

1.3 Autorecon

Overview

Having an effective (both in data enumerated and efficiency in the time spent) approach to enumeration is a high priority for any penetration tester. While there are many approaches to enumeration, any task that you need to do more than once should be automated. Autorecon allows you to automate the enumeration phase of your penetration test, without compromising on detailed results.

Autorecon

How is it used

At its core, Autorecon is a Python application. It is installed by cloning the GitHub repository, and installing the dependencies. Once it is installed, you simply point the script at a target (the machine you are enumerating) and let it work through the scans. Once it is done, it will provide a directory which will be packed full of the scan results. You will also find a list of the commands Autorecon ran, along with a list of additional `manual commands` you may be interested in.

Autorecon also has the ability to scan targets from a `target file`. This is a text file with the `IP` address of each target, one per line.

Installing and using Autorecon will be covered in an exercise shortly.

Why is this important

Scanning targets is another penetration testing staple. Depending on the services initially uncovered with a recon tool such as `nmap`, the results will flow into another collection of tools for related scans. These in turn, can lead to a third tier of tools. If you are keeping up, this means you need to keep track of many dependent tools and commands.

Imagine you have started scanning a target, you find a service, and forgot to execute a command (to err is human). As a result, you have not seen a service, so you don't keep digging and you end up missing an opportunity for accessing the machine.

From the scenario above, it is clear that having consistent and `repeatable` results is extremely important in remaining methodical when enumerating targets.

Real-world applications

Consider the following scenario

You have been asked to conduct black-box testing on five targets in a lab. On one of the machines, you have been told to focus on the payment system web interface. For the other

four, you add their IP addresses to the targets file and run `Autorecon`. By the time you are finished conducting your command injection testing on the web application, you have detailed results from Autorecon, conveniently stored in an organised folder structure, along with a history of all commands run.

It should be clear that there are significant efficiencies in the use of automated tools as above. Having organised notes has already been discussed as being one of (if not the most) important aspects of penetration testing.

Potential Issues

Autorecon is an aggressive tool, it is more than capable of crashing some services and triggering preventative messages (such as fail2ban).

You need to run Autorecon with `root` level privileges. If you don't it will appear to work (the worst kind of broken), but it will miss out on enumerating key services. If you follow the setup section, you will be setup with shortcuts that automatically call it with `sudo`.

You will encounter some situations where you are unable to use autorecon, such as that descried below.

Consider a scenario where you are attempting to scan a target over a pivot, or you have identified the target has services hosted on an IPv6 address. You will quickly encounter issues with autorecon in that it will attempt to use tools that do not function over certain pivot types, or that it does not currently work with IPv6 addresses at all.

In these instances, you would likely need to revert to manual enumeration. We will look at that in a future module.

Exercise

Installation

Start off by updating your list of packages:

```
sudo apt update
```

And install the package dependencies. Most will already be there, but just in case:

```
sudo apt install seclists curl enum4linux gobuster nbtscan nikto nmap  
onesixtyone oscanner smbclient smbmap smtp-user-enum snmp sslscan  
sipvicious tnscomd10g whatweb wkhtmltopdf python3-venv python3 python3-pip
```

Now to setup `pipx`

Don't use `sudo` or install as root for this part.

```
python3 -m pip install --user pipx
python3 -m pipx ensurepath
```

Close and re-open your terminal so it's all loaded.

You can check your version of `pipx` now, make sure it's modern (0.16.0.0+, you will have issues if it's 0.12.x.x which is the version installed through the kali package manager)

If you are on version `0.12.x.x`, run `sudo apt remove pipx` and remove the package manager version.

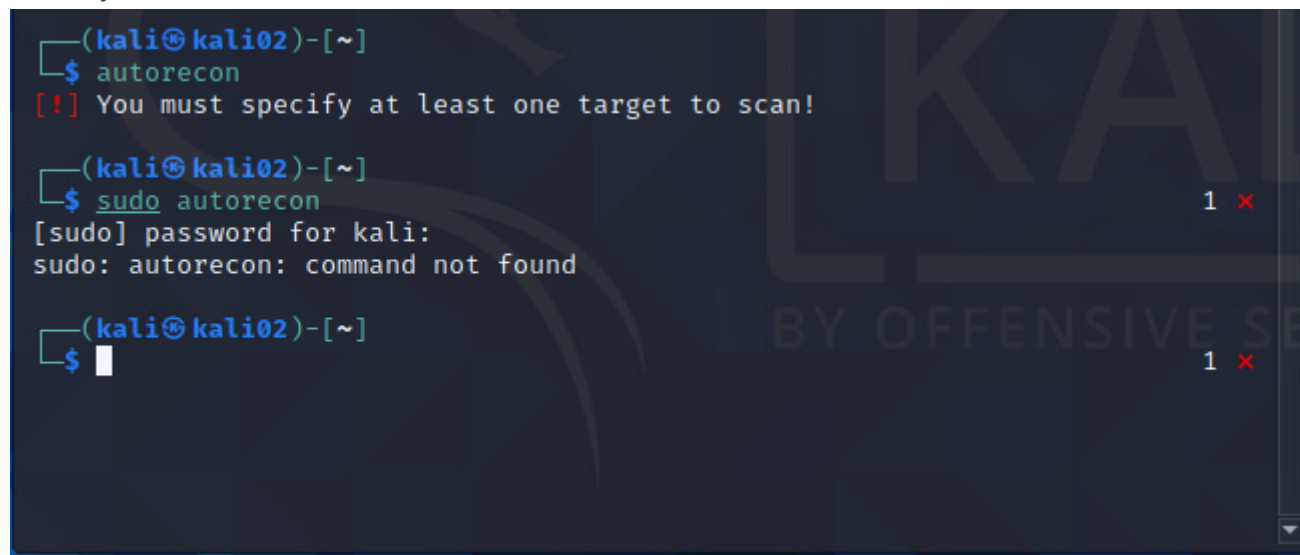
Now to install `AutoRecon`

```
pipx install git+https://github.com/Tib3rius/AutoRecon.git
```

Assuming all went well, you will see an output like that below.

```
(kali㉿kali02)-[~]
└─$ pipx install git+https://github.com/Tib3rius/AutoRecon.git
    installed package autorecon 1.0.0, Python 3.9.1
    These apps are now globally available
      - autorecon
done! ✨ ✨ ✨
```

Let's try and run it:



```
(kali㉿kali02)-[~]
└─$ autorecon
[!] You must specify at least one target to scan!

(kali㉿kali02)-[~]
└─$ sudo autorecon
[sudo] password for kali:
sudo: autorecon: command not found

(kali㉿kali02)-[~]
└─$
```

It works fine as the `kali` user, but not with `sudo`. If you recall, we need to run AutoRecon as `root` or with `sudo` so we need to get that fixed.

Fixing sudo

This part is easy and the `AutoRecon` GitHub offers a solution:

To make this easier, you could add the following alias to your `~/.profile` (or equivalent):

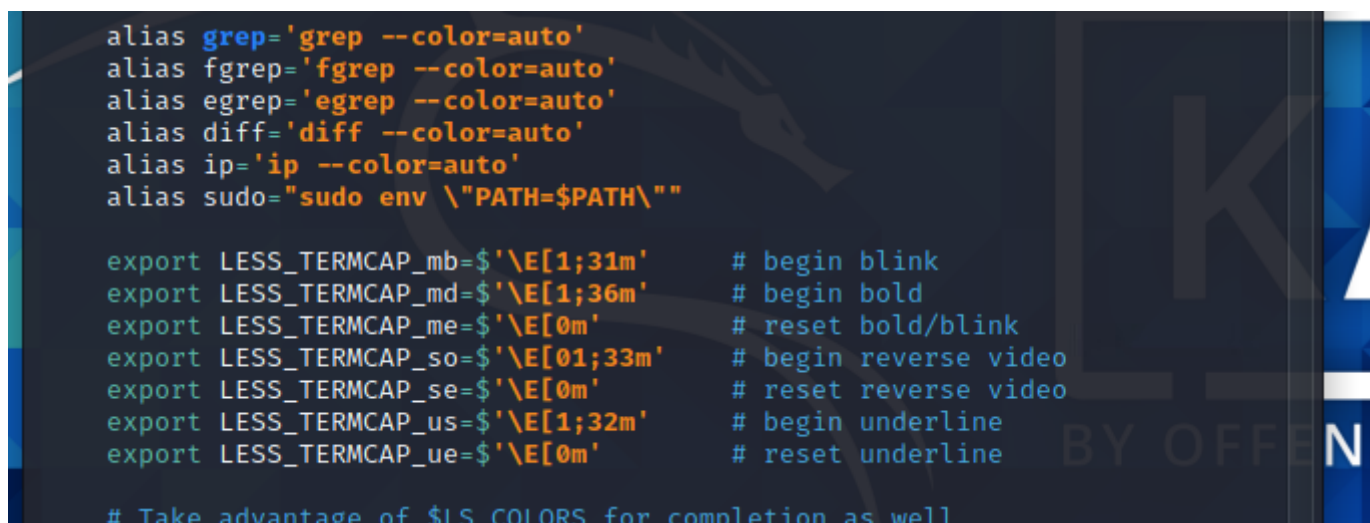
```
alias sudo="sudo env \"PATH=$PATH\""
```

As the default shell on Kali is `ZSH`, we need to modify `.zshrc` which is in our home folder.

```
nano ~/.zshrc
```

Scroll down to the `alias` section and add the following line

```
alias sudo="sudo env \"PATH=$PATH\""
```



```
alias grep='grep --color=auto'
alias fgrep='fgrep --color=auto'
alias egrep='egrep --color=auto'
alias diff='diff --color=auto'
alias ip='ip --color=auto'
alias sudo="sudo env \"PATH=$PATH\""
```

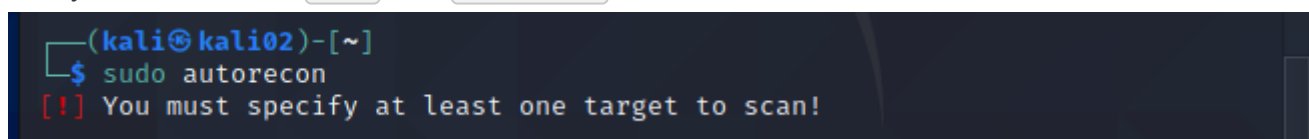
```
export LESS_TERMCAP_mb=$'\E[1;31m'      # begin blink
export LESS_TERMCAP_md=$'\E[1;36m'      # begin bold
export LESS_TERMCAP_me=$'\E[0m'         # reset bold/blink
export LESS_TERMCAP_so=$'\E[01;33m'     # begin reverse video
export LESS_TERMCAP_se=$'\E[0m'         # reset reverse video
export LESS_TERMCAP_us=$'\E[1;32m'     # begin underline
export LESS_TERMCAP_ue=$'\E[0m'         # reset underline
```

```
# Take advantage of $LS_COLORS for completion as well
```

Once that is done, either logout / login or use:

```
source ~/.zshrc
```

And you can now use `sudo` with `AutoRecon`



```
(kali@kali02)-[~]
$ sudo autorecon
[!] You must specify at least one target to scan!
```

Usage

Single Target

A single target scan instructs `AutoRecon` to only target one host on the network.

As per below, no additional flags or parameters are required.

```
sudo autorecon 192.168.0.114
```

Multi-target, single line

This example allows you to scan multiple targets (or networks using CIDR notation)

```
sudo autorecon 192.168.0.0/24 10.10.78.180 127.0.0.1
```

Note: `AutoRecon` will expand the CIDR notation in 192.168.0.0/24 and include all hosts automatically.

Target File

A target file is handy when you want to scan a collection of targets, without a need to constantly start AutoRecon.

Create your targets file. In the example below, I have called mine `targets.txt` and it contains four targets (one per line).

```
└─(kali㉿kali02)-[~]  
└─$ cat targets.txt  
192.168.0.114  
192.168.0.115  
192.168.0.137  
192.168.0.231
```

To specify the targets file, we use the `-t` flag.

```
sudo autorecon -t targets.txt
```

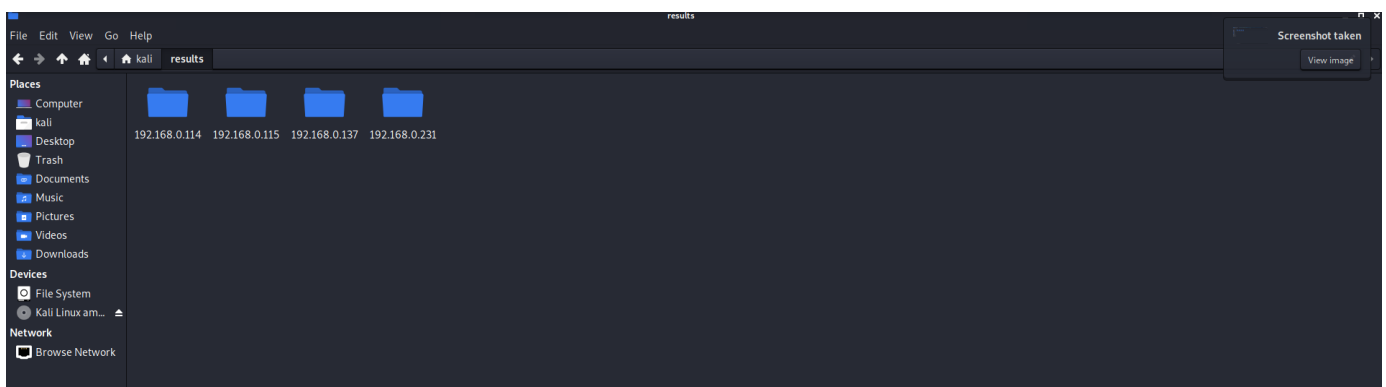
Note: remember, AutoRecon will appear to work without root privileges. Remember to run with `sudo`

Note: From the image, you can see that `AutoRecon` is `threaded`, meaning that it starts scans on multiple targets at once. This is much faster than scanning targets one at a time.

```
kali@kali02: ~  
File Actions Edit View Help  
  
(kali@kali02)-[~]  
$ sudo autorecon -t targets.txt  
[*] Scanning target 192.168.0.114  
[*] Scanning target 192.168.0.115  
[*] Running service detection nmap-top-20-udp on 192.168.0.115  
[*] Running service detection nmap-top-20-udp on 192.168.0.114  
[*] Scanning target 192.168.0.137  
[*] Running service detection nmap-top-20-udp on 192.168.0.137  
[*] Scanning target 192.168.0.231  
[*] Running service detection nmap-full-tcp on 192.168.0.114  
[*] Running service detection nmap-quick on 192.168.0.114  
[*] Running service detection nmap-top-20-udp on 192.168.0.231  
[*] Running service detection nmap-full-tcp on 192.168.0.137  
[*] Running service detection nmap-full-tcp on 192.168.0.115  
[*] Running service detection nmap-full-tcp on 192.168.0.231  
[*] Running service detection nmap-quick on 192.168.0.137  
[*] Running service detection nmap-quick on 192.168.0.115  
[*] Running service detection nmap-quick on 192.168.0.231  
[*] Service detection nmap-quick on 192.168.0.115 finished successfully in 34 seconds  
[*] Found ssh on tcp/22 on target 192.168.0.115  
[*] Found http on tcp/80 on target 192.168.0.115  
[*] Found rpcbind on tcp/111 on target 192.168.0.115  
[*] Found netbios-ssn on tcp/139 on target 192.168.0.115  
[!] [tcp/139/nbtscan] Scan cannot be run against tcp port 139. Skipping.  
[*] Found ssl/http on tcp/443 on target 192.168.0.115
```

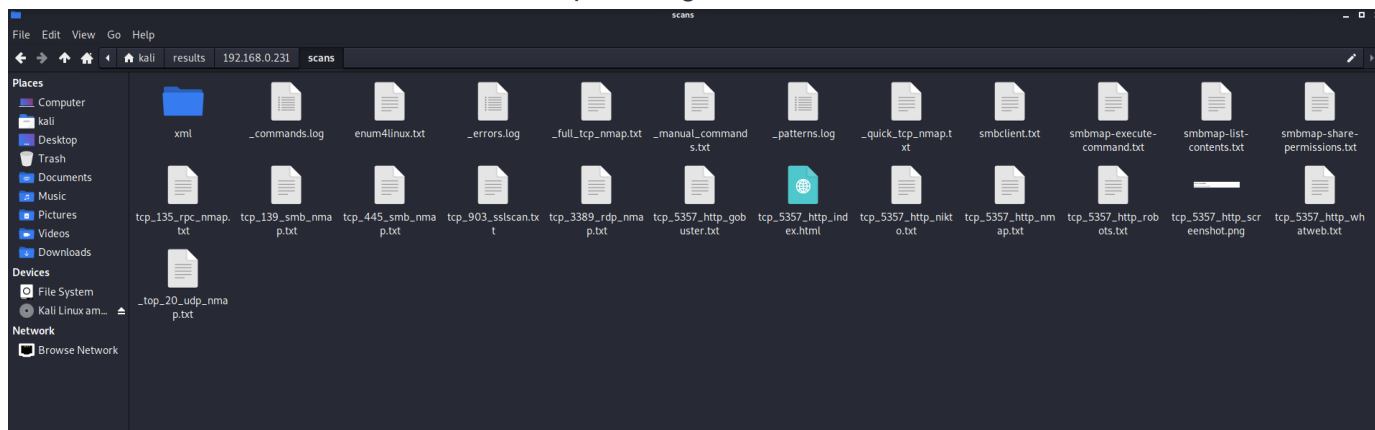
Accessing results

By default, `AutoRecon` will store the results within a folder called `results` within the folder it is run.



You can drill down on these folders to locate sub-folders, like the `scans` folder shown below. This

folder contains the results of all scans completed against identified services.



Results on a web server

If you prefer viewing the results in a web server, you can use Python. You may recall these commands from the `enumeration` module. I have included both `Python2` and `Python3`, though you should focus on `Python3` as `Python2` is End-Of-Life.

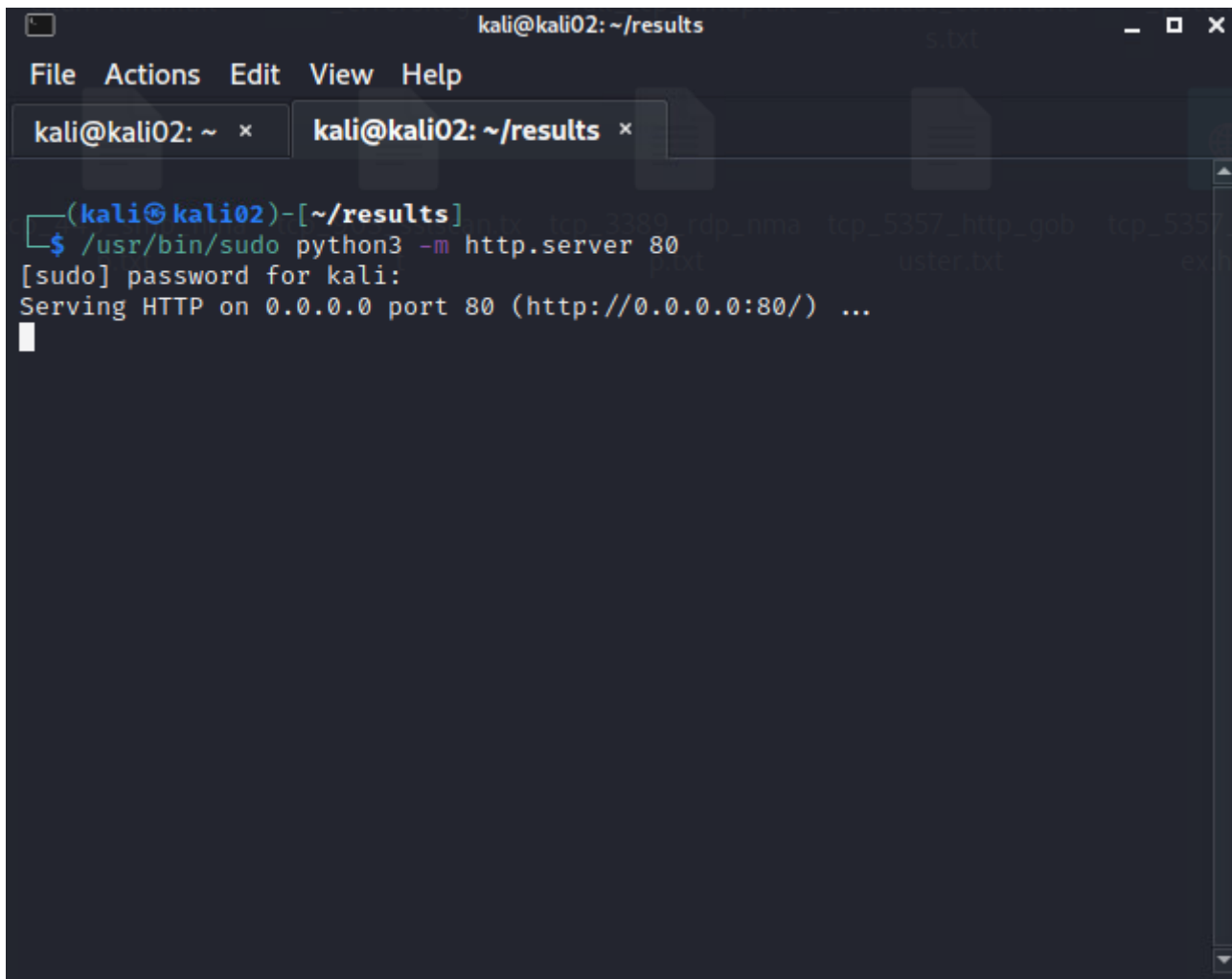
Move into the results folder, and spawn your web server of choice using Python.

Python3

```
/usr/bin/sudo python3 -m http.server 80
```

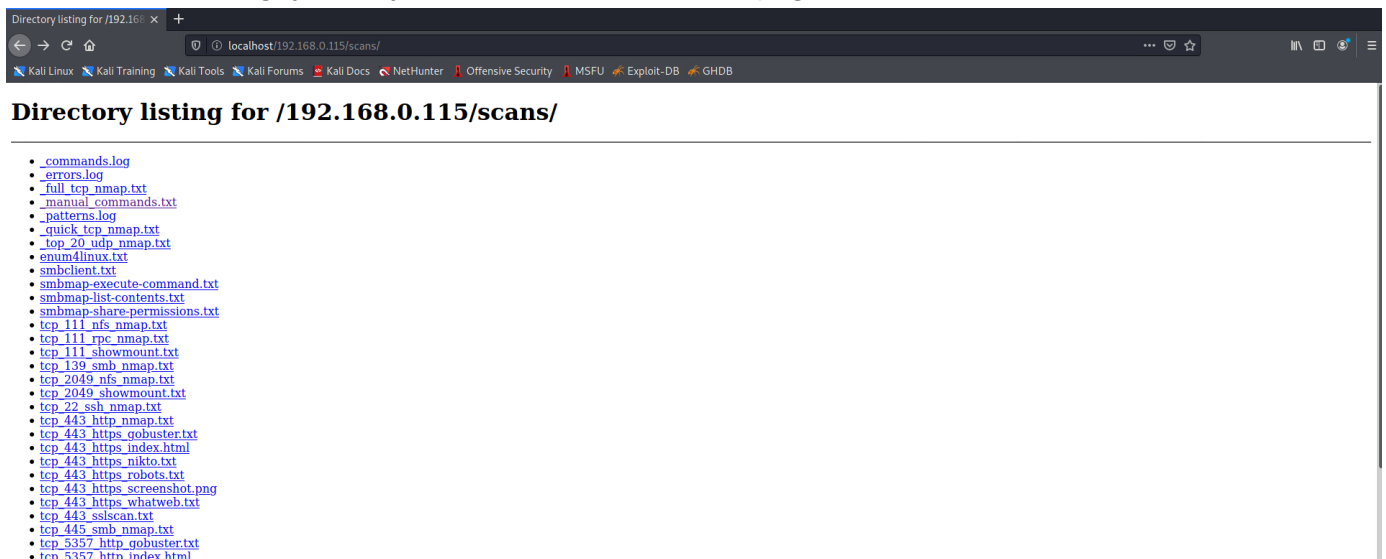
Python2

```
/usr/bin/sudo python2.7 -m SimpleHTTPServer 80
```



A terminal window titled 'kali@kali02: ~/results' with a menu bar (File, Actions, Edit, View, Help) and two tabs. The active tab shows the command prompt '(kali@kali02)-[~/results]' where the user enters `/usr/bin/sudo python3 -m http.server 80`. The prompt changes to `[sudo]` and the user enters the password 'kali'. The output is 'Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...' followed by a cursor.

Once that is running, you may view the results as a web page



You may find this easier to read than using the kali file manager.

```
localhost/192.168.0.115/scans X +
localhost/192.168.0.115/scans/_manual_commands.txt
Kali Linux Kali Training Kali Tools Kali Forums Kali Docs NetHunter Offensive Security MSFU Exploit-DB GHDB

[*] ssh on tcp/22

[-] Bruteforce logins:

hydra -L "/usr/share/seclists/Usernames/top-usernames-shortlist.txt" -P "/usr/share/seclists/Passwords/darkweb2017-top100.txt" -e nsr -s 22 -o "/home/kali/results/192.168.0.115/scans/tcp_22_ssh_hydra.txt" ssh://192.168.0.115
medusa -U "/usr/share/seclists/Usernames/top-usernames-shortlist.txt" -P "/usr/share/seclists/Passwords/darkweb2017-top100.txt" -e ns -n 22 -o "/home/kali/results/192.168.0.115/scans/tcp_22_ssh_medusa.txt" -H ssh -h 192.168.0.115

[*] http on tcp/80

[-] (dirsearch) Multi-threaded recursive directory/file enumeration for web servers using various wordlists:

dirsearch -u http://192.168.0.115:80/ -t 16 -r -e txt,html,php,asp,aspx,jsp -f -w /usr/share/seclists/Discovery/Web-Content/big.txt --plain-text-report="/home/kali/results/192.168.0.115/scans/tcp_80_http_dirsearch_big.txt"
dirsearch -u http://192.168.0.115:80/ -t 16 -r -e txt,html,php,asp,aspx,jsp -f -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt --plain-text-report="/home/kali/results/192.168.0.115/scans/tcp_80_http_dirsearch_dirbuster.txt"

[-] (dirb) Recursive directory/file enumeration for web servers using various wordlists (same as dirsearch above):

dirb http://192.168.0.115:80/ /usr/share/seclists/Discovery/Web-Content/big.txt -l -r -S -X ".txt,.html,.php,.asp,.aspx,.jsp" -o "/home/kali/results/192.168.0.115/scans/tcp_80_http_dirb_big.txt"
dirb http://192.168.0.115:80/ /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -l -r -S -X ".txt,.html,.php,.asp,.aspx,.jsp" -o "/home/kali/results/192.168.0.115/scans/tcp_80_http_dirb_dirbuster.txt"

[-] (gobuster v3) Directory/file enumeration for web servers using various wordlists (same as dirb above):

gobuster dir -u http://192.168.0.115:80/ -w /usr/share/seclists/Discovery/Web-Content/big.txt -e -k -l -s "200,204,301,302,307,403,500" -x "txt,html,php,asp,aspx,jsp" -z -o "/home/kali/results/192.168.0.115/scans/tcp_80_http_gobuster_big.txt"
gobuster dir -u http://192.168.0.115:80/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -e -k -l -s "200,204,301,302,307,403,500" -x "txt,html,php,asp,aspx,jsp" -z -o "/home/kali/results/192.168.0.115/scans/tcp_80_http_gobuster_dirbuster.txt"

[-] (gobuster v1 & v2) Directory/file enumeration for web servers using various wordlists (same as dirb above):

gobuster -u http://192.168.0.115:80/ -w /usr/share/seclists/Discovery/Web-Content/big.txt -e -k -l -s "200,204,301,302,307,403,500" -x "txt,html,php,asp,aspx,jsp" -o "/home/kali/results/192.168.0.115/scans/tcp_80_http_gobuster_big.txt"
gobuster -u http://192.168.0.115:80/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -e -k -l -s "200,204,301,302,307,403,500" -x "txt,html,php,asp,aspx,jsp" -o "/home/kali/results/192.168.0.115/scans/tcp_80_http_gobuster_dirbuster.txt"
```

Adding an Alias

You may also use the `alias` below to make hosting the web server easier. For example, typing `pyweb 80` would host a web server on port 80, using the current folder as a web root. If you add the `alias` to your `~/.zshrc`, be sure to run `source ~/.zshrc` and logout / login before you try and use it.

```
alias pyweb="/usr/bin/sudo python3 -m http.server"
```

Once the alias is `sourced`, you can call it with `pyweb 1337` to host a web server of the current folder on port `1337`.

Assessment

Nil.