Kali Linux and dnswalk

Version 0.1, Last Updated: 23 Oct 2021

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

Table of Contents

1. Introduction

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

"The DNSWalk tool is a Domain Name System (DNS) that is used to identify information about the complete list of IP addresses, as well as the corresponding hostnames, that are stored in a specific server. It works by using a DNS zone transfer. As the DNSWalk tool performs zone transfers of the specified domains, it checks the database in multiple ways for accuracy and consistency … One of the tools that cybersecurity teams use to determine if DNS zone transfer vulnerabilities exist for the server that corresponds to the domain name, is DNSWalk. If there are vulnerabilities, users are then able to obtain useful information to resolve them." **source:** https://niccs.cisa.gov/training/search/cybrary/how-use-dnswalk-bswr

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.

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.

3. dnswalk

If you have not done so, go ahead and run `vagrant up` from inside of your directory containing the Vagrantfile.

Log into Kali Linux.

username:root

password:toor

From what I found this year, the software certificate has expired.

Run the following commands as the root user to update the Kali linux software certificate:

```
cd /root/

curl -O https://archive.kali.org/archive-key.asc

apt-key add ./archive-key.asc
```

In a terminal as the root user, run:

```
apt udpate
apt upgrade -y

systemctl reboot
```

3.1. Install

Open a terminal and run the command:

```
sudo apt install dnswalk
```

3.2. Options

Listing the available options:

```
root@kali-linux-vagrant:~# dnswalk --help
/usr/bin/dnswalk version [unknown] calling Getopt::Std::getopts (version 1.12 [paranoid]),
running under Perl version 5.32.1.

Usage: dnswalk [-OPTIONS [-MORE_OPTIONS]] [--] [PROGRAM_ARG1 ...]

The following single-character options are accepted:
    With arguments: -D
    Boolean (without arguments): -r -f -i -a -d -m -F -l

Options may be merged together.  -- stops processing of options.
Space is not required between options and their arguments.
  [Now continuing due to backward compatibility and excessive paranoia.
   See 'perldoc Getopt::Std' about $Getopt::Std::STANDARD_HELP_VERSION.]
Usage: dnswalk domain
domain MUST end with a '.'
```

Explanation:

```
-r     Recursively descend sub-domains of the specified domain.
-a     Turn on warning of a duplicate A records.
-d     Print debugging and ‘status’ information to stderr.  (Use only if redirecting stdout).
-m     Perform checks only if the zone has been modified since the previous run.
-F     Perform  “fascist”  checking.  When checking an A record, compare the PTR name for each IP address with the
forward name and report mismatches.
-i     Suppress check for invalid characters in a domain name.
-l     Perform  “lame delegation”  checking.   For every NS record, check to see that the listed host is indeed
returning authoritative answers for this domain.
```

3.3. Usage

As shown above, the domain you test MUST end with a '.'

A fully-qualified (unambiguous) DNS domain names have a dot at the end. Administrators running DNS servers usually know this [if you miss the trailing dots out, your DNS configuration is most likely will not work] but the general public usually doesn’t. A domain name that doesn’t have a dot at the end is not fully-qualified and is potentially ambiguous. This was documented in the DNS specification, RFC 1034, way back in 1987:

```
Since a complete domain name ends with the root label, this leads to a printed form which ends in a dot. We use
this property to distinguish between:

- a character string which represents a complete domain name (often called "absolute"). For example,
"poneria.ISI.EDU."

- a character string that represents the starting labels of a domain name which is incomplete, and should be
completed by local software using knowledge of the local domain (often called "relative"). For example, "poneria"
used in the ISI.EDU domain.
```

3.4. My Target DNS service

Is running the DNS service Unbound.

Config file for said service (/etc/unbound/unbound.conf):

```
include: "/etc/unbound/unbound.conf.d/*.conf"

## Debian 10 unbound config
#
server:
  val-log-level: 2
  use-syslog: yes
  verbosity: 1
  #
  access-control: 127.0.0.0/8 allow
  access-control: 10.0.0.0/8 deny
  access-control: 192.168.0.0/16 deny
  #
  access-control: 172.16.0.0/20 deny
  access-control: 172.17.0.0/20 deny
  access-control: 172.18.0.0/20 deny
  access-control: 172.19.0.0/20 deny
  access-control: 172.20.0.0/17 deny
  access-control: 172.20.128.0/20 deny
  access-control: 172.20.144.0/21 deny
  access-control: 172.20.152.0/22 deny
  #
  access-control: 172.20.156.0/24 allow
  #
  access-control: 172.20.157.0/24 deny
  access-control: 172.20.158.0/23 deny
  access-control: 172.20.160.0/19 deny
  access-control: 172.20.192.0/18 deny
  access-control: 172.21.0.0/20 deny
```

```
  access-control: 172.22.0.0/20 deny
  access-control: 172.23.0.0/20 deny
  access-control: 172.24.0.0/20 deny
  access-control: 172.25.0.0/20 deny
  access-control: 172.26.0.0/20 deny
  access-control: 172.27.0.0/20 deny
  access-control: 172.28.0.0/20 deny
  access-control: 172.29.0.0/20 deny
  access-control: 172.30.0.0/20 deny
  access-control: 172.31.0.0/20 deny
  #
  aggressive-nsec: yes
  cache-max-ttl: 14400
  cache-min-ttl: 1200
  directory: /etc/unbound
  do-ip4: yes
  do-ip6: yes
  do-tcp: yes
  hide-identity: yes
  hide-version: yes
  interface: 0.0.0.0
  pidfile: /var/run/local_unbound.pid
  port: 53
  prefetch: yes
  rrset-roundrobin: yes
  so-reuseport: yes
  tls-cert-bundle: "/etc/ssl/certs/ca-certificates.crt"
  use-caps-for-id: yes
  username: unbound
  root-hints: "/etc/unbound/root.hints"
  num-threads: 2
  msg-cache-slabs: 4
  rrset-cache-slabs: 4
  infra-cache-slabs: 4
  key-cache-slabs: 4
  rrset-cache-size: 128m
  msg-cache-size: 32m
  so-rcvbuf: 1m

# my stuff for homelab
private-domain: "fortress.lan"
local-zone: "fortress.lan" static
local-data: "router.fortress.lan IN A 172.20.156.1"
local-data: "dns.fortress.lan IN A 172.20.156.5"
local-data: "* IN A 172.20.156.115"
local-data: "master01.fortress.lan IN A 172.20.156.120"
local-data: "master02.fortress.lan IN A 172.20.156.121"
local-data: "master03.fortress.lan IN A 172.20.156.122"
local-data: "infra01.fortress.lan IN A 172.20.156.123"
local-data: "infra02.fortress.lan IN A 172.20.156.124"
local-data: "infra03.fortress.lan IN A 172.20.156.125"
local-data: "node01.fortress.lan IN A 172.20.156.126"
local-data: "node02.fortress.lan IN A 172.20.156.127"
local-data: "node03.fortress.lan IN A 172.20.156.128"
local-data: "node04.fortress.lan IN A 172.20.156.129"
local-data: "node05.fortress.lan IN A 172.20.156.130"
local-data: "storage01.fortress.lan IN A 172.20.156.131"
local-data: "storage02.fortress.lan IN A 172.20.156.132"
local-data: "storage03.fortress.lan IN A 172.20.156.133"

local-data: "wmaster01.fortress.lan IN A 172.20.156.140"
local-data: "wmaster02.fortress.lan IN A 172.20.156.141"
local-data: "wmaster03.fortress.lan IN A 172.20.156.142"
local-data: "winfra01.fortress.lan IN A 172.20.156.143"
local-data: "winfra02.fortress.lan IN A 172.20.156.144"
local-data: "winfra03.fortress.lan IN A 172.20.156.145"
local-data: "wnode01.fortress.lan IN A 172.20.156.146"
local-data: "wnode02.fortress.lan IN A 172.20.156.147"
local-data: "wnode03.fortress.lan IN A 172.20.156.148"
local-data: "wnode04.fortress.lan IN A 172.20.156.149"
local-data: "wnode05.fortress.lan IN A 172.20.156.150"
local-data: "wstorage01.fortress.lan IN A 172.20.156.151"
local-data: "wstorage02.fortress.lan IN A 172.20.156.152"
local-data: "wstorage03.fortress.lan IN A 172.20.156.153"
```

```
# DNS blackhole
local-zone: "doubleclick.net" redirect  # Unbound from pkg built with libevent; increase threads and slabs to the
  # number of real cpu cores to reduce lock contention. Increase cache size to
  # store more records and allow each thread to serve an increased number of
  # concurrent client requests.
  #msg-cache-size: 256M
  #rrset-cache-size: 512M
  #outgoing-range: 8192
  #num-queries-per-thread: 4096
local-data: "doubleclick.net A 127.0.0.1"
local-zone: "googlesyndication.com" redirect
local-data: "googlesyndication.com A 127.0.0.1"
local-zone: "googleadservices.com" redirect
local-data: "googleadservices.com A 127.0.0.1"
local-zone: "google-analytics.com" redirect
local-data: "google-analytics.com A 127.0.0.1"
local-zone: "ads.youtube.com" redirect
local-data: "ads.youtube.com A 127.0.0.1"
local-zone: "adserver.yahoo.com" redirect
local-data: "adserver.yahoo.com A 127.0.0.1"
local-zone: "ask.com" redirect
local-data: "ask.com A 127.0.0.1"
```

3.5. Attack Run

Start with the `-a` option. "Turn on warning of a duplicate A records."

```
root@kali-linux-vagrant:~# dnswalk -a fortress.lan.
Checking fortress.lan.
BAD: SOA record not found for fortress.lan.
BAD: fortress.lan. has NO authoritative nameservers!
BAD: All zone transfer attempts of fortress.lan. failed!
0 failures, 0 warnings, 3 errors.
```

Next, test with `-F`. "Perform fascist checking. When checking an A record, compare the PTR name for each IP address with the forward name and report mismatches."

```
root@kali-linux-vagrant:~# dnswalk -F fortress.lan.
Checking fortress.lan.
BAD: SOA record not found for fortress.lan.
BAD: fortress.lan. has NO authoritative nameservers!
BAD: All zone transfer attempts of fortress.lan. failed!
0 failures, 0 warnings, 3 errors.
```

Next, test with `-l`. "Perform `lame delegation` checking. For every NS record, check to see that the listed host is indeed returning authoritative answers for this domain."

```
root@kali-linux-vagrant:~# dnswalk -l fortress.lan.
Checking fortress.lan.
BAD: SOA record not found for fortress.lan.
BAD: fortress.lan. has NO authoritative nameservers!
BAD: All zone transfer attempts of fortress.lan. failed!
0 failures, 0 warnings, 3 errors.
```

Finally, test with `-r` to recursively desend into sub-domains for the specified domain.

```
root@kali-linux-vagrant:~# dnswalk -r fortress.lan.
Checking fortress.lan.
BAD: SOA record not found for fortress.lan.
BAD: fortress.lan. has NO authoritative nameservers!
BAD: All zone transfer attempts of fortress.lan. failed!
0 failures, 0 warnings, 3 errors.
```

We are not getting very far with my Unbound DNS server. I would love to see this at my day job on a current Windows DNS server with many sub-domains. That won't happen, ever!

What a tcpdump looks like on the DNS server.

```
tcpdump -i eth0 -s 1524 -nnvltqXX 'udp port 53 and host 172.20.156.78'
```

output:

```
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 1524 bytes
IP (tos 0x0, ttl 64, id 37671, offset 0, flags [DF], proto UDP (17), length 58)
    172.20.156.78.59075 > 172.20.156.5.53: UDP, length 30
    0x0000:  b827 eb88 4fbf 2477 03c4 7d2c 0800 4500  .'..O.$w..},..E.
    0x0010:  003a 9327 4000 4011 170f ac14 9c4e ac14  .:.'@.@......N..
    0x0020:  9c05 e6c3 0035 0026 04fc e304 0100 0001  .....5.&........
    0x0030:  0000 0000 0000 0866 6f72 7472 6573 7303  .......fortress.
    0x0040:  6c61 6e00 0006 0001                      lan.....
IP (tos 0x0, ttl 64, id 6656, offset 0, flags [none], proto UDP (17), length 58)
    172.20.156.5.53 > 172.20.156.78.59075: UDP, length 30
    0x0000:  2477 03c4 7d2c b827 eb88 4fbf 0800 4500  $w..},.'..O...E.
    0x0010:  003a 1a00 0000 4011 d036 ac14 9c05 ac14  .:....@..6......
    0x0020:  9c4e 0035 e6c3 0026 90b4 e304 8580 0001  .N.5...&........
    0x0030:  0000 0000 0000 0866 6f72 7472 6573 7303  .......fortress.
    0x0040:  6c61 6e00 0006 0001                      lan.....
```

4. Conclusion

From this paper, we have learned how to run dnswalk with 4 different parameters. The attack's target is running Unbound, "Unbound is a validating, recursive, caching DNS resolver", which based off of the output, looks to be misconfigured locally [it is not, since the service/application is a validating, recursive, caching DNS resolver [perhaps a better test would be to query NSD]]. The whole point of `dnswalk` is to discover misconfigurations on DNS servers, and then as an IT professsional, resolve those misconfigurations so that there is less chance of being compromised via a misconfigured service. Take the time to read the README here: https://github.com/davebarr/dnswalk/blob/master/README to fully understand the comments about WARNINGs and ERRORs.

I personally enjoyed writing and creating this paper covering dnswalk. We need more tools like this that are open source. Huge shout out to David Barr for writing this awesome tool. My appologies for taking so long to discover this tool.

4.1. Future research:

Other areas to consider outside the scope of this paper:

- the `dig` command. e.g. 'dig @172.20.156.5 +trace fortress.lan'
- the `dnsenum` command. e.g. 'dnsenum -r fortress.lan'
- the `whois` command. e.g. 'whois 172.20.156.5'
- the `theHarvester` command. e.g. 'theHarvester -d fortress.lan -l 10 -b google'
- the `dnsmap` command. e.g. 'dnsmap fortress.lan'

5. Appendix

References

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

https://niccs.cisa.gov/training/search/cybrary/how-use-dnswalk-bswr

https://nlnetlabs.nl/projects/unbound/about/

5.1. Home Page:

https://github.com/davebarr/dnswalk

5.2. Source Code:

https://salsa.debian.org/debian/dnswalk