How to securely isolate and execute Amap from Kali Linux

Version 0.1, Last Updated: 2024-08-10

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

Table of Contents

Chapter 1. Introduction

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

What does `amap` do:

```
"Without filled databases containing triggers and responses, the tool is
worthless, so I ask you to help us fill the database. How to do this? Well,
whenever a client application connects to a server, some kind of handshake
is exchanged (at least, usually. Syslogd for instance won't say nothing,
and snmpd without the right community string neither). Anyway, amap takes
the first packet sent back and compares it to a list of signature responses.
Really simple, actually. And in reality, it turns out really to be that
simple, at least, for most protocols." -- README.md from the source code, see
Appendix.
```

Key functions included with amap:

1. **Application Fingerprinting:** AMAP can identify services running on open ports by analyzing their responses to specific queries or data packets. This is useful when services are running on non-standard ports, making it difficult to identify them through standard port-scanning tools like Nmap.
2. **Protocol Identification:** It identifies common protocols such as HTTP, FTP, SSH, DNS, and others. This helps in detecting which protocol is used on a specific port, even if it's unexpected or obscured.
3. **Detection of Banner-Obscured Services:** AMAP can handle situations where banner grabbing (the typical method of identifying a service) is unreliable because the services are hiding or obfuscating their banner information. This makes it effective against services that try to avoid detection.

4. **Service Versioning:** In addition to identifying the service type, AMAP attempts to discover the version of the application running on the identified port. This is crucial for vulnerability assessments, where specific versions may be associated with known security vulnerabilities.
5. **Flexible Targeting:** AMAP can scan specific hosts or entire network ranges. You can specify individual IP addresses, subnets, or use hostnames as input targets.
6. **Stealthy Service Detection:** By using randomized port-scanning techniques and less aggressive fingerprinting methods, AMAP can help evade some types of intrusion detection systems (IDS).
7. **Support for a Wide Range of Services:** It supports a vast library of services, which helps in detecting unusual or less common services in a network, beyond the typical ones like web or mail servers.
8. **Command-line Tool:** As a command-line tool, AMAP offers flexibility for scripting and automation, allowing it to be integrated into larger security testing workflows.
9. **Integration with Other Tools:** AMAP can be used in conjunction with other network scanning tools (such as Nmap) to enhance the effectiveness of network reconnaissance efforts.
10. **Port Scanning:** AMAP can be combined with simple port scanning to not only identify open ports but also attempt to determine the type of service running on them, going beyond what a regular port scanner might accomplish.

Warning: The source code page has a README.md that states: "NOTE: THIS TOOL IS OUTDATED. USE NMAP."

Therefore, this paper serves as a historical examination of an older tool,

intended for informational purposes only. For professionals conducting network scanning, it is recommended to utilize more current tools such as `nmap` or other modern alternatives.

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.

Chapter 3. Amap

3.1. Install

Before you install, see if the tool is already installed with the command:

```
type -a amap
# could also run: 'which amap'
```

If there is no output, you can then run:

```
apt install amap
```

Output:

```
Installing:
  amap

Summary:
  Upgrading: 0, Installing: 1, Removing: 0, Not Upgrading: 1726
  Download size: 68.5 kB
  Space needed: 181 kB / 23.6 GB available

Get:1 http://kali.download/kali kali-rolling/main amd64 amap amd64 5.4-4kali3
[68.5 kB]
Fetched 68.5 kB in 0s (422 kB/s)
Selecting previously unselected package amap.
(Reading database ... 375393 files and directories currently installed.)
Preparing to unpack .../amap_5.4-4kali3_amd64.deb ...
Unpacking amap (5.4-4kali3) ...
Setting up amap (5.4-4kali3) ...
Processing triggers for kali-menu (2022.3.1) ...
Processing triggers for man-db (2.10.2-1) ...
```

3.2. Help page

As always, to see the help page on UNIX and Linux binaries, you typically add the parameter: `--help`.

```
┌──(root㉿kali-linux-vagrant)-[~]
└─# amap --help
amap v5.4 (c) 2011 by van Hauser <vh@thc.org> www.thc.org/thc-amap
Syntax: amap [-A|-B|-P|-W] [-1buSRHUdqv] [[-m] -o <file>] [-D <file>] [-t/-T
sec] [-c cons] [-C retries] [-p proto] [-i <file>] [target port [port] ...]
Modes:
  -A         Map applications: send triggers and analyse responses (default)
  -B         Just grab banners, do not send triggers
  -P         No banner or application stuff - be a (full connect) port scanner
Options:
  -1         Only send triggers to a port until 1st identification. Speeeeed!
  -6         Use IPv6 instead of IPv4
  -b         Print ascii banner of responses
  -i FILE    Nmap machine readable outputfile to read ports from
  -u         Ports specified on commandline are UDP (default is TCP)
  -R         Do NOT identify RPC service
  -H         Do NOT send application triggers marked as potentially harmful
  -U         Do NOT dump unrecognised responses (better for scripting)
  -d         Dump all responses
  -v         Verbose mode, use twice (or more!) for debug (not recommended :-)
  -q         Do not report closed ports, and do not print them as unidentified
  -o FILE [-m] Write output to file FILE, -m creates machine readable output
  -c CONS    Amount of parallel connections to make (default 32, max 256)
  -C RETRIES Number of reconnects on connect timeouts (see -T) (default 3)
  -T SEC     Connect timeout on connection attempts in seconds (default 5)
  -t SEC     Response wait timeout in seconds (default 5)
  -p PROTO   Only send triggers for this protocol (e.g. ftp)
  TARGET PORT   The target address and port(s) to scan (additional to -i)
amap is a tool to identify application protocols on target ports.
Note: this version was NOT compiled with SSL support!
Usage hint: Options "-bqv" are recommended, add "-1" for fast/rush checks.
```

3.3. Running the tool

From the developer's README.md:

```
Amap takes nmap -oM output files as input. You can specify a single IP
address and port)s= on the command line, but usually, you'd run it from a nmap
file, thusly:

   # (first "nmap -sS -oM results.nmap -p 1-65535 TARGET" of course)
   # amap -i results.nmap -o results.amap -m
```

```
or:

   # amap 127.0.0.1 443

or:

   # amap target 21-6000
```

From the above, yes we could run the sequence:

```
# firstly
nmap -sS -oM results.nmap -p 1-65535 <remote_target_IP_Address>

# secondly
amap -i results.nmap -o results.amap -m
```

Instead of doing that, I am thinking of just running the tooling with something like:

```
amap <remote_target_IP_Address> 21-65535

# or for our secure lab, the IP address of DVWA changed this week to:
192.168.99.152

amap 192.168.99.152 21-65535 -q
```

The `-q` will suppress an ugly listing at the end of the output.

Output:

```
┌──(root㉿kali-linux-vagrant)-[~]
└─# amap 192.168.99.152 21-65535 -q
amap v5.4 (www.thc.org/thc-amap) started at 2024-08-10 08:54:21 - APPLICATION
MAPPING mode

Protocol on 192.168.99.152:80/tcp matches http
Protocol on 192.168.99.152:80/tcp matches http-apache-2
Protocol on 192.168.99.152:22/tcp matches ssh
Protocol on 192.168.99.152:22/tcp matches ssh-openssh
Protocol on 192.168.99.152:111/tcp matches rpc
```

```
Protocol on 192.168.99.152:111/tcp matches rpc-rpcbind-v4

amap v5.4 finished at 2024-08-10 08:54:35
```

Now I am curious. Let's see if we can scan my homelab.

```
┌──(root㉿kali-linux-vagrant)-[~]
└─# time nmap -sS -oM results.nmap -p 1-65535 172.20.156.120 172.20.156.121
172.20.156.122
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-10 09:01 EDT
```

Incidentally: If you are not tracking the above command, we are pivoting away from the author's secure DVWA setup and scanning a different subnet that is running Raspberry Pi's version 4. This is intentional and meant to demonstrate a different set of results.

The output from the file is:

```
┌──(root㉿kali-linux-vagrant)-[~]
└─# cat results.nmap
# Nmap 7.94SVN scan initiated Sat Aug 10 10:13:31 2024 as: /usr/lib/nmap/nmap
-sS -oM results.nmap -p 1-65535 172.20.156.120 172.20.156.121 172.20.156.122
Host: 172.20.156.120 () Status: Up
Host: 172.20.156.120 () Ports: 22/open/tcp//ssh///, 179/open/tcp//bgp///,
2379/open/tcp//etcd-client///, 2380/open/tcp//etcd-server///,
6443/open/tcp//sun-sr-https///, 9100/open/tcp//jetdirect///,
10250/open/tcp/////, 10256/open/tcp/////, 30000/open/tcp//ndmps///,
30253/open/tcp/////, 31679/open/tcp/////, 31982/open/tcp/////,
32000/open/tcp///// Ignored State: closed (65522)
Host: 172.20.156.121 () Status: Up
Host: 172.20.156.121 () Ports: 22/open/tcp//ssh///, 179/open/tcp//bgp///,
2379/open/tcp//etcd-client///, 2380/open/tcp//etcd-server///,
6443/open/tcp//sun-sr-https///, 9100/open/tcp//jetdirect///,
10250/open/tcp/////, 10256/open/tcp/////, 30000/open/tcp//ndmps///,
30253/open/tcp/////, 31679/open/tcp/////, 31982/open/tcp/////,
32000/open/tcp///// Ignored State: closed (65522)
Host: 172.20.156.122 () Status: Up
Host: 172.20.156.122 () Ports: 22/open/tcp//ssh///, 179/open/tcp//bgp///,
2379/open/tcp//etcd-client///, 2380/open/tcp//etcd-server///,
6443/open/tcp//sun-sr-https///, 9100/open/tcp//jetdirect///,
10250/open/tcp/////, 10256/open/tcp/////, 30000/open/tcp//ndmps///,
30253/open/tcp/////, 31679/open/tcp/////, 31982/open/tcp/////,
32000/open/tcp///// Ignored State: closed (65522)
```

```
# Nmap done at Sat Aug 10 10:14:12 2024 -- 3 IP addresses (3 hosts up) scanned
in 41.26 seconds
```

This looks like a kubernetes control nodes because of the ETCD server.

Now we can run:

```
time amap -i results.nmap -o results.amap -m -q
```

Sidebar: The author of this paper loves using the `time` command to collect metrics on how long jobs take to run. It helps in sampling small runs to then estimate the job completion time of larger jobs.

Output:

```
┌──(root㉿kali-linux-vagrant)-[~]
└─# time amap -i results.nmap -o results.amap -m -q
Warning: output file already exists. Moving to results.amap.old
amap v5.4 (www.thc.org/thc-amap) started at 2024-08-10 10:17:42 - APPLICATION
MAPPING mode

Protocol on 172.20.156.120:9100/tcp matches http
Protocol on 172.20.156.120:10250/tcp matches http
Protocol on 172.20.156.120:10256/tcp matches http
Protocol on 172.20.156.120:30000/tcp matches http
Protocol on 172.20.156.120:30253/tcp matches http
Protocol on 172.20.156.120:31679/tcp matches http
Protocol on 172.20.156.120:31982/tcp matches http
Protocol on 172.20.156.120:32000/tcp matches http
Protocol on 172.20.156.121:6443/tcp matches http
Protocol on 172.20.156.121:9100/tcp matches http
Protocol on 172.20.156.121:10250/tcp matches http
Protocol on 172.20.156.121:10256/tcp matches http
Protocol on 172.20.156.121:30000/tcp matches http
Protocol on 172.20.156.121:31679/tcp matches http
Protocol on 172.20.156.122:22/tcp matches ssh
Protocol on 172.20.156.122:22/tcp matches ssh-openssh
Protocol on 172.20.156.120:22/tcp matches ssh
Protocol on 172.20.156.120:22/tcp matches ssh-openssh
Protocol on 172.20.156.120:6443/tcp matches http
Protocol on 172.20.156.121:22/tcp matches ssh
Protocol on 172.20.156.121:22/tcp matches ssh-openssh
```

```
Protocol on 172.20.156.121:30253/tcp matches http
Protocol on 172.20.156.121:31982/tcp matches http
Protocol on 172.20.156.121:32000/tcp matches http
Protocol on 172.20.156.122:6443/tcp matches http
Protocol on 172.20.156.122:9100/tcp matches http
Protocol on 172.20.156.122:10250/tcp matches http
Protocol on 172.20.156.120:6443/tcp matches ssl
Protocol on 172.20.156.122:10256/tcp matches http
Protocol on 172.20.156.122:30000/tcp matches http
Protocol on 172.20.156.122:30253/tcp matches http
Protocol on 172.20.156.122:31679/tcp matches http
Protocol on 172.20.156.122:31982/tcp matches http
Protocol on 172.20.156.122:32000/tcp matches http
Protocol on 172.20.156.120:2379/tcp matches ssl
Protocol on 172.20.156.120:2380/tcp matches ssl
Protocol on 172.20.156.120:10250/tcp matches ssl
Protocol on 172.20.156.121:6443/tcp matches ssl
Protocol on 172.20.156.120:31679/tcp matches ssl
Protocol on 172.20.156.120:31982/tcp matches ssl
Protocol on 172.20.156.121:2379/tcp matches ssl
Protocol on 172.20.156.121:2380/tcp matches ssl
Protocol on 172.20.156.122:2379/tcp matches ssl
Protocol on 172.20.156.122:2380/tcp matches ssl
Protocol on 172.20.156.121:10250/tcp matches ssl
Protocol on 172.20.156.121:31679/tcp matches ssl
Protocol on 172.20.156.121:31982/tcp matches ssl
Protocol on 172.20.156.122:6443/tcp matches ssl
Protocol on 172.20.156.122:10250/tcp matches ssl
Protocol on 172.20.156.122:31679/tcp matches ssl
Protocol on 172.20.156.122:31982/tcp matches ssl

amap v5.4 finished at 2024-08-10 10:19:42

real    119.79s
user    52.83s
sys 60.81s
cpu 94%
```

Chapter 4. Conclusion

Based on the analysis conducted above, the Kali Linux tool `amap` proves to be effective in identifying the services running on a remote target's ports, even in cases where there is a discrepancy between the port number and the service's default protocol port. For instance, `amap` can detect an HTTPS server running on TCP port 9753, despite this being non-standard.

I concur with the original developer's evaluation of the tool and believe that utilizing `nmap` for scanning tasks may provide a more streamlined approach. While `amap` is not an ineffective tool, it has become less prominent over time and should be considered to be of diminished relevance.

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

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

https://www.thc.org/

Source Code: https://gitlab.com/kalilinux/packages/amap