admin_password}'/g'
/var/www/html/dvwa/includes/DBMS/MySQL.php
sed -i '/gordonb/ s/abc123/'${DVWA_admin_password}'/g'
/var/www/html/dvwa/includes/DBMS/MySQL.php
sed -i '/1337/ s/charley/'${DVWA_admin_password}'/g'
/var/www/html/dvwa/includes/DBMS/MySQL.php
sed -i '/pablo/ s/letmein/'${DVWA_admin_password}'/g'
/var/www/html/dvwa/includes/DBMS/MySQL.php
sed -i '/smithy/ s/password/'${DVWA_admin_password}'/g'
/var/www/html/dvwa/includes/DBMS/MySQL.php
}


install_base
config_mysql
```

```
install_dvwa
update_mysql_user_pws
config_php
config_nginx
```

Save and write this file.

If you have issues with copying and pasting the above file because code blocks in PDFs always copy correctly [NOT!], you could use curl, i.e. Make sure the bootstrap.sh file ends up in the same directory as the Vagrantfile.

```
$ curl -o bootstrap.sh  http://securityhardening.com/files/bootstrap_sh_20200928.txt
```

From a Terminal, change directory to:

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

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

```
$ vagrant up
```

This will download the appropriate images and start the virtual machines. Once running, through the VirtuaBox GUI, login as root. Password is “toor”, root backwards. Edit the following file: `/etc/ssh/sshd_config`

And change the line: `#PermitRootLogin prothibit-password` To: `PermitRootLogin yes` Meaning strip the comment out on the beginning of the line and alter `prohibit-password` to `yes`.

Then restart the ssh daemon:

```
# kill HUP $(pgrep sshd)
```

Notice, you are on a Bridged adapter, this will open the instance to allow root to ssh in with the most unsecure password in the world. Only make this change (allowing root to login via SSH) if you require root SSH access. You can change the root user’s password, which is highly recommended.

For the DVWA instance, I would first run ‘vagrant status’ to capture the name that vagrant is using for the running instance.

```
# vagrant status
```

Choose

```
Current machine states:
```

```
kali-linux-vagrant running (virtualbox)
dvwa-vagrant running (virtualbox)
```

This environment represents multiple VMs. The VMs are all listed above with their current state. For more information about a specific VM, run `vagrant status NAME`.

From there, log into the DVWA instance with:

```
$ vagrant ssh dvwa-vagrant
```

And then get the current IP address.

```
$ ip a
```

Choose the second network adapter, it should look like:

```
ubuntu@dvwa:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen
1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group
default qlen 1000
    link/ether 02:53:17:3c:de:80 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global enp0s3
        valid_lft forever preferred_lft forever
    inet6 fe80::53:17ff:fe3c:de80/64 scope link
        valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group
default qlen 1000
    link/ether 08:00:27:f0:77:2d brd ff:ff:ff:ff:ff:ff
    inet 172.20.156.76/24 brd 172.20.156.255 scope global enp0s8
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fef0:772d/64 scope link
        valid_lft forever preferred_lft forever
```

The test network used for this paper uses 172.20.156.0/24 as the network range [shown here in section 3]. Therefore, the adapter, enp0s8 is what he is looking for. The IP to use as a target is 172.20.156.76. Write down your value.

NOTE: For this paper, I changed the network adapter on the Kali VM from Host-Only to a Bridged Adapter so that I can scan my Homelab and test my Firewall's configuration.

Chapter 3. Netdiscover

Let's see what the man page says.

```
netdiscover(8)                                          active/passive ARP
reconnaissance tool                                         netdiscover(8)

NAME
       netdiscover - active/passive ARP reconnaissance tool

SYNOPSIS
       netdiscover [-i device] [-r range | -l file | -p] [-m file] [-F filter]
                   [-s time] [-c count] [-n node] [-dfPLNS]

DESCRIPTION
       netdiscover  is an active/passive ARP reconnaissance tool, initially developed
to gain information about wireless networks without DHCP servers in wardriving
scenarios. It can also be used on switched networks. Built on top of libnet and
libpcap, it can passively detect online hosts or search for them by sending ARP
requests.

       Furthermore, it can be used to inspect your network's ARP traffic, or find
network addresses using auto scan mode, which will scan for common  local networks.

OPTIONS
       -i device
              The network interface to sniff and inject packets. If no interface is
specified, first available will be used.

       -r range
              Scan  a given range instead of auto scan. Valid range values area for
example: 192.168.0.0/24, 192.168.0.0/16 or 192.168.0.0/8.  Currently, acceptable
ranges are /8, /16 and /24 only.

       -l file
              Scan ranges contained on the given file. It must contain only one range
per line.

       -p     Enable passive mode. In passive mode, netdiscover does not send
anything, but does only sniff.

       -m file
              Scan a list of known MACs and host names.

       -F filter
              Customize pcap filter expression (default: "arp").

       -s time
              Sleep given time in milliseconds between each ARP request injection.
(default 1)
```

```
       -c count
              Number of times to send each ARP request. Useful for networks with
packet loss, so it will scan given times for each host. (default 1)

       -n node
              Last IP octet of the source IP used for scanning. You can change it if
the default host (x.x.x.67) is already used. (allowed range is 2 to 253, default 67)

       -d     Ignore  configuration files at home dir (for autoscan and fast mode
only). This will use default ranges and IPs for autoscan and fast mode. See below for
information about configuration files.

       -f     Enable fast mode scan. This will only scan for .1, .100 and .254 on each
network. This mode is useful  while  searching  for  ranges  being used. After you
found such range you can make a specific range scan to find online boxes.

       -P     Produces  an  output suitable to be redirected into a file or to be
parsed by another program, instead of using interactive mode.  Enabling this option,
netdiscover will stop after scanning given ranges.

       -L     Similar to -P but continue program execution to capture ARP packets
passively after the active scan.  phase to  capture  ARP  packets  passively.

       -N     Do not print header. Only valid when -P or -L is enabled.

       -S     (DEPRECATED)  Enable sleep time suppression between each request.  If
set, netdiscover will sleep after having scanned 255 hosts instead of sleeping after
each one. This mode was used in netdiscover 0.3 beta4 and before. Avoid this option in
networks  with  packet  loss,  or  in wireless networks with low signal level. (also
called hardcore mode)

USAGE
       If  passive  mode  (-p), scan list (-l) or scan range (-r) options aren't
enabled, netdiscover will scan for common LAN addresses (192.168.0.0/16, 172.16.0.0/12
and 10.0.0.0/8).

       Screen control keys:

           h  Show help screen.
           j  Scroll down (or down arrow).
           k  Scroll up (or up arrow).
           .  Scroll page up.
           ,  Scroll page down.
           q  Close help screen or end application.

       Screen views:

           a  Show ARP replies list.
           r  Show ARP requests list.
           u  Show unique hosts detected.
```

```
CONFIG FILES
       There are 2 configuration files that netdiscover will look for, each time it is
executed. If files doesn't exist,  netdiscover  will  use  default
       values. You can use the -d switch to disable reading and loading configuration
files.

       ~/.netdiscover/ranges
              This  file  contains  a  list of ranges (one per line) used for auto
scan mode instead of default ranges. By default netdiscover will use a
              list of common ranges used on local networks.

       Example:

           192.168.21.0/24
           172.26.0.0/16
           10.0.0.0/8

       ~/.netdiscover/fastips
              List containing the last octet of the IPs to be scanned on each subnet,
when using fast mode (-f), by default (1,100,154). You must  put  a
              number per line.

USAGE EXAMPLES
       Scan common LAN addresses on eth0:

           # netdiscover -i eth0

       Fast scan common LAN addresses on eth0 (search only for gateways):

           # netdiscover -i eth0 -f

       Scan some fixed ranges:

           # netdiscover -i eth0 -r 172.26.0.0/24
           # netdiscover -r 192.168.0.0/16
           # netdiscover -r 10.0.0.0/8

       Scan common LAN addresses with sleep time 0.5 milliseconds instead of default
1:

           # netdiscover -s 0.5

       Scan fixed range on fast mode with sleep time 0.5 milliseconds instead of
default 1:

           # netdiscover -r 192.168.0.0/16 -f -s 0.5

       Scan a range using 101 as last octet for SOURCE IP

           # netdiscover -r 10.1.0.0/16 -n 101
```

```
       Only sniff for ARP traffic, don't send nothing:

           # netdiscover -p

AUTHOR
       netdiscover was written by Jaime Penalba Estebanez <jpenalbae@gmail.com>.

       This  manual  page  was  originally written by Nicolas Weyland, for the Debian
project. This man page has been merged into netdiscover project and modified from the
original by Jaime Penalba and Joao Eriberto Mota Filho.

netdiscover-0.10                                                    29 Oct 2022
netdiscover(8)
```

Let's see what the help page says.

```
┌──(root㉿kali-linux-vagrant)-[~]
└─# netdiscover -help
Netdiscover 0.10 [Active/passive ARP reconnaissance tool]
Written by: Jaime Penalba <jpenalbae@gmail.com>

Usage: netdiscover [-i device] [-r range | -l file | -p] [-m file] [-F filter] [-s
time] [-c count] [-n node] [-dfPLNS]
  -i device: your network device
  -r range: scan a given range instead of auto scan. 192.168.6.0/24,/16,/8
  -l file: scan the list of ranges contained into the given file
  -p passive mode: do not send anything, only sniff
  -m file: scan a list of known MACs and host names
  -F filter: customize pcap filter expression (default: "arp")
  -s time: time to sleep between each ARP request (milliseconds)
  -c count: number of times to send each ARP request (for nets with packet loss)
  -n node: last source IP octet used for scanning (from 2 to 253)
  -d ignore home config files for autoscan and fast mode
  -f enable fastmode scan, saves a lot of time, recommended for auto
  -P print results in a format suitable for parsing by another program and stop after
active scan
  -L similar to -P but continue listening after the active scan is completed
  -N Do not print header. Only valid when -P or -L is enabled.
  -S enable sleep time suppression between each request (hardcore mode)

If -r, -l or -p are not enabled, netdiscover will scan for common LAN addresses.
```

3.1. Default scan with a subnet range

We can quickly run a scan against a subnet with the `-r` option. As the help page shows above, this can be targeted against Classless Inter Domain Routing (CIDR) with the exact sizes of: `/24`, `/16`, and `/8`.

```
┌──(root㉿kali-linux-vagrant)-[~]
└─# netdiscover -i eth1 -r 192.168.99.0/24
```

Output:

```
Currently scanning: Finished!   |   Screen View: Unique Hosts

5 Captured ARP Req/Rep packets, from 5 hosts.   Total size: 300
_____________________________________________________________________________
  IP            At MAC Address     Count     Len  MAC Vendor / Hostname
-----------------------------------------------------------------------------
192.168.99.1    60:22:32:28:cf:bf      1      60  Ubiquiti Inc
192.168.99.2    60:22:32:28:cf:bf      1      60  Ubiquiti Inc
192.168.99.188  24:77:03:c4:7d:2c      1      60  Intel Corporate
192.168.99.253  76:7c:f2:43:32:87      1      60  Unknown vendor
192.168.99.209  68:5d:43:3f:7c:f8      1      60  Intel Corporate
```

3.2. Scan with subnet ranges in a file

For testing purposes only, I created a file with all of my valid local subnets to see what the tool could find.

```
┌──(root㉿kali-linux-vagrant)-[~]
└─# cat ranges.txt
192.168.42.0/24
192.168.43.0/24
192.168.55.0/24
192.168.77.0/24
192.168.99.0/24
172.20.156.0/24
172.20.157.0/24
172.31.254.0/24
```

And then ran the scan with the following options:

```
┌──(root㉿kali-linux-vagrant)-[~]
└─# netdiscover -i eth1 -l ./ranges.txt
```

Output:

```
Currently scanning: Finished!   |   Screen View: Unique Hosts

5 Captured ARP Req/Rep packets, from 5 hosts.   Total size: 300
_____________________________________________________________________________
  IP            At MAC Address     Count     Len  MAC Vendor / Hostname
```

```
-----------------------------------------------------------------------------
192.168.99.1    60:22:32:28:cf:bf      1      60  Ubiquiti Inc
192.168.99.2    60:22:32:28:cf:bf      1      60  Ubiquiti Inc
192.168.99.188  24:77:03:c4:7d:2c      1      60  Intel Corporate
192.168.99.253  76:7c:f2:43:32:87      1      60  Unknown vendor
192.168.99.209  68:5d:43:3f:7c:f8      1      60  Intel Corporate
```

What I like about this a lot is this tool just validated the 33 Firewall rules I set up in my UniFi UDM's Firewall. I can see traffic on the local network only, but routing to the other subnets is prohibited.

3.3. Netdiscover Passive mode

With the `-p` option, we can run the tool in passive mode, where it does not actively send packets outbound.

```
┌────(root kali-linux-vagrant)-[~]
└──# netdiscover -i eth1 -p
```

Output:

```
Currently scanning: (passive)   |   Screen View: Unique Hosts

4 Captured ARP Req/Rep packets, from 2 hosts.   Total size: 240
-----------------------------------------------------------------------------
  IP            At MAC Address     Count     Len  MAC Vendor / Hostname
-----------------------------------------------------------------------------
192.168.99.209  68:5d:43:3f:7c:f8      3     180  Intel Corporate
192.168.99.1    60:22:32:28:cf:bf      1      60  Ubiquiti Inc
```

We can see the current host, with the last octet of `.209` as well as the router with the last octet of `.1`.

Chapter 4. Conclusion

We have gone through and shown the basic usage on this tool and some working examples. The results are real world and showing the Author's Homelab. Netdiscover is what I consider an enumerating tool, in that a network person can leverage this to understand exactly what is on a network.

"Netdiscover is an active/passive address reconnaissance tool, mainly developed for those wireless networks without dhcp server, when you are wardriving. It can be also used on hub/switched networks.

Built on top of libnet and libpcap, it can passively detect online hosts, or search for them, by actively sending ARP requests.

Netdiscover can also be used to inspect your network ARP traffic, or find network addresses using auto scan mode, which will scan for common local networks.

Netdiscover uses the OUI table to show the vendor of each MAC address discovered and is very useful for security checks or in pentests." — Netdiscover page, Kali Linux Tools

The author is re-stating the developer's original messaging because it drives the intent and purpose of this tool. It is meant to enumerate both Wireless and Wired networks and does that one job very well. The only quirk the author could find in the tool is that it does NOT handle CIDR addresses outside of [`/24`, `/16`, `/8`], meaning as a user, you get to expand your subnets that you wish to scan.

E.g. Instead of:

```
172.20.156.0/23
```

You get to expand that to:

```
172.20.156.0/24
172.20.157.0/24
```

The `Netdiscover` tool is another useful tool to add to your toolbelt when you need to quickly discover what is on a subnet and don't wish to pull out `nmap` or said is not available because of security restrictions at your local site.

Until next time, continue securing the network and Living your Life!

Chapter 5. Appendix

References

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

https://github.com/netdiscover-scanner/netdiscover

https://kalilinuxtutorials.com/netdiscover-scan-live-hosts-network/