

How to securely isolate and execute Doona from Kali Linux

Introduction

The motivation behind this paper is to explore using Doona that comes with Kali Linux. Doona is forked from the Bruteforce Exploit Detector(BED) tool. In case you didn't know, BED is a program which is designed to check daemons for potential buffer overflows, format string bugs etc. Doona adds a significant number of features/changes from the BED fork. The secure isolation comes from using the private network from VirtualBox, here the 172.20.156.0/24 subnet. Private means it is Host-only and cannot route outbound.

Requirements

If you see the following \$ symbol on a command line to execute, what that means is that the command is executed as a regular user, i.e. the Ubuntu user. Ignore the leading \$ and execute the rest of the command.

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

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

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

VirtualBox

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

The author is running on Ubuntu 18.04, so following to this URL:

https://www.virtualbox.org/wiki/Linux_Downloads

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

Clean VirtualBox Networking

Run these two commands from a Terminal:

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

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

```
loopback mappings (ipv4)
  127.0.0.1=2
```

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

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

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

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

```
VBoxManage natnetwork remove --netname <NetworkName_from_above>
VBoxManage natnetwork remove --netname 192.168.139-NAT
# repeat as many times as necessary to delete all of them.
```

```
VBoxManage dhcpserver remove --netname <DHCP_Server_NetworkName_from_above>
VBoxManage dhcpserver remove --netname 192.168.139-NAT
# repeat as many times as necessary to delete all of them.
```

Add VirtualBox Networking

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

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

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

```
VBoxManage hostonlyif create
```

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

```
VBoxManage dhcpserver add \
  --ifname vboxnet0 \
  --ip 172.20.0.3 \
  --lowerip 172.20.0.101 \
```

```
--upperip 172.20.0.254 \  
--netmask 255.255.255.0
```

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

Vagrant

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

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

Kali Linux and Damn Vulnerable Web Application (DVWA)

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

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

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

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

From a Terminal, change directory to:

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

Vagrantfile:

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

```
# -*- mode: ruby -*-  
# vi: set ft=ruby :  
  
$os_update = <<SCRIPT  
apt-get update  
SCRIPT  
  
$install_doona = <<SCRIPT  
apt-get install -y doona  
SCRIPT  
  
# Vagrantfile API/syntax version.  
VAGRANTFILE_API_VERSION = "2"
```

```

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

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

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

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

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

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

    conf.vm.box = "ubuntu/xenial64"

    conf.vm.hostname = "dvwa-vagrant"

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

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

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

    conf.vm.provider "virtualbox" do |vb|
      vb.memory = "1024"
      vb.cpus = "2"
      vb.gui = false
      vb.customize ["modifyvm", :id, "--vram", "32"]
      vb.customize ["modifyvm", :id, "--accelerate3d", "off"]
      vb.customize ["modifyvm", :id, "--ostype", "Ubuntu_64"]
      vb.customize ["modifyvm", :id, "--boot1", "dvd"]
      vb.customize ["modifyvm", :id, "--boot2", "disk"]
      vb.customize ["modifyvm", :id, "--audio", "none"]
      vb.customize ["modifyvm", :id, "--clipboard", "hosttoguest"]
      vb.customize ["modifyvm", :id, "--draganddrop", "hosttoguest"]
      vb.customize ["modifyvm", :id, "--paravirtprovider", "kvm"]
    end
    conf.vm.provision :shell, path: "bootstrap.sh"
  end
end

```

Save and write this file.

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

bootstrap.sh (include the shebang in your file, the first line with '#!/usr/bin/env bash'):

```
#!/usr/bin/env bash
PHP_FPM_PATH_INI='/etc/php/7.0/fpm/php.ini'
PHP_FPM_POOL_CONF='/etc/php/7.0/fpm/pool.d/www.conf'
MYSQL_ROOT_PW='Assword12345'
MYSQL_dwva_user='dwva_root'
MYSQL_dwva_password='sunshine'
DVWA_admin_password='admin'
recaptcha_public_key='u8392ihj32k18hujalkshuil32'
recaptcha_private_key='89ry8932873832lih32ilj32'

install_base() {
    add-apt-repository -y ppa:nginx/stable
    sudo apt-get update
    sudo apt-get dist-upgrade -y
    sudo apt-get install -y nginx mariadb-server mariadb-client php php-common php-cgi php-fpm
    php-gd php-cli php-pear php-mcrypt php-mysql php-gd git vim
}

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

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

    service mysql restart
}

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

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

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

    systemctl restart php7.0-fpm
}

config_nginx(){
cat << 'EOF' > /etc/nginx/sites-enabled/default
server
{
    listen 80;
```

```

root /var/www/html;
index index.php index.html index.htm;
#server_name localhost
location "/"
{
    index index.php index.html index.htm;
    #try_files $uri $uri/ =404;
}

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

systemctl restart nginx
}

install_dvwa(){

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

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

### chmod uploads and log file to be writable by nobody
chmod 777 ./hackable/uploads/
chmod 777 ./external/phpids/0.6/lib/IDS/tmp/phpids_log.txt

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

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

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

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

```

```
sed -i '/smithy/ s/password/'${DVWA_admin_password}.'/g'  
/var/www/html/dvwa/includes/DBMS/MySQL.php  
}
```

```
install_base  
config_mysql  
install_dvwa  
update_mysql_user_pws  
config_php  
config_nginx
```

Save and write this file.

If you have issues with copying and pasting the above file, you could use `wget`, i.e. Make sure the `bootstrap.sh` file ends up in the same directory as the Vagrantfile. I just googled for the string, “DVWA AND bootstrap.sh”, case sensitive here with the AND.

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

From a Terminal, change directory to:

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

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

```
$ vagrant up
```

This will download the appropriate images and start the virtual machines.

Once running, through the VirtuaBox GUI, login as root. Password is “toor”, root backwards. Edit the following file:

```
/etc/ssh/sshd_config
```

And change the line:

```
#PermitRootLogin prothibit-password
```

To:

```
PermitRootLogin yes
```

Then restart the ssh daemon:

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

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

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

```
# vagrant status
```

Choose the second network adapter, it should look like:

Current machine states:

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

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

From there, log into the DVWA instance with:

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

And then get the current IP address.

```
$ ip a
```

Choose the second network adapter, it should look like:

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

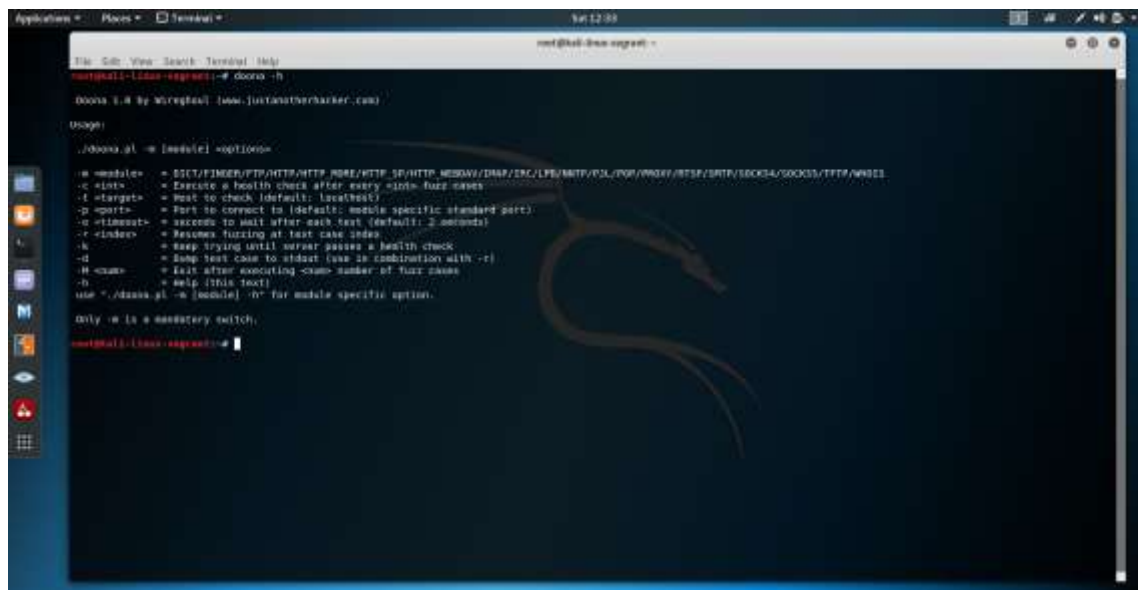

The author's home wireless network uses 172.20.156.0/24 as the network range. Therefore, the adapter, enp0s8 is what he is looking for. The IP to use as a target is 172.20.156.76. Write down your value.

doona (Kali-Linux version)

Fire up both vagrant boxes of Kali-Linux and DVWA with vagrant up.

Login to kali linux. Username: root, Password: toor.

Open a terminal and I would recommend to at least look at the help page of Doona with the command:
doona -h



```
root@kali:~# doona -h
doona 1.0 by w3rgr34n | www.justanotherhacker.com

Usage:
./doona.pl -m [module] -o [options]

-m module - DICHT/FINGER/FTP/HTTP/HTTPS/MS01/SMTP/IP/HTTP_MISDIR/DNS/IRC/LPB/SMTP/SQL/POP/PPH/RTSP/SNTP/SOCKS4/SOCKS5/TFTP/WSH2
-c sleep - Execute a health check after every -jobs fuzz cases
-f target - Host to check (default: localhost)
-p ports - Port to connect to (default: module specific standard port)
-e timeout - seconds to wait after each fuzz (default: 2 seconds)
-r retries - Maximes fuzzing at host case sides
-k keep - keep trying until server passes a health check
-d dump - dump test case to stdout (use in combination with -r)
-n count - Exit after executing -count number of fuzz cases
-h help - this tool

use './doona.pl -m [module] -o' for module specific option.

only -m is a mandatory switch.
root@kali:~#
```

This might seem silly, but the help page gives us a list of all of the parameters we can pass to the executable, in this case, a Perl script, named doona.pl. I am pointing this out because we can see the code if we need to by running:
\$ less \$(which doona)

Let us launch the script at our target (DVWA) which we know is on 172.20.156.76

With the command:

\$ doona -m HTTP -t 172.20.156.76 -p 80

```
root@kali:~# nmap -sV -p 80 172.20.156.76 -o-
Nmap scan report for 172.20.156.76
Host is up (0.0000s latency).
Banner: HTTP/1.0
OS: Linux 3.10 (Ubuntu)
Service detected: Doona 1.0 by Wireghoul (www.justanotherhacker.com)
+ Buffer overflow testing
1/39 [XAXAX] ..... (45)
2/39 [XAXAX / HTTP/1.0] ..... (90)
3/39 [HEAD XAXAX HTTP/1.0] ..... (135)
4/39 [HEAD /XAXAX HTTP/1.0] ..... (180)
5/39 [HEAD /XAXAX HTTP/1.0] ..... (225)
6/39 [HEAD / XAXAX] ..... (270)
7/39 [HEADXAXAX / HTTP/1.0] ..... (315)
8/39 [GET XAXAX HTTP/1.0] ..... (360)
9/39 [GET /XAXAX HTTP/1.0] ..... (405)
10/39 [GET /XAXAX.html HTTP/1.0] ..... (450)
11/39 [GET /index.XAXAX HTTP/1.0] ..... (495)
12/39 [GET /-XAXAX HTTP/1.0] ..... (540)
13/39 [GET /?XAXAX HTTP/1.0] ..... (585)
14/39 [GET /?XAXAX=x HTTP/1.0] ..... (630)
15/39 [GET /?x=XAXAX HTTP/1.0] ..... (675)
16/39 [GET / XAXAX] ..... (720)
17/39 [GET / HTTP/XAXAX] ..... (765)
18/39 [GET /XAXAX] ..... (810)
19/39 [GETXAXAX / HTTP/1.0] ..... (855)
20/39 [POST XAXAX HTTP/1.0] ..... (900)
21/39 [POST /XAXAX HTTP/1.0] ..... (945)
22/39 [POST /XAXAX HTTP/1.0] ..... (990)
23/39 [POST / XAXAX] ..... (1035)
24/39 [POST /XAXAX] ..... (1080)
25/39 [POST / HTTP/1.0XAXAX] ..... (1125)
26/39 [POST / HTTP/1.0Content-Length:] ..... (1170)
27/39 [POST / HTTP/1.0Content-Type: n] ..... (1215)
28/39 [POST / HTTP/1.0Content-Type: n] ..... (1260)
29/39 [POST / HTTP/1.0Content-Type: n] ..... (1305)
30/39 [OPTIONS XAXAX HTTP/1.0] ..... (1350)
31/39 [OPTIONS /XAXAX HTTP/1.0] ..... (1395)
32/39 [OPTIONS / XAXAX] ..... (1440)
33/39 [PUT XAXAX HTTP/1.0] .....
```

The left most column tells us the sequence number this is cycling through, the middle column with <[attack]> tells us exactly what attack is running, and the right most column (digits) tells us how many vulnerabilities the tools has discovered.

```
42/532 GET / HTTP/1.0 [Pragma: XAXAX] ..... (125762)
43/532 GET / HTTP/1.0 [Expect: XAXAX] ..... (125791)
44/532 GET / HTTP/1.0 [Range: XAXAX] ..... (125820)
45/532 GET / HTTP/1.0 [Range: bytes=1-XAXAX] ..... (125849)
46/532 GET / HTTP/1.0 [Range: bytes=0-1,XAXAX] ..... (125878)
47/532 GET / HTTP/1.0 [Content-Length: XAXAX] ..... (125907)
48/532 GET / HTTP/1.0 [Content-Type: XAXAX] ..... (125936)
49/532 GET / HTTP/1.0 [Content-Type: text/html; XAXAX] ..... (125965)
50/532 GET / HTTP/1.0 [Content-Type: XAXAX/html; char] ..... (125994)
51/532 GET / HTTP/1.0 [Content-Type: text/XAXAX; char] ..... (126023)
52/532 GET / HTTP/1.0 [Content-Type: text/html; XAXAX] ..... (126052)
53/532 GET / HTTP/1.0 [Content-Type: text/html; chars] ..... (126081)
54/532 GET / HTTP/1.0 [Content-Encoding: XAXAX] ..... (126110)
55/532 GET / HTTP/1.0 [Content-Encoding: XAXAXCache-c] ..... (126139)
56/532 GET / HTTP/1.0 [Content-Language: XAXAX] ..... (126168)
57/532 GET / HTTP/1.0 [Cache-control: XAXAX] ..... (126197)
58/532 GET / HTTP/1.0 [Cache-control: max-age=XAXAX] ..... (126226)
59/532 GET / HTTP/1.0 [Cache-control: min-fresh=XAXAX] ..... (126255)
60/532 GET / HTTP/1.0 [Cache-control: max-stale=XAXAX] ..... (126284)
61/532 GET / HTTP/1.0 [Cookie: XAXAX] ..... (126313)
62/532 GET / HTTP/1.0 [Cookie: XAXAX=abc] ..... (126342)
63/532 GET / HTTP/1.0 [Cookie: abc=XAXAX] ..... (126371)
64/532 GET / HTTP/1.0 [Content-Location: XAXAX] ..... (126400)
65/532 GET / HTTP/1.0 [Content-Language: XAXAX] ..... (126429)
66/532 GET / HTTP/1.0 [Content-MD5: XAXAX] ..... (126458)
67/532 GET / HTTP/1.0 [Content-Range: 0-XAXAX/1024] ..... (126487)
68/532 GET / HTTP/1.0 [Content-Range: XAXAX-500/1024] ..... (126516)
69/532 GET / HTTP/1.0 [Content-Range: 0-500/XAXAX] ..... (126545)
70/532 GET / HTTP/1.0 [X-Header: XAXAX XAXAX] ..... (126574)
71/532 GET / HTTP/1.0 [TE: XAXAX] ..... (126603)
72/532 GET / HTTP/1.0 [Trailer: XAXAX] ..... (126632)
73/532 GET / HTTP/1.0 [Transfer-Encoding: XAXAX] ..... (126661)
74/532 GET / HTTP/1.0 [Via: XAXAX] ..... (126690)
75/532 GET / HTTP/1.0 [X-Forwarded-For: XAXAX] ..... (126719)
76/532 GET / HTTP/1.0 [Upgrade: XAXAX/1.0Connection: ] ..... (126748)
77/532 POST / HTTP/1.0 [XAXAX: XAXAX] ..... (126777)
78/532 POST / HTTP/1.0 [User-Agent: XAXAX] ..... (126806)
79/532 POST / HTTP/1.0 [Host: XAXAX] ..... (126835)
80/532 POST / HTTP/1.0 [Host: XAXAX:80] ..... (126864)
81/532 POST / HTTP/1.0 [Host: somehost:XAXAX] ..... (126893)
82/532 POST / HTTP/1.0 [Accept: XAXAX] .....
```

The above image is what the tool looks like after running for four days. The test system has 8 cores, running at 2.6GHz. I'm pointing this out because we know that the install of DVWA has many vulnerabilities, but looking above, this tool has discovered 126,893 vulnerabilities. I would love to know what this tool finds when running against something like SharePoint or another modern web server that is mis-configured.

Conclusion

By following this paper, you have setup a secure lab that is flexible enough to test many different configurations/tests. Doona is a great tool. The above shows how to run it and how powerful the results are from the output of the tool. For initial runs, I would recommend to tack on the parameter of '-M 500' to limit the count of how many tests are performed. Running the tool on the Host-only network protects the tests and your work. Make sure – as a final warning – that you do not alter this pattern. If you perform security tests on anything other than assets that you own AND contained within your own network, you open yourself to legal troubles.