



# AutoRecon

## Network Reconnaissance

**Original Author:** *Pavandeep Singh*

**Tool's Author:** *Tib3rius*



## Table of Contents

Abstract.....	3
Introduction.....	4
<b>Features</b> .....	4
<b>Installation</b> .....	4
<b>Usage</b> .....	7
<b>Multi-Target Scan</b> .....	9
<b>Concurrent Scan</b> .....	10
<b>Single Target Argument</b> .....	11
<b>Heartbeat Argument</b> .....	13
<b>Nmap Arguments</b> .....	13
<b>Verbosity Scans</b> .....	16
<b>Only Scans Dir Argument</b> .....	17
<b>Results</b> .....	19
<b>Enum4Linux Scan</b> .....	22
<b>SMBMap Scan</b> .....	23
<b>SMTP Scan</b> .....	24
<b>WhatWeb Scan</b> .....	25
<b>XML Results</b> .....	26
<b>Nikto Scan</b> .....	27
<b>Web Services Screenshots</b> .....	28
Conclusion .....	29
References .....	29



## Abstract

The AutoRecon tool is designed as a network reconnaissance tool. It is a multi-threaded tool that performs automated enumeration of services. The purpose of this tool is to save time while cracking CTFs and other penetration testing environments or exams. It is useful in real-world engagements as well.

The purpose of this report is to shift the focus on one of the finest reconnaissance tools for penetration testing.

**Disclaimer: This report is provided for educational and informational purpose only (Penetration Testing). Penetration Testing refers to legal intrusion tests that aim to identify vulnerabilities and improve cybersecurity, rather than for malicious purposes.**



## Introduction

The AutoRecon works in a sequential process. Initially, it performs port scans or service detection scans. Then using the results of these scans as a reference it further launches enumeration scans of those services using other tools. For example, if HTTP is found, it will check for webpages and if it will get those, it will start Nikto scan with go buster and other tools concurrently. The tool itself is designed to be personalized by making changes in the configuration files.

## Features

- It uses pattern matching to increase the speed and accuracy in results.
- It logs all the commands that it executes so that the user can check in case of errors.
- It supports multiple targets at once. It uses IP Addresses, or IP Ranges and resolvable hostname.
- It supports customizable enumerations on different services.

## Installation

There are multiple methods to install AutoRecon. Users can run it on a Docker Instance. It can be used by a small fraction of people that don't use Kali Linux or any other Linux Distribution. To install using Docker please refer to the GitHub page of the AutoRecon. We are focusing on the installation methods with pipx. It is the recommended method by the developer of the tool and it works like charm. Before Beginning the Installation there are a few requirements in order to run AutoRecon. The AutoRecon is built on Python 3. It will not run on Python 2. This is because it uses a library called asyncio which is not supported in earlier versions. If you don't already have it installed, you can do it by running the following commands as root. We are installing the pip for python3 as well to install any additional packages.

```
apt install python3  
apt install python3-pip
```

After this we will use pipx, to manage the python packages; This will take the packages that we install and put them inside their own virtual environment. This will avoid any conflicting packages inside your machine. First, we will be installing the Virtual Environment using Python3.

```
apt install python3-venv
```

```
(root@kali)~[~/Desktop]
# apt install python3-venv
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  python-pip-whl python3.9-venv
The following NEW packages will be installed:
  python-pip-whl python3-venv python3.9-venv
0 upgraded, 3 newly installed, 0 to remove and 29 not upgraded.
Need to get 1,954 kB of archives.
After this operation, 2,324 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ftp.harukasan.org/kali kali-rolling/main amd64 python-pip-whl all 20.3.4-1 [1,947 kB]
```

Now we will download the pipx using python3-pip

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

```
(root@kali)~[~/Desktop]
# python3 -m pip install --user pipx
Collecting pipx
  Downloading pipx-0.16.1.0-py3-none-any.whl (51 kB)
    | 51 kB 1.5 MB/s
Collecting userpath>=1.4.1
  Downloading userpath-1.4.2-py2.py3-none-any.whl (14 kB)
Requirement already satisfied: colorama>=0.4.4 in /usr/lib/python3/dist-packages (from pipx) (0.4.4)
Collecting argcomplete<2.0, >=1.9.4
  Downloading argcomplete-1.12.2-py2.py3-none-any.whl (38 kB)
Requirement already satisfied: packaging>=20.0 in /usr/lib/python3/dist-packages (from pipx) (20.9)
Requirement already satisfied: distro in /usr/lib/python3/dist-packages (from userpath>=1.4.1->pipx) (1.5.0)
Requirement already satisfied: click in /usr/lib/python3/dist-packages (from userpath>=1.4.1->pipx) (7.1.2)
```

Now we will make changes to the PATH Variables to add pipx. After completion, we will need to reopen the terminal so that it can take effect.

```
python3 -m pipx ensurepath
```

```
(root@kali)-[~/Desktop]
# python3 -m pipx ensurepath
Success! Added /root/.local/bin to the PATH environment variable.

Consider adding shell completions for pipx. Run 'pipx completions' for
instructions.

You will need to open a new terminal or re-login for the PATH changes to
take effect.

Otherwise pipx is ready to go! ✨ ✨ ✨
```

AutoRecon takes a lot of different tools and runs them on the target defined by the user. Although most of the tools are present in Kali itself let's install or update those tools to use AutoRecon in its full glory. As the seclists is quite large in size more like 386 Megabytes so that is going to take some time.

The list of tools that required are:

- curl
- enum4linux
- gobuster
- nbtscan
- Nikto
- nmap
- onesixtyone
- oscanner
- smbclient
- SMBMap
- smtp-user-enum
- SNMP walk
- sslscan
- svwar
- tnscommand10g
- whatweb
- wkhtmltopdf

The command to install or update these tools is mentioned below:

```
apt install seclists curl enum4linux gobuster nbtscan nikto
nmap onesixtyone oscanner smbclient smbmap smtp-user-enum
snmp sslscan sipvicious tnscommand10g whatweb wkhtmltopdf
```

```
(root@kali)-[~/Desktop]
# apt install seclists curl enum4linux gobuster nbtscan n
cious tnsnmd10g whatweb wkhtmltopdf
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
curl is already the newest version (7.74.0-1.1).
curl set to manually installed.
enum4linux is already the newest version (0.8.9-1kali3).
enum4linux set to manually installed.
nbtscan is already the newest version (1.6-3).
nbtscan set to manually installed.
```

Now that all the pre-requisites are installed, all that is needed to do is install the AutoRecon directly from its GitHub Repository using pipx. Very simple syntax to use as it can be observed.

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

```
(root@kali)-[~/Desktop]
# pipx install git+https://github.com/Tib3rius/AutoRecon.git
installed package autorecon 1.0.0, Python 3.9.2
These apps are now globally available
- autorecon
done! 🌟 🌟 🌟
```

## Usage

As discussed earlier, AutoRecon is an Enumeration tool. It requires a target or a set of targets. This can be IP Addresses, or CIDR Notations or hostnames as well. When triggered with the -h parameter it shows the user a help screen as depicted in the image below. It tells us that we can provide a target directly or if you have multiple targets. Then you can put the target IP Addresses into a file and then pass it as a parameter with the -t flag.

```
autorecon -h
```

```
(root@kali)~# autorecon -h
usage: autorecon [-h] [-t TARGET_FILE] [-ct <number>] [-cs <number>] [--profile PROFILE_NAME]
               [--heartbeat HEARTBEAT] [--nmap NMAP] [--nmap-append NMAP_APPEND] [-o OUTPUT DIR]
               [targets...]

Network reconnaissance tool to port scan and automatically enumerate services found on targets.

positional arguments:
  targets                IP addresses (e.g. 10.0.0.1), CIDR notation (e.g. 10.0.0.1/24) or resolvable hostnames

optional arguments:
  -h, --help            show this help message and exit
  -t TARGET_FILE, --targets TARGET_FILE
                        Read targets from file.
  -ct <number>, --concurrent-targets <number>
                        The maximum number of target hosts to scan concurrently. Default is 10.
  -cs <number>, --concurrent-scans <number>
                        The maximum number of scans to perform per target host. Default is 1.
  --profile PROFILE_NAME
                        The port scanning profile to use (defined in port-scan-profiles directory).
  -o OUTPUT DIR, --output OUTPUT DIR
                        Output directory.
```

As we saw earlier that AutoRecon has a large number of parameters but most of these can be left default. The key thing to remember is the required argument “target”. It can be a space-separated list of either IP Addresses or CIDR Notations or even resolvable hostnames. We can also create a file with the targets in it. It should be in the format of one per new line. We need to reference that target file using the -t argument.

```
autorecon 192.168.126.132
```





```
(root@kali)-[~]
# autorecon 192.168.126.132
[*] Scanning target 192.168.126.132
[*] Running service detection nmap-top-20-udp on 192.168.126.132
[*] Running service detection nmap-quick on 192.168.126.132
[*] Running service detection nmap-full-tcp on 192.168.126.132
[*] Service detection nmap-quick on 192.168.126.132 finished success
[*] Found ftp on tcp/21 on target 192.168.126.132
[*] Found ssh on tcp/22 on target 192.168.126.132
[*] Found telnet on tcp/23 on target 192.168.126.132
[*] Found smtp on tcp/25 on target 192.168.126.132
[*] Found domain on tcp/53 on target 192.168.126.132
[*] Found http on tcp/80 on target 192.168.126.132
[*] Found rpcbind on tcp/111 on target 192.168.126.132
[*] Found netbios-ssn on tcp/139 on target 192.168.126.132
[!] [tcp/139/nbtscan] Scan cannot be run against tcp port 139. Skip
[*] Found netbios-ssn on tcp/445 on target 192.168.126.132
[!] [tcp/445/enum4linux on 192.168.126.132] Scan should only be run
[!] [tcp/445/nbtscan] Scan cannot be run against tcp port 445. Skip
[!] [tcp/445/smbclient on 192.168.126.132] Scan should only be run
[*] Found exec on tcp/512 on target 192.168.126.132
[*] Found login on tcp/513 on target 192.168.126.132
[*] Found tcpwrapped on tcp/514 on target 192.168.126.132
[*] Found java-rmi on tcp/1099 on target 192.168.126.132
[*] Found bindshell on tcp/1524 on target 192.168.126.132
[*] Found nfs on tcp/2049 on target 192.168.126.132
[*] Found ftp on tcp/2121 on target 192.168.126.132
[*] Found mysql on tcp/3306 on target 192.168.126.132
[*] Found postgresql on tcp/5432 on target 192.168.126.132
[*] Found vnc on tcp/5900 on target 192.168.126.132
[*] Found X11 on tcp/6000 on target 192.168.126.132
[*] Found irc on tcp/6667 on target 192.168.126.132
[*] Found rdp on tcp/3389 on target 192.168.126.132
```

## Multi-Target Scan

By default, AutoRecon will scan 5 target hosts at the same time but that number can be toggled using the -ct parameter. This is basically the number of targets getting scanned at the same time. To demonstrate, we collected some IP Address in the network and then entered them into a text file. Then used that text file to provide targets to AutoRecon.

```
cat target.txt
autorecon -t targets.txt
```

```
(root@kali)~# cat targets.txt
192.168.126.1
192.168.126.2
192.168.126.254

www.hackingarticles.in

(root@kali)~# autorecon -t targets.txt
[*] Scanning target 192.168.126.1
[*] Scanning target 192.168.126.2
[*] Scanning target 192.168.126.254
[*] Running service detection nmap-top-20-udp on 192.168.126.1
[*] Running service detection nmap-top-20-udp on 192.168.126.254
[*] Running service detection nmap-top-20-udp on 192.168.126.2
[*] Running service detection nmap-quick on 192.168.126.1
[*] Running service detection nmap-quick on 192.168.126.254
[*] Running service detection nmap-quick on 192.168.126.2
[*] Running service detection nmap-full-tcp on 192.168.126.1
[*] Running service detection nmap-quick on 192.168.126.2
[*] Running service detection nmap-full-tcp on 192.168.126.254
[*] Running service detection nmap-full-tcp on 192.168.126.2
```

## Concurrent Scan

Another parameter to look at is the `-cs` which is the Concurrent Scans. This is basically the number of scans that are being performed per target. By default, the setting is set to 10. When changed to any other value such as 2 then only 2 scans will be performed per host. Once it is finished it will run another instance of the scan.

Hence, each of the targets that are being scanned will at least have 3 nmap scans running, basically a full TCP, a top 1000 TCP and a top 20 UDP.

```
autorecon -cs 5 192.168.126.132
```

```
(root@kali)-[~]
# autorecon -cs 5 192.168.126.132
[*] Scanning target 192.168.126.132
[*] Running service detection nmap-top-20-udp on 192.168.126.132
[*] Running service detection nmap-quick on 192.168.126.132
[*] Running service detection nmap-full-tcp on 192.168.126.132
[*] Service detection nmap-quick on 192.168.126.132 finished success
[*] Found ftp on tcp/21 on target 192.168.126.132
[*] Found ssh on tcp/22 on target 192.168.126.132
[*] Found telnet on tcp/23 on target 192.168.126.132
[*] Found smtp on tcp/25 on target 192.168.126.132
[*] Found domain on tcp/53 on target 192.168.126.132
[*] Found http on tcp/80 on target 192.168.126.132
[*] Found rpcbind on tcp/111 on target 192.168.126.132
[*] Found netbios-ssn on tcp/139 on target 192.168.126.132
[!] [tcp/139/nbtscan] Scan cannot be run against tcp port 139. Skipp
[*] Found netbios-ssn on tcp/445 on target 192.168.126.132
[!] [tcp/445/enum4linux on 192.168.126.132] Scan should only be run
[!] [tcp/445/nbtscan] Scan cannot be run against tcp port 445. Skipp
[!] [tcp/445/smbclient on 192.168.126.132] Scan should only be run o
[*] Found exec on tcp/512 on target 192.168.126.132
[*] Found login on tcp/513 on target 192.168.126.132
[*] Found tcpwrapped on tcp/514 on target 192.168.126.132
[*] Found java-rmi on tcp/1099 on target 192.168.126.132
[*] Found bindshell on tcp/1524 on target 192.168.126.132
[*] Found nfs on tcp/2049 on target 192.168.126.132
[*] Found ftp on tcp/2121 on target 192.168.126.132
```

## Single Target Argument

The `--single-target` argument enables the users to scan the host but changing the directory structure. It means that the AutoRecon will only scan the target but no directory will be created for that particular target.

```
autorecon 192.168.126.133 --single-target
```

```
(root@kali)-[~]
# autorecon 192.168.126.133 --single-target
[*] Scanning target 192.168.126.133
[*] Running service detection nmap-quick on 192.168.126.133
[*] Running service detection nmap-top-20-udp on 192.168.126.133
[*] Running service detection nmap-full-tcp on 192.168.126.133
[*] Service detection nmap-quick on 192.168.126.133 finished successful
[*] Found ftp on tcp/21 on target 192.168.126.133
[*] Found ssh on tcp/22 on target 192.168.126.133
[*] Found telnet on tcp/23 on target 192.168.126.133
[*] Found smtp on tcp/25 on target 192.168.126.133
[*] Found domain on tcp/53 on target 192.168.126.133
[*] Found http on tcp/80 on target 192.168.126.133
[*] Found rpcbind on tcp/111 on target 192.168.126.133
```

Due to the use of the `--single-target` parameter, it didn't create a directory by the name of the target inside the results folder.

```
ls -la results
cat results/report/notes.txt
```

```
(root@kali)-[~]
# ls -la results
total 24
drwxr-xr-x  6 root root 4096 Mar 22 11:40 .
drwx----- 16 root root 4096 Mar 22 11:40 ..
drwxr-xr-x  2 root root 4096 Mar 22 11:40 exploit
drwxr-xr-x  2 root root 4096 Mar 22 11:40 loot
drwxr-xr-x  3 root root 4096 Mar 22 11:41 report
drwxr-xr-x  3 root root 4096 Mar 22 11:41 scans

(root@kali)-[~]
# cat results/report/notes.txt
[*] ftp found on tcp/21.

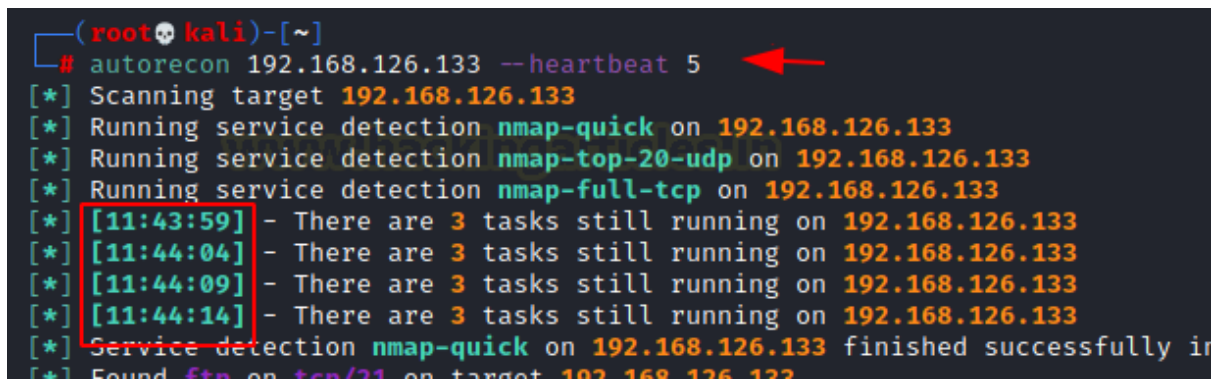
[*] ssh found on tcp/22.

[*] telnet found on tcp/23.
```

## Heartbeat Argument

The `--heartbeat` argument allows the users to configure the duration of the updates that are provided by AutoRecon. By default, it is 60 seconds. It means AutoRecon will update the user what is going on and which scans are running every 60 seconds.

```
autorecon 192.168.126.133 --heartbeat 5
```



```
(root@kali)-[~]
# autorecon 192.168.126.133 --heartbeat 5
[*] Scanning target 192.168.126.133
[*] Running service detection nmap-quick on 192.168.126.133
[*] Running service detection nmap-top-20-udp on 192.168.126.133
[*] Running service detection nmap-full-tcp on 192.168.126.133
[*] [11:43:59] - There are 3 tasks still running on 192.168.126.133
[*] [11:44:04] - There are 3 tasks still running on 192.168.126.133
[*] [11:44:09] - There are 3 tasks still running on 192.168.126.133
[*] [11:44:14] - There are 3 tasks still running on 192.168.126.133
[*] Service detection nmap-quick on 192.168.126.133 finished successfully in
[*] Found ftp on tcp/21 on target 192.168.126.133
```

## Nmap Arguments

Now here we have two options, we can either replace our own parameters instead of the ones that are provided here, by using the `--nmap` argument and passing the parameters that we want to perform.

```
autorecon 192.168.126.133 --nmap sV
```

```
(root@kali)-[~]
# autorecon 192.168.126.133 --nmap sV
[*] Scanning target 192.168.126.133
[*] Running service detection nmap-top-20-udp on 192.168.126.133
[*] Running service detection nmap-quick on 192.168.126.133
[*] Running service detection nmap-full-tcp on 192.168.126.133
[*] Service detection nmap-quick on 192.168.126.133 finished success
[*] Found ftp on tcp/21 on target 192.168.126.133
[*] Found ssh on tcp/22 on target 192.168.126.133
[*] Found telnet on tcp/23 on target 192.168.126.133
[*] Found smtp on tcp/25 on target 192.168.126.133
[*] Found domain on tcp/53 on target 192.168.126.133
[*] Found http on tcp/80 on target 192.168.126.133
[*] Found rpcbind on tcp/111 on target 192.168.126.133
[*] Found netbios-ssn on tcp/139 on target 192.168.126.133
[!] [tcp/139/nbtscan] Scan cannot be run against tcp port 139. Skip
```

In the previous step, we added the `-sV` argument to the nmap scan, now in order to check we will read the `commands.log` file to see that it indeed uses the `-sV` parameter while scanning. It should also be noted that the default parameters `-vv`, `-reason`, `-Pn` are not used.

```
cat results/192.168.126.133/scans/_commands.log
```

```
(root@kali)-[~]
# cat results/192.168.126.133/scans/_commands.log
nmap -sV -sU -A --top-ports=20 --version-all -oN "/root/results/192.168.126.133/scans/_auto_recon_commands.log" 192.168.126.133
nmap -sV -sV -sC --version-all -oN "/root/results/192.168.126.133/scans/_quick_tcp_nmap_commands.log" 192.168.126.133
nmap -sV -A --osscan-guess --version-all -p- -oN "/root/results/192.168.126.133/scans/_osscan_commands.log" 192.168.126.133
if [ "False" = "True" ]; then sslscan --show-certificate --no-colour 192.168.126.133
nmap -sV -sV -p 21 --script="banner,(ftp* or ssl*)" and not (brute or broadcast or do
txt" -oX "/root/results/192.168.126.133/scans/xml/tcp_21_ftp_nmap.xml" 192.168.126.133
```

We can use the `--nmap-append` option to add our parameters but not override the AutoRecon default parameters. It will append our parameters to it.

```
autorecon 192.168.126.133 --nmap-append sS
```





```
(root@kali)-[~]
# autorecon 192.168.126.133 --nmap-append sS
[*] Scanning target 192.168.126.133
[*] Running service detection nmap-top-20-udp on 192.168.126.133
[*] Running service detection nmap-quick on 192.168.126.133
[*] Running service detection nmap-full-tcp on 192.168.126.133
[*] Service detection nmap-quick on 192.168.126.133 finished successfully in 39 s
[*] Found ftp on tcp/21 on target 192.168.126.133
[*] Found ssh on tcp/22 on target 192.168.126.133
[*] Found telnet on tcp/23 on target 192.168.126.133
[*] Found smtp on tcp/25 on target 192.168.126.133
[*] Found domain on tcp/53 on target 192.168.126.133
[*] Found http on tcp/80 on target 192.168.126.133
[*] Found rpcbind on tcp/111 on target 192.168.126.133
```

Let's again check if the argument we added i.e., -sS has been appended with -vv, --reason, -Pn. It can be confirmed from a detailed read of the commands.log file.

```
cat results/192.168.126.133/scans/_commands.log
```

```
(root@kali)-[~]
# cat results/192.168.126.133/scans/_commands.log
nmap -vv --reason -Pn sS -sU -A --top-ports=20 --version-all --xml/_top_20_udp_nmap.xml" 192.168.126.133

nmap -vv --reason -Pn sS -sV -sC --version-all -oN "/root/.nmap.xml" 192.168.126.133

nmap -vv --reason -Pn sS -A --osscan-guess --version-all --xml/_full_tcp_nmap.xml" 192.168.126.133

if [ "False" = "True" ]; then sslscan --show-certificate

nmap -vv --reason -Pn sS -sV -p 21 --script="banner,(ftp* /tcp_21_ftp_nmap.txt" -oX "/root/results/192.168.126.133/s

if [ "False" = "True" ]; then sslscan --show-certificate

nmap -vv --reason -Pn sS -sV -p 22 --script="banner,ssh2- /root/results/192.168.126.133/scans/xml/tcp_22_ssh_nmap.xml
```

## Verbosity Scans

AutoRecon has different levels of verbosity. By default, it doesn't run with any verbosity that means it just informs the user when it initiates a scan and when the scan finishes, it does not provide any details regarding those tasks. With the `-v` argument, it will be telling the user more about the scans like it will show the complete commands it is running, it will also provide more information about the services that were detected and are being further enumerated.

```
autorecon -v 192.168.154.130
```

```
(root@kali)-[~]
└─# autorecon -v 192.168.154.130
[*] Scanning target 192.168.154.130
[*] Running service detection nmap-top-20-udp on 192.168.154.130 with nmap -vv --reason -Pn -sU -A --top-ports=20 --version-all -oN "/root/results/192.168.154.130/scans/_top_20_udp_nmap.txt" -oX "/root/results/192.168.154.130/scans/xml/_top_20_udp_nmap.xml" 192.168.154.130
[*] Running service detection nmap-quick on 192.168.154.130 with nmap -vv --reason -Pn -sV -sC --version-all -oN "/root/results/192.168.154.130/scans/_quick_tcp_nmap.txt" -oX "/root/results/192.168.154.130/scans/xml/_quick_tcp_nmap.xml" 192.168.154.130
[*] Running service detection nmap-full-tcp on 192.168.154.130 with nmap -vv --reason -Pn -A --osscan-guess --version-all -p- -oN "/root/results/192.168.154.130/scans/_full_tcp_nmap.txt" -oX "/root/results/192.168.154.130/scans/xml/_full_tcp_nmap.xml" 192.168.154.130
[*] Service detection nmap-quick on 192.168.154.130 finished successfully in 23 seconds
[*] Found ftp on tcp/21 on target 192.168.154.130
[*] Found ssh on tcp/22 on target 192.168.154.130
[*] Found telnet on tcp/23 on target 192.168.154.130
[*] Found smtp on tcp/25 on target 192.168.154.130
[*] Found domain on tcp/53 on target 192.168.154.130
[*] Found http on tcp/80 on target 192.168.154.130
[*] Found rpcbind on tcp/111 on target 192.168.154.130
[*] Found netbios-ssn on tcp/139 on target 192.168.154.130
[!] [tcp/139/nbtscan] Scan cannot be run against tcp port 139. Skipping.
[*] Found netbios-ssn on tcp/445 on target 192.168.154.130
[!] [tcp/445/enum4linux on 192.168.154.130] Scan should only be run once and it appears to have already been queued. Skipping.
[!] [tcp/445/nbtscan] Scan cannot be run against tcp port 445. Skipping.
[!] [tcp/445/smbclient on 192.168.154.130] Scan should only be run once and it appears to have already been queued. Skipping.
[*] Found exec on tcp/512 on target 192.168.154.130
[*] Found login on tcp/513 on target 192.168.154.130
[*] Found tcpwrapped on tcp/514 on target 192.168.154.130
[*] Found java-rmi on tcp/1099 on target 192.168.154.130
[*] Found bindshell on tcp/1524 on target 192.168.154.130
[*] Found nfs on tcp/2049 on target 192.168.154.130
[*] Found ftp on tcp/2121 on target 192.168.154.130
[*] Found mysql on tcp/3306 on target 192.168.154.130
[*] Found postgresql on tcp/5432 on target 192.168.154.130
[*] Found vnc on tcp/5900 on target 192.168.154.130
[*] Found X11 on tcp/6000 on target 192.168.154.130
[*] Found irc on tcp/6667 on target 192.168.154.130
```

You can also use `-vv` which stands for Very Verbose. It is not recommended as it will print all the scans and their results in real-time. It clutters up the screen with too much information.





```
autorecon -vv 192.168.154.130
```

```
(root@kali)-[~]
# autorecon -vv 192.168.154.130
[*] Scanning target 192.168.154.130
[*] Running service detection nmap-top-20-udp on 192.168.154.130 with nmap -vv --reason -Pn -sU -A --top-20-udp --xml-output-dir /_top_20_udp_nmap.xml" 192.168.154.130
[*] Running service detection nmap-quick on 192.168.154.130 with nmap -vv --reason -Pn -sV -sC --version-intensity 10 192.168.154.130
[*] Running service detection nmap-full-tcp on 192.168.154.130 with nmap -vv --reason -Pn -A --osscan-guess --xml-output-dir /_full_tcp_nmap.xml" 192.168.154.130
[-] [192.168.154.130 nmap-top-20-udp] Host discovery disabled (-Pn). All addresses will be marked 'up' and scanned.
[-] [192.168.154.130 nmap-quick] Host discovery disabled (-Pn). All addresses will be marked 'up' and scanned.
[-] [192.168.154.130 nmap-full-tcp] Host discovery disabled (-Pn). All addresses will be marked 'up' and scanned.
[-] [192.168.154.130 nmap-quick] Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-21 08:19 EDT
[-] [192.168.154.130 nmap-top-20-udp] Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-21 08:19 EDT
[-] [192.168.154.130 nmap-full-tcp] Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-21 08:19 EDT
[-] [192.168.154.130 nmap-quick] NSE: Loaded 153 scripts for scanning.
[-] [192.168.154.130 nmap-quick] NSE: Script Pre-scanning.
[-] [192.168.154.130 nmap-quick] NSE: Starting runlevel 1 (of 3) scan.
[-] [192.168.154.130 nmap-quick] Initiating NSE at 08:19
[-] [192.168.154.130 nmap-quick] Completed NSE at 08:19, 0.00s elapsed
[-] [192.168.154.130 nmap-quick] NSE: Starting runlevel 2 (of 3) scan.
[-] [192.168.154.130 nmap-quick] Initiating NSE at 08:19
[-] [192.168.154.130 nmap-quick] Completed NSE at 08:19, 0.00s elapsed
[-] [192.168.154.130 nmap-quick] NSE: Starting runlevel 3 (of 3) scan.
[-] [192.168.154.130 nmap-quick] Initiating NSE at 08:19
[-] [192.168.154.130 nmap-quick] Completed NSE at 08:19, 0.00s elapsed
[-] [192.168.154.130 nmap-quick] Initiating ARP Ping Scan at 08:19
[-] [192.168.154.130 nmap-quick] Scanning 192.168.154.130 [1 port]
[-] [192.168.154.130 nmap-quick] Completed ARP Ping Scan at 08:19, 0.05s elapsed (1 total hosts)
[-] [192.168.154.130 nmap-quick] Initiating Parallel DNS resolution of 1 host. at 08:19
[-] [192.168.154.130 nmap-quick] Completed Parallel DNS resolution of 1 host. at 08:19, 0.00s elapsed
[-] [192.168.154.130 nmap-quick] Initiating SYN Stealth Scan at 08:19
[-] [192.168.154.130 nmap-quick] Scanning 192.168.154.130 [1000 ports]
[-] [192.168.154.130 nmap-quick] Discovered open port 80/tcp on 192.168.154.130
[-] [192.168.154.130 nmap-quick] Discovered open port 111/tcp on 192.168.154.130
[-] [192.168.154.130 nmap-quick] Discovered open port 3306/tcp on 192.168.154.130
[-] [192.168.154.130 nmap-quick] Discovered open port 21/tcp on 192.168.154.130
[-] [192.168.154.130 nmap-quick] Discovered open port 23/tcp on 192.168.154.130
[-] [192.168.154.130 nmap-quick] Discovered open port 139/tcp on 192.168.154.130
[-] [192.168.154.130 nmap-quick] Discovered open port 5900/tcp on 192.168.154.130
[-] [192.168.154.130 nmap-quick] Discovered open port 445/tcp on 192.168.154.130
[-] [192.168.154.130 nmap-quick] Discovered open port 53/tcp on 192.168.154.130
[-] [192.168.154.130 nmap-quick] Discovered open port 25/tcp on 192.168.154.130
```

## Only Scans Dir Argument

AutoRecon creates a bunch of directories based on the type of evidence it collects. But there are some situations where all that is required is the scan results. This is where the Only Scans Dir argument comes into action. This prevents the creation of other directories.

```
autorecon 192.168.154.130 --only-scans-dir
```



```
(root@kali)-[~]
# autorecon 192.168.126.133 --only-scans-dir
[*] Scanning target 192.168.126.133
[*] Running service detection nmap-top-20-udp on 192.168.126.133
[*] Running service detection nmap-quick on 192.168.126.133
[*] Running service detection nmap-full-tcp on 192.168.126.133
[*] Service detection nmap-quick on 192.168.126.133 finished successfully
[*] Found ftp on tcp/21 on target 192.168.126.133
[*] Found ssh on tcp/22 on target 192.168.126.133
[*] Found telnet on tcp/23 on target 192.168.126.133
[*] Found smtp on tcp/25 on target 192.168.126.133
[*] Found domain on tcp/53 on target 192.168.126.133
[*] Found http on tcp/80 on target 192.168.126.133
[*] Found rpcbind on tcp/111 on target 192.168.126.133
[*] Found netbios-ssn on tcp/139 on target 192.168.126.133
[!] [tcp/139/nbtscan] Scan cannot be run against tcp port 139. Skipping.
[*] Found netbios-ssn on tcp/445 on target 192.168.126.133
[!] [tcp/445/enum4linux on 192.168.126.133] Scan should only be run once a
```

Here, we can see that inside the results/target directory we have only one directory by the name of scan which will contain the scan results. Talking about the results, let's discuss the results that AutoRecon produces in detail.

```
ls -la results
```

```
ls -la results/192.168.126.133
```

```
(root@kali)-[~]
# ls -la results
total 12
drwxr-xr-x  3 root root 4096 Mar 22 11:45 .
drwxr-xr-x 16 root root 4096 Mar 22 11:45 ..
drwxr-xr-x  3 root root 4096 Mar 22 11:45 192.168.126.133

(root@kali)-[~]
# ls -la results/192.168.126.133
total 12
drwxr-xr-x 3 root root 4096 Mar 22 11:45 .
drwxr-xr-x 3 root root 4096 Mar 22 11:45 ..
drwxr-xr-x 3 root root 4096 Mar 22 11:45 scans

(root@kali)-[~]
#
```



## Results

AutoRecon when initiated with a scan, it creates a result directory. The name of the directory can be configured using the `-o` parameter. If no parameter is mentioned, it will create the results directory in the current folder. Inside the results directory, it will divide into the different targets. Suppose you scanned like 4 IP Addresses; it will create 4 directories with the IP Address as name. You could go inside any one of them to find different directories created according to the nature of the finding. So, the exploit directory would have any particular exploit that the target is vulnerable to. Although keep in mind that the exploit will have to be surely working. It means that it won't show up if there is some suspicion that the exploit will work or not. The absolute surety will create entries inside that directory. Then we have the loot directory it will be anything the AutoRecon grabbed from the host machine. Repot would be the stuff that could go into a Penetration Testing Report for example notes etc.

```
ls -la | grep results
cd results
cd 192.168.126.132
tree
```

```
(root@kali)~#  
# ls -la | grep results  
drwxr-xr-x 3 root root 4096 Mar 12 06:11 results  
  
(root@kali)~#  
# cd results  
  
(root@kali)~/results#  
# ls  
192.168.126.132  
  
(root@kali)~/results#  
# cd 192.168.126.132  
  
(root@kali)~/results/192.168.126.132#  
# tree  
.  
├── exploit  
├── loot  
├── report  
│   ├── local.txt  
│   ├── notes.txt  
│   ├── proof.txt  
│   └── screenshots  
└── scans  
    ├── _commands.log  
    ├── enum4linux.txt  
    ├── errors.log  
    ├── full_tcp_nmap.txt  
    ├── _manual_commands.txt  
    ├── _patterns.log  
    ├── _quick_tcp_nmap.txt  
    ├── smbclient.txt  
    ├── smbmap-execute-command.txt  
    ├── smbmap-list-contents.txt  
    ├── smbmap-share-permissions.txt  
    ├── tcp_1099_rmi_nmap.txt  
    └── tcp_111_nfs_nmap.txt
```

We can see that the notes inside the report folder contain the basic findings that were detected by the Nmap. It shows different services that were found running on the target application using AutoRecon. It can be used as a kind of quick reference guide.

```
cat ~/results/192.168.126.132/report/notes.txt
```

```
(root@kali)-[~/results/192.168.126.132]
# cat report/notes.txt
[*] ftp found on tcp/21.
[*] ssh found on tcp/22.
[*] telnet found on tcp/23.
[*] smtp found on tcp/25.
[*] domain found on tcp/53.
[*] http found on tcp/80.
```

If we look further, we have the full nmap scan report.

```
cat ~/results/192.168.126.132/scans/_full_tcp_nmap.txt
```



```
(root@kali)~[~/results/192.168.126.132]
# cat scans/ full tcp nmap.txt
# Nmap 7.91 scan initiated Fri Mar 12 06:12:28 2021 as: nmap -vv --reason -Pn -A --o
ap.txt -oX /root/results/192.168.126.132/scans/xml/_full_tcp_nmap.xml 192.168.126.13
adjust_timeouts2: packet supposedly had rtt of -377990 microseconds. Ignoring time.
adjust_timeouts2: packet supposedly had rtt of -377990 microseconds. Ignoring time.
adjust_timeouts2: packet supposedly had rtt of -378606 microseconds. Ignoring time.
adjust_timeouts2: packet supposedly had rtt of -378606 microseconds. Ignoring time.
Nmap scan report for 192.168.126.132
Host is up, received arp-response (0.00087s latency).
Scanned at 2021-03-12 06:11:02 EST for 160s
Not shown: 65505 closed ports
Reason: 65505 resets
PORT      STATE SERVICE      REASON      VERSION
21/tcp    open  ftp          syn-ack ttl 64 vsftpd 2.3.4
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_ftp-syst:
|_STAT:
|_FTP server status:
|_Connected to 192.168.126.128
|_Logged in as ftp
|_TYPE: ASCII
|_No session bandwidth limit
|_Session timeout in seconds is 300
|_Control connection is plain text
|_Data connections will be plain text
|_vsFTPD 2.3.4 - secure, fast, stable
|_End of status
22/tcp    open  ssh          syn-ack ttl 64 OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2)
|_ssh-hostkey:
|_1024 60:0f:cf:e1:c0:5f:6a:74:d6:90:24:fa:c4:d5:6c:cd (DSA)
|_ssh-dss AAAAB3NzaC1kc3MAAACBALz4hsc8a2SrqnW960qV8xwBG0JC+jI7fWxm5METIJH4tKr/xUTw
```

## Enum4Linux Scan

It also runs the Enum4Linux scan upon detecting the operating system like Linux. The result for this scan is located at the following location: results/<targetname>/scans/enum4linux.txt.

```
cat ~/results/192.168.126.132/scans/enum4linux.txt
```



```
(root@kali)-[~/results/192.168.126.132]
# cat scans/enum4linux.txt
Starting enum4linux v0.8.9 ( http://labs.portcullis.co.uk/application/enum4linux)

=====
| Target Information |
=====
Target ..... 192.168.126.132
RID Range ..... 500-550,1000-1050
Username ..... ''
Password ..... ''
Known Usernames .. administrator, guest, krbtgt, domain admins, root, bin, none

=====
| Enumerating Workgroup/Domain on 192.168.126.132 |
=====
[+] Got domain/workgroup name: WORKGROUP

=====
| Nbtstat Information for 192.168.126.132 |
=====
Looking up status of 192.168.126.132
METASPLOITABLE <00> - B <ACTIVE> Workstation Service
METASPLOITABLE <03> - B <ACTIVE> Messenger Service
METASPLOITABLE <20> - B <ACTIVE> File Server Service
.. _MSBROWSE_. <01> - <GROUP> B <ACTIVE> Master Browser
WORKGROUP <00> - <GROUP> B <ACTIVE> Domain/Workgroup Name
WORKGROUP <1d> - B <ACTIVE> Master Browser
WORKGROUP <1e> - <GROUP> B <ACTIVE> Browser Service Elections

MAC Address = 00-00-00-00-00-00

=====
| Session Check on 192.168.126.132 |
=====
[E] Server doesn't allow session using username '', password ''. Aborting rema
```

## SMBMap Scan

Among other scans, AutoRecon also conducts SMBMap upon find the SMB service running on the application. SMBMap enumerates the different shares on the network by the target machine with the allowed permissions on that particular share. This scan result is located at the following location: results/<targetname>/scans/smbmap-share-permissions.txt.

```
cat ~/results/192.168.126.132/scans/smbmap-share-permissions.txt
```

```
(root@kali)-[~]
# cat results/192.168.126.133/scans/smbmap-share-permissions.txt
[+] IP: 192.168.126.133:139      Name: 192.168.126.133
    Disk
    -----
    print$      NO ACCESS
    tmp         READ, WRITE
    opt         NO ACCESS
    IPC$        NO ACCESS
    ADMIN$      NO ACCESS
[!] RPC Authentication error occurred
[!] Authentication error on 192.168.126.133
[+] IP: 192.168.126.133:445      Name: 192.168.126.133
    Disk
    -----
    print$      NO ACCESS
    tmp         READ, WRITE
    opt         NO ACCESS
    IPC$        NO ACCESS
    ADMIN$      NO ACCESS
```

## SMTP Scan

A simple enumeration for the SMTP users is also performed using the script called smtp-user-enum. It is performed in case the SMTP service is detected on the target machine. It enumerates for the users that created on the SMTP instance. In our demonstration, we found that there are 4 users that exist on the target server. The result for the enumeration scan can be found at the following location. results/<targetname>/scans/tcp\_25\_smtp\_user-enum.txt.

```
cat ~/results/192.168.126.132/scans/tcp_25_smtp_user-enum.txt
```



```
(root@kali)-[~]
# cat results/192.168.126.133/scans/tcp_25_smtp_user-enum.txt
Starting smtp-user-enum v1.2 ( http://pentestmonkey.net/tools/smtp-user-enum )

| Scan Information |
|-----|
Mode ..... VRFY
Worker Processes ..... 5
Usernames file ..... /usr/share/seclists/Usernames/top-usernames-shortlist.txt
Target count ..... 1
Username count ..... 17
Target TCP port ..... 25
Query timeout ..... 5 secs
Target domain .....

##### Scan started at Mon Mar 22 12:00:27 2021 #####
192.168.126.133: root exists
192.168.126.133: mysql exists
192.168.126.133: user exists
192.168.126.133: ftp exists
##### Scan completed at Mon Mar 22 12:00:27 2021 #####
4 results.

17 queries in 1 seconds (17.0 queries / sec)
```

## WhatWeb Scan

Another one of the scan results to look for is the WhatWeb enumeration scan. It uses the WhatWeb functionality to grab the banner on various services and then analyzing the versions and releases of the various web-based services and frameworks. The result for the WhatWeb scan can be located at the following location:  
results/<targetname>/scans/tcp\_8180\_http\_whatweb.txt

```
file:///root/results/192.168.126.132/scans/tcp_8180_http_whatweb.txt
```



```

file:///root/results/192.168.126.133/scans/tcp_8180_http_whatweb.txt

WhatWeb report for http://192.168.126.133:8180
Status      : 200 OK
Title       : Apache Tomcat/5.5
IP          : 192.168.126.133
Country     : RESERVED, ZZ

Summary     : Apache, HTTPServer[Apache-Coyote/1.1], Email[dev@tomcat.apache.org,users@tomcat.apache.org]

Detected Plugins:
[ Apache ]
  The Apache HTTP Server Project is an effort to develop and
  maintain an open-source HTTP server for modern operating
  systems including UNIX and Windows NT. The goal of this
  project is to provide a secure, efficient and extensible
  server that provides HTTP services in sync with the current
  HTTP standards.

  Google Dorks: (3)
  Website      : http://httpd.apache.org/

[ Email ]
  Extract email addresses. Find valid email address and
  syntactically invalid email addresses from mailto: link
  tags. We match syntactically invalid links containing
  mailto: to catch anti-spam email addresses, eg. bob at
  gmail.com. This uses the simplified email regular
  expression from
  http://www.regular-expressions.info/email.html for valid
  email address matching.

  String       : dev@tomcat.apache.org,users@tomcat.apache.org
  String       : dev@tomcat.apache.org,users@tomcat.apache.org

[ HTTPServer ]
  HTTP server header string. This plugin also attempts to
  identify the operating system from the server header.

  String       : Apache-Coyote/1.1 (from server string)

[ PoweredBy ]
  This plugin identifies instances of 'Powered by x' text and

```

## XML Results

AutoRecon also crafts a few results in the XML format for a clean and easy read. One of them results in our demonstration is the Nmap Scan for the FTP service. The XML result for the WhatWeb scan can be located at the following location:

```
file:///root/results/192.168.126.132/scans/xml/tcp_21_ftp_nmap.xml
```



```

cpe:/a:vsftpd:vsftpd:2.3.4 FTP server status: Connected to 192.168.126.128 Logged in as ftp TYPE: ASCII No
Control connection is plain text Data connections will be plain text vsFTPD 2.3.4 - secure, fast, stable End of
(Exploitable) BID:48539 CVE:CVE-2011-2523 vsFTPD version 2.3.4 backdoor, this was reported on 2011-07-0
uid=0(root) gid=0(root) https://www.securityfocus.com/bid/48539 https://github.com/rapid7/metasploit-frame
/vsftpd_234_backdoor.rb http://scarybeastsecurity.blogspot.com/2011/07/alert-vsftpd-download-backdoored.p
2011-2523

```

## Nikto Scan

All the scans that the AutoRecon additionally run are very useful in any Penetration Assessment but Nikto is one that might do the most enumeration as it after Nmap if any tool that can extract more data is Nikto. The scan result for the Nikto scan is located at this location:

```
cat ~/results/192.168.126.132/scans/tcp_8180_http_nikto.txt
```

```

(root@kali)~[~]
# cat results/192.168.126.133/scans/tcp_8180_http_nikto.txt
Nikto v2.1.6

+ Target IP: 192.168.126.133
+ Target Hostname: 192.168.126.133
+ Target Port: 8180
+ Start Time: 2021-03-22 12:01:22 (GMT-4)

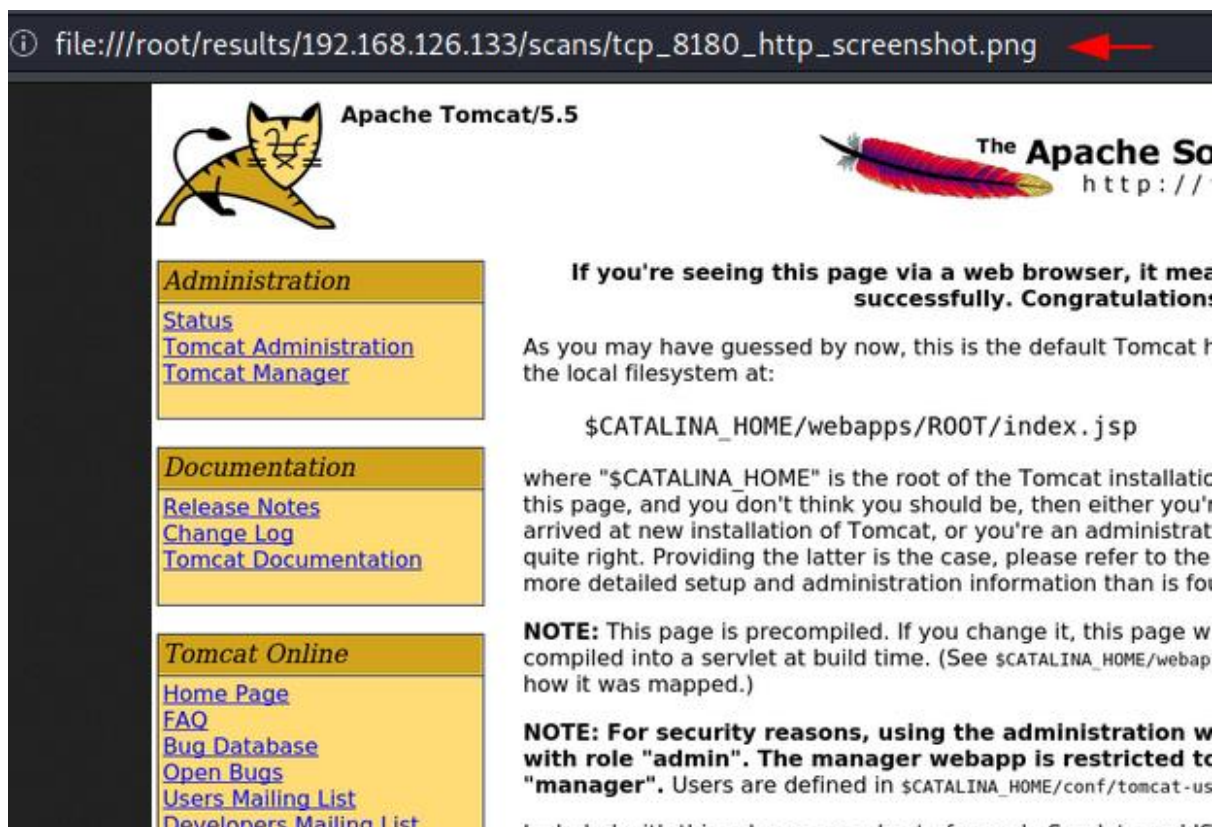
+ Server: Apache-Coyote/1.1
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the u
+ The X-Content-Type-Options header is not set. This could allow the user a
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ OSVDB-39272: /favicon.ico file identifies this app/server as: Apache Tomcat
+ Allowed HTTP Methods: GET, HEAD, POST, PUT, DELETE, TRACE, OPTIONS
+ OSVDB-397: HTTP method ('Allow' Header): 'PUT' method could allow clients
+ OSVDB-5646: HTTP method ('Allow' Header): 'DELETE' may allow clients to r
+ Web Server returns a valid response with junk HTTP methods, this may caus
+ /: Appears to be a default Apache Tomcat install.
+ Cookie JSESSIONID created without the httponly flag
+ OSVDB-376: /admin/contextAdmin/contextAdmin.html: Tomcat may be configure
+ OSVDB-3092: /admin/: This might be interesting...
+ OSVDB-3233: /tomcat-docs/index.html: Default Apache Tomcat documentation
+ OSVDB-3233: /manager/html-manager-howto.html: Tomcat documentation found.
+ OSVDB-3233: /manager/manager-howto.html: Tomcat documentation found.
+ OSVDB-3092: /webdav/index.html: WebDAV support is enabled.
+ OSVDB-3233: /jsp-examples/: Apache Java Server Pages documentation.
+ /admin/account.html: Admin login page/section found.
+ /admin/controlpanel.html: Admin login page/section found.

```

## Web Services Screenshots

When faced with an HTTP service that might contain webpages, AutoRecon snaps a screenshot of the webpage. In our demonstration, there was a HTTP service running on port 8180. It was the Apache Tomcat default page. This is super helpful when solving CTFs as we need to take a look at the web services. This way we can know if it is worth browsing web service or not.

`file:///root/results/192.168.126.132/scans/tcp_8180_http_screenshot.png`





## Conclusion

Hence, one can make use of these commands as a cybersecurity professional to assess vulnerabilities on systems and keep these systems away from threat.

## References

- <https://www.hackingarticles.in/comprehensive-guide-to-autorecon/>
- <https://github.com/Tib3rius/AutoRecon>