10th Gurugram Police Cyber Security Summer Internship 2022

Gurugram Police & CyberPeace Foundation

20TH JUNE 2022 TO 20TH JULY 2022

Commissionerate of
Police Shanti Nagar,
Shivaji Nagar,
Sector 11, Gurugram, Haryana 122018

Under the guidance of

Dr. Rakshit Tandon
GPCSSI 2022

TABLE OF CONTENTS

DECLARATION

We certify that the work contained in this report is original and has been done by us under the guidance of **Dr. Rakshit Tandon Sir.**

a. The work has not been submitted to any other sources.

b. We have followed the guidelines provided by this Internship.

Name and Signature of Project Team Members:

Sr. No.	**Name of Team Member**	**Signature**
1.	Aksh Puri *(B.Tech, Dr. Akhilesh Das Gupta Institute of Technology & Management)*	
2.	*Asif Mohammad Khan (BCA, Jamia Hamdard University)*	
3.	*MD Tajdar Alam Ansari (BSC, Bhairab Ganguly College)*	
4.	Yash Chavhan (B.tech CSE AI & ML)	

<u>CERTIFICATE</u>

I hereby affirm, to the best of my knowledge and belief, based on inspections, observations, testing of the project and upon reports submitted by others, that this **Vulnerability mapping tool (RoboCap)**is substantially complete and operable. The Project was completed in accordance with the department's issued guidelines.

Date :

(Dr. Rakshit Tandon)

ACKNOWLEDGEMENT

The success and end result of this project requires a lot of guidance and endorsement from many people and we are fortunate to get all of these throughout our entire internship project.

We were able to accomplish this project only with such assistance and supervision and therefore, we will never forget to thank them.

We respect and thank **Dr. Rakshit Tandon , Commissioner of Police (Gurugram) , DCP Headquarters , Cyber peace Foundation , Gurugram Police** who allowed us to work on this specific project in **10th Gurugram Police Cyber Security Summer Internship 2022** and gave us all the support and guidance that motivated us to complete the project properly.

Despite being busy dealing with corporate matters we are very grateful to him for such admonition.

In addition, we would like to express our heartfelt gratitude to all the **Commissionerate police staff** for their timely support.

Yours Sincerely,

Aksh
Asif
Tajdar
Yash

Abstract

RoboCap is a tool used for Web/Network Recon and Vulnerability mapping. Basically designed for playing CTFs, can be used for more uses, such as, Vulnerability, scanning, Aggressive Scan based on nmap,Ports Enumeration, HTTP Methods, Exploits, SMB Enumeration, Web Technology's, SSL Testing, Fuzzing, HTTP Methods with Nikto and much more.

It is a great tool for pentesting environments to make things automated and save time in listing vulnerabilities.

This tool is also set to run user defined modules and run them such as OWASP vulnerabilities, fingerprinting, DNS, Samba, and even AD.

Methodology

IT first runs 2 nmap scans in tandem, one scan looks specifically for service versions to run against searchsploit and the other is a scan dependent on the argument. Every scan profile checks for services running, the type of scan is the only difference. After the scans are finished, the services/ports open and operating systems along with script output (if avaliable) is extracted and further analyzed.

If a certain service is found, RoboCap will begin enumerating by firing off a number of tools and create a dir for that service (i.e detecting http starts up nikto, wafw00f, gobuster, and others). If a dependency required is not detected, that dependency will be auto installed and checked if there is a new update everytime the tool is run. RoboCap outputs this information in 2 main sections(scan type and loot dirs) with sub directories branching off depending on what is found.

Source Code

```
#!/bin/bash
dir=$(dirname $(readlink -f $0))

if [ ! -x "$(command -v nmap)" ];then
        echo "[+] nmap not detected...Installing"
        sudo apt-get install nmap -y > installing;rm installing
fi

if [ ! -x "$(command -v nikto)" ];then
        echo "[+] nikto not detected. Installing..."
        sudo apt-get install nikto -y > installing;rm installing
fi

if [ ! -x "$(command -v gobuster)" ];then
        echo "[+] gobuster not detected. Installing..."
        sudo apt-get install gobuster -y > installing;rm installing
fi

if [ ! -x "$(command -v whatweb)" ];then
        echo "[+] whatweb not detected. installing..."
        sudo apt-get install whatweb -y > installing;rm installing
fi

if [ ! -x "$(command -v onesixtyone)" ];then
        echo "[+] onesixtyone not detected. Installing..."
        sudo apt-get install onesixtyone -y > installing;rm installing
fi

if [ ! -x "$(command -v rpcbind)" ];then
        echo "rpcbind not detected. Installing..."
        sudo apt-get install rpcbind -y > installing;rm installing
fi

if [ ! -x "$(command -v snmp-check)" ];then
        echo "[+] snmp-check not detected. Installing..."
        sudo apt-get install snmp-check -y > installing;rm installing
fi

if [ ! -x "$(command -v snmpwalk)" ];then
        echo "[+] snmpwalk not detected. Installing..."
        sudo apt-get install snmpwalk -y > installing;rm installing
fi

if [ ! -x "$(command -v fierce)" ];then
        echo "[+] fierce not detected. Installing..."
        sudo apt-get install fierce -y > installing;rm installing
fi

if [ ! -x "$(command -v dnsrecon)" ];then
        echo "[+] dnsrecon not detected. Installing..."
        sudo apt-get installl dnsrecon -y > installing;rm installing
fi

if [ ! -x "$(command -v dnsenum)" ];then
        echo "[+] dnsenum not detected. Installing..."
        sudo apt-get install dnsenum -y > installing;rm installing
fi

if [ ! -x "$(command -v oscanner)" ];then
        echo "[+] oscanner not detected. Installing..."
        sudo apt-get install oscanner -y > installing;rm installing
```

```
fi

if [ ! -x "$(command -v wafw00f)" ];then
		echo "[+] wafw00f not detected. Installing..."
		sudo apt-get install wafw00f -y > installing;rm installing
fi

if [ ! -x "$(command -v odat)" ];then
		echo "[+] odat not detected. installing..."
		sudo apt-get install odat -y > installing;rm installing
fi

if [ ! -x "$(command -v jq)" ];then
		echo "[+] jq not detected. installing..."
		sudo apt-get install jq -y > installing;rm installing
fi

if [ ! -x "$(command -v tput)" ];then
		echo "[+] tput not detected. installing..."
		sudo apt-get install tput -y > installing;rm installing
fi

source /home/mactavish/Documents/autoenum/functions/banner.sh
source /home/mactavish/Documents/autoenum/functions/upgrade.sh
source /home/mactavish/Documents/autoenum/functions/scans.sh
source /home/mactavish/Documents/autoenum/functions/enum.sh
source /home/mactavish/Documents/autoenum/functions/help_general.sh
source /home/mactavish/Documents/autoenum/functions/menu.sh


if [[ $1 == '-nr' ]];then nr=1;fi
clear
banner
if [ $nr ];then tput setaf 2;echo -en "\n[*] autoenum set to noresolve mode";tput sgr0;sleep 0.5;fi
get_ip
halp_meh
menu
```

```
#!/bin/bash

redis_enum (){
		mkdir $loot/redis
    tput setaf 2;echo "[+] Starting redis enum";tput sgr0
		nmap --script redis-info -sV -p 6379 $IP | tee -a $loot/redis/redis_info
		echo "msf> use auxiliary/scanner/redis/redis_server" >> $loot/redis/manual_cmds
}

snmp_enum (){
		mkdir $loot/snmp
    tput setaf 2;echo "[+] Starting snmp enum";tput sgr0
		onesixtyone -c /usr/share/doc/onesixtyone/dict.txt $IP | tee -a $loot/snmp/snmpenum
#		create algo to check which version of snmp is runnign or pull it off a banner grab
		snmp-check -c public -v 1 -d $IP | tee -a $loot/snmp/snmpcheck
		if grep -q "SNMP request timeout" "$loot/snmp/snmpcheck";then
		rm $loot/snmp/snmpcheck
		snmpwalk -c public -v2c $IP | tee -a $loot/snmp/uderstuff
		echo "snmpwalk -c public -v2c $IP" >> $loot/snmp/cmds_run &
		if grep -q "timeout" "$loot/snmp/uderstuff";then rm $loot/snmp/uderstuff;else mv $loot/snmp/uderstuff $loot/snmp/snmpenum;fi
		else
		mv $loot/snmp/snmpcheck $loot/snmp/snmpenum
		fi
		echo "onesixtyone -c /usr/share/doc/onesixtyone/dict.txt $IP" >> $loot/snmp/cmds_run &
		echo "snmp-check -c public $IP" >> $loot/snmp/cmds_run &
		wait
		rm $IP/autoenum/loot/raw/snmp_found
}

rpc_enum (){
		mkdir $loot/rpc
   tput setaf 2;echo "[+] Starting rpc enum";tput sgr0
		port=$(cat $loot/raw/rpc_found | grep "rpc" | awk '{print($1)}' | cut -d '/' -f 1)
		nmap -sV -p $port --script=rpcinfo >> $loot/rpc/ports
		if grep -q "" "$loot/rpc/ports";then rm $loot/rpc/ports;fi
		rpcbind -p $IP | tee -a $loot/rpc/versions
		if grep -q "nfs" "$loot/rpc/ports";then nfs_enum;fi
		rm $loot/raw/rpc_found
}

nfs_enum (){
		mkdir $loot/nfs
   tput setaf 2;echo "[+] Starting nfs enum";tput sgr0
		nmap -p 111 --script nfs* $IP | tee $loot/nfs/scripts
		# add chunk to automount if share is found
		share=$(cat $loot/nfs/scripts | grep "|_ " -m 1 | awk '{print($2)}')
		if grep -q "mfs-showmount" "$loot/nfs/scripts";then
		mkdir $loots/nfs/mount
		# pull share location and assign it to share var
		mount -o nolock $IP:$share $loot/nfs/mount
		fi
}

pop3_enum (){
		mkdir $loot/pop3
   tput setaf 2;echo "[+] Starting pop3 enum";tput sgr0
		nmap -sV --script pop3-brute $IP | tee -a $loot/pop3/brute
		echo "telnet $IP 110" >> $loot/pop3/manual_cmds
		rm $loot/raw/pop3_found
}

imap_enum (){
		echo "[+] Work in progress"
}

ldap_enum (){
		mkdir $loot/ldap
   tput setaf 2;echo "[+] Starting ldap enum";tput sgr0
		nmap -vv -Pn -sV -p 389 --script='(ldap* or ssl*) and not (brute or broadcast or dos or external or fuzzer)' $IP | tee -a
$loot/ldap/ldap_scripts
		#ldapsearch -x -h $rhost -s base namingcontexts | tee -a $loot/ldap/ldapsearch &
		echo "nmap -vv -Pn -sV -p 389 --script='(ldap* or ssl*) and not (brute or broadcast or dos or external or fuzzer)' $IP" >>
$loot/ldap/cmds_run &
		wait
		rm $loot/raw/ldap_found
}

dns_enum (){
		mkdir $loot/dns
		# mainly for pentesting use, not neccesary rn for oscp. retest later when adding to this
		#host $IP >> $loot/dns/host_out
		#host -t mx $IP >> $loot/dns/host_out
		#host -t txt $IP >> $loot/dns/host_out
		#host -t ns $IP >> $loot/dns/host_out
		#host -t ptr $IP >> $loot/dns/host_out
		#host -t cname $IP >> $loot/dns/host_out
		#host -t a $IP >> $loot/dns/host_out
		#for host in <list of subs>;do host -l <host> <dns server addr>;done
		#fierce -dns $IP
		#dnsenum --enum $IP
		#dnsrecon -d $IP
		#gobuster -dns $IP

		echo " "
}

ftp_enum (){
		mkdir -p $loot/ftp
		echo "[+] Starting FTP enum..."
		cat $loot/raw/ftp_found | awk '{print($1)}' | cut -d '/' -f 1 > $loot/ftp/port_list
		for port in $(cat $loot/ftp/port_list);do
		nmap -sV -Pn -p $port --script=ftp-anon,ftp-bounce,ftp-libopie,ftp-proftpd-backdoor,ftp-vsftpd-backdoor,ftp-vuln-cve2010-4221,ftp-syst
-v $IP | tee -a $loot/ftp/ftp_scripts
		done
		echo "nmap -sV -Pn -p $port
--script=ftp-anon,ftp-bounce,ftp-libopie,ftp-proftpd-backdoor,ftp-vsftpd-backdoor,ftp-vuln-cve2010-4221,ftp-syst -v $IP " >> $loot/ftp/cmds_run &
		wait
		rm $loot/ftp/port_list
		rm $loot/raw/ftp_found
		echo "[+] FTP enum complete"
}

smtp_enum (){
		mkdir $loot/smtp
   echo "[+] Starting SNMP enum..."
		cat $loot/raw/snmp_found | awk '{print($1)}' | cut -d '/' -f 1 > $loot/smtp/port_list
		for port in $(cat $loot/smtp/port_list);do
		smtp-user-enum -M VRFY -U /usr/share/metasploit-framework/data/wordlists/unix_users.txt -t $IP -p $port | tee -a $loot/smtp/users
		done
		if grep -q "0 results" "$loot/smtp/users";then rm $loot/smtp/users;fi
		echo "nc -nvv $IP $port" >> $loot/smtp/maunal_cmds
		echo "telnet $IP $port" >> $loot/smpt/manual_cmds
		echo "smtp-user-enum -M VRFY -U /usr/share/metasploit-framework/data/wordlists/unix_users.txt -t $IP -p $port" >>
$loot/smtp/cmds_run &
		wait
		rm $loot/smtp/port_list
		rm $loot/raw/smtp_found
}

oracle_enum (){
		mkdir $loot/oracle
   echo "[+] Starting Oracle enum..."
		#swap out port with port(s) found running oracle
		nmap -sV -p 1521 --script oracle-enum-users.nse,oracle-sid-brute.nse,oracle-tns-version.nse | tee -a $loot/oracle/nmapstuff
		oscanner -v -s $IP -P 1521 | tee -a $loot/oracle/
		echo "[+] Running ODAT..."
		odat tnscmd -s $rhost --version --status --ping 2>/dev/null | tee -a $loot/oracle/odat_tnscmd
		odat sidguesser -s $rhost 2>/dev/null | tee -a $loot/oracle/odat_enum
		rm $loot/raw/oracle_found
}

http_enum (){
		mkdir -p $IP/autoenum/loot/http
		echo "[+] http enum starting..."
   pct=$(cat $loot/raw/http_found | wc -l)
   if [[ $pct -gt 1 ]];then
		 echo "[+] Multiple HTTP ports detected"
		for port in $(cat $loot/raw/http_found);do
				 mkdir $loot/http/$port
				echo "[+] Firing up nikto on port $port"
				nikto -ask=no -h $IP:$port -T 123b | tee -a  $loot/http/$port/nitko
				echo "[+] checking ssl for possible holes on port $port"
				 sslscan --show-certificate $IP:$port | tee -a $loot/http/$port/sslinfo &
				 echo "[+] Curling interesting files on port $port"
```

```
                    curl -sSiK $IP:$port/index.html | tee -a $loot/http/$port/landingpage &
                    curl -sSik $IP:$port/robots.txt | tee -a $loot/http/$port/robots.txt &
                    echo -e "\n[+] Pulling headers/plugin info with whatweb on port $port"
                    whatweb -a3 $IP:$port 2>/dev/null | tee -a $loot/http/$port/whatweb &
                    wait
                   echo "[+] bruteforcing dirs on $IP:$port"
                   gobuster dir -re -t 65 -u http://$IP:$port -w /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt -o
$loot/http/$port/dirs_found -k
#                   if IIS detected
#                   echo "[*] IIS detected"
#                  echo "[+] enumerating dav..."
#                  mkdir -p $loot/dav
#                  davtest -url http://$IP:$port | tee -a $loot/dav/dav_enum_$port
#                   if wordpress detected
#                   echo -e "[*] WordPress detected\nRunning wpscan"
#                   run wpscan | tee -a $loot/http/wpscan_$port
          done
          elif [[ $pct == 1 ]];then
           port=$(cat $loot/raw/http_found)
          echo "[+] firing up nikto"
          nikto -ask=no -h $IP:$port >> $loot/http/nikto_out &
           #echo "[+] Running unican in background"
          #uniscan -u http://$IP -bqweds >> $loot/http/uniscan
          echo "[+] checking ssl for possible holes"
          sslscan --show-certificate $IP:$port | tee -a $loot/http/sslinfo
           echo "[+] Pulling headers/plugin info with whatweb"
           whatweb -a3 $IP:$port 2>/dev/null | tee -a $loot/http/whatweb
          echo "[+] Curling interesting files"
          curl -sSiK $IP:$port/index.html | tee -a $loot/http/landingpage &
          curl -sSik $IP:$port/robots.txt | tee -a $loot/http/robots.txt &
           wait
          echo "[+] bruteforcing dirs on $IP"
          gobuster dir -re -t 65 -u $IP:$port -w /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt -o $loot/http/dirs_found -k
#          if IIS detected
#         echo "[+] enumerating dav..."
#         davtest -url http://$IP | tee -a $loot/http/dav_enum
#                  if wordpress detected
#                  echo -e "[*] WordPress detected\nRunning wpscan"
#                  run wpscan | tee -a $loot/http/wpscan_$port

          fi
          touch $loot/http/cmds_run
          echo "uniscan -u http://$IP -qweds" >> $loot/http/cmds_run &
          echo "sslscan --show-certificate $IP:80 " >> $loot/http/cmds_run &
          echo "nikto -h $IP" >> $loot/http/cmds_run &
          echo "gobuster dir -re -t 45 -u $IP -w /usr/share/wordlists/dirb/common.txt" >> $loot/http/cmds_run &
          echo "curl -sSiK $IP" >> $loot/http/cmds_run &
          echo "curl -sSiK $IP/robots.txt" >> $loot/http/cmds_run &
          echo "whatweb -v -a 3 $IP" >> $loot/http/cmds_run &
#         echo "wafw00f http://$IP" >> $loot/http/cmds_run &
          wait
          echo "[+] http enum complete!"
}

smb_enum (){
          echo "[+] Starting SMB enum..."
          mkdir -p $loot/smb
          mkdir -p $loot/smb/shares
          # checks for eternal blue and other common smb vulns
          nmap --script smb-vuln-ms17-010.nse --script-args=unsafe=1 -p 139,445 $IP | tee -a $loot/smb/eternalblue
          if ! grep -q "smb-vuln-ms17-010:" "auotenum/loot/smb/eternalblue"; then rm $loot/smb/eternalblue;fi
          nmap --script smb-vuln-ms08-067.nse --script-args=unsafe=1 -p 445 $IP | tee -a $loot/smb/08-067
          if ! grep -q "smb-vuln-ms08-067:" "autoenum/loot/smb/08-067";then rm $loot/smb/08-067;fi
          nmap --script smb-vuln* -p 139,445 $IP | tee -a $loot/smb/gen_vulns
          #shares n' stuff
          nmap --script smb-enum-shares -p 139,445 $IP | tee -a $loot/smb/shares/nmap_shares
          smbmap -H $IP -R | tee -a $loot/smb/shares/smbmap_out
          smbclient -N -L \\\\$IP | tee -a $loot/smb/shares/smbclient_out
          if grep -q "Not enough '\' characters in service" "$loot/smb/shares/smbclient_out";then smbclient -N -H \\\\\\$IP | tee -a
```

```
$loot/smb/shares/smbclient_out;fi
        if grep -q "Not enough '\' characters in service" "$loot/smb/shares/smbclient_out";then smbclient -N -H \\$IP | tee -a
$loot/smb/shares/smbclient_out;fi
        if grep -q "Not enough '\' characters in service" "$loot/smb/shares/smbclient_out";then rm $loot/smb/shares/smbclient_out; echo
"smbclient could not be auotmatically run, rerun smbclient -N -H [IP] manauly" >> $loot/smb/notes;fi
        if grep -q "Error NT_STATUS_UNSUCCESSFUL" "$loot/smb/shares/smbclient_out";then rm $loot/smb/shares/smbclient;fi
        if [[ -s "$loot/smb/shares/smbclient_out" ]];then echo "smb shares open to null login, use rpcclient -U '' -N [ip] to run rpc commands,
use smbmap -u null -p '' -H $IP -R to verify this" >> $loot/smb/notes;fi
        find ~ -path '*/$IP/autoenum/loot/smb/*' -type f > $loot/smb/files
        for file in $(cat $loot/smb/files);do
        if grep -q "QUITTING!" "$file" || grep -q "ERROR: Script execution failed" "$file" || grep "segmentation fault" "$file";then rm $file;fi
        done
        touch $loot/smb/cmds_run
        echo "nmap --script smb-vuln-ms17-010.nse --script-args=unsafe=1 -p 139,445 $IP " >> $loot/smb/cmds_run &
        echo "nmap --script smb-vuln-ms08-067.nse --script-args=unsafe=1 -p 445 $IP" >> $loot/smb/cmds_run &
        echo "nmap --script smb-vuln* -p 139,445 $IP" >> $loot/smb/cmds_run &
        echo "nmap --script smb-enum-shares -p 139,445 $IP" >> $loot/smb/cmds_run &
        echo "smbmap -H $IP -R " >> $loot/smb/cmds_run &
        echo "smbclient -N -L \\\\$IP " >> $loot/smb/cmds_run &
        wait
        rm $loot/smb/files
        rm $loot/raw/smb_found
        echo "[+] SMB enum complete!"
}

linux_enum (){
        #get exact snmp version
        echo "[-] Work in Progress"
}

windows_enum (){
        # get exact snmp version
        # pull entire MIB into sections
        echo "[-] Work in Progress"
}
```

```
#!/bin/bash

cleanup (){
        echo "[+] Cleaning up..."
        find $IP/autoenum/ -type d -empty -delete
        find $IP/autoenum/ -type f -empty -delete
        if [[ -f "installed" ]];then rm installed;fi
}

get_ip (){
        echo -e
        echo "Enter a target IP or hostname "
        tput bold;tput setaf 1; echo -en "Autoenum > ";tput sgr0;read unchecked_IP
        if [ $nr ];then
        if [[ $unchecked_IP =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]];then
                IP="$unchecked_IP";sleep 1
                tput setaf 4;echo -e "[+] IP set to $IP";tput sgr0;echo -e
        fi
        else
        if [[ $unchecked_IP =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]];then
```

```
                    IP="$unchecked_IP";sleep 1
                    cwd=$(pwd);ping -c 1 -W 3 $IP | head -n2 | tail -n1 > $cwd/tmp
                    if ! grep -q "64 bytes" "tmp";then
                    echo -e "[-] IP failed to resolve\n[-] Exiting..."
                    exit
                    fi
                    rm $cwd/tmp
                    tput setaf 4;echo -e "[+] IP set to $IP";tput sgr0;echo -e
        elif [[ $unchecked_IP =~ [a-z,A-Z,0-9].[a-z]$ ]] || [[ $unchecked_IP =~ [a-z].[a-z,A-Z,0-9].[a-z]$ ]];then
                    IP=$(host $unchecked_IP | head -n1 | awk '{print($4)}')
                    tput setaf 4;echo -e "$unchecked_IP resolved to $IP\n";tput sgr0
        else
                    tput setaf 8
                    echo "[-] Invalid IP or hostname detected."
                    echo -e "[-] Example:\n\t[>] 192.168.1.5\n\t[>] google.com"
                    tput sgr0
                    get_ip
        fi
        fi
}

shell_preserve (){
        echo "[+] You have entered shell mode. use done to exit"
        while true ;do
        echo -en "[+] Command > ";read cmd
        if [[ "$cmd" =~ "done" ]];then
                    $cmd  2>/dev/null;echo -e
                    break
         elif [[ "$cmd" =~ "exit" ]];then
                     echo -en "[-] Exit shell mode? [y/n] > ";read opt
                     if [[ "$opt" == "y" ]];then
                              echo -e "[-] Exiting shell mode\n"
                              break
                     fi
        else
                    $cmd 2>/dev/null
        fi
        done
}

halp_meh (){
        tput smul;echo "General Commands:";tput rmul
        echo -e "[*] ping"
        echo -e "[*] help"
        echo -e "[*] banner"
        echo -e "[*] clear"
#       echo -e "[*] home"
        echo -e "[*] reset"
        echo -e "[*] commands"
        echo -e "[*] shell"
        echo -e "[*] upgrade"
        echo -e "[*] set target"
#       echo -e "[*] use [tool]"
        echo -e "[*] exit"
        echo -e
        tput smul;echo "Scan Profiles:";tput rmul
        tput bold;echo -e "[~] Main:";tput sgr0
        echo -e "[*] aggr"
        echo -e "[*] reg"
   echo -e "[*] top 1k"
   echo -e "[*] top 10k"
        echo -e "[*] aggr+vuln"
        echo -e "[*] reg+vuln"
   echo -e "[*] top 1k+vuln"
   echo -e "[*] top 10k+vuln"
   echo -e "[*] udp"
        echo -e
        tput bold;echo -e "[~] Auxiliary:";tput sgr0
        echo -e "[*] vuln"
```

```
		echo -e "[*] quick"
#		tput smul;echo "Standalone Utils:";tput rmul
#		echo -e "[*] amass"
#		echo -e
#		tput smul;echo "Module Commands:";tput rmul
#		echo -e "[*] list modules"
#		echo -e "[*] set module";
    echo -e;sleep 0.5
}

halp_meh_pws (){
		tput smul;echo "General Commands:";tput rmul
		echo "[*] ping - Verify host is up/accepting ping probes"
		echo "[*] help - displays this page"
		echo "[*] banner - display banner"
		echo "[*] clear - clears screen"
#		echo "[*] home - returns to home module"
		echo "[*] reset - run this if text is unviewable after a scan"
		echo "[*] commands - shows all avaliable commands"
		echo "[*] shell - allows you to run commands as if in a terminal"
		echo "[*] upgrade - checks to see if any dependencies require an update"
		echo "[*] set target - opens prompt to change target IP"
#		echo "[*] use [tool] - invokes use of a standalone tool"
		echo -e
		tput smul;echo "Scan Profiles:";tput rmul
		tput bold;echo "[~] Main - These scans are 'the works', enumerate further depending on services discovered ";tput sgr0
		echo "[*] aggr - scans all ports aggressively"
		echo "[*] reg - scans all ports normally, no scripts and checks only for OS"
    echo "[*] top 1k - run a number of scans on the first 1000 ports"
    echo "[*] top 10k - runs a number of scans on the first 10000 ports"
		echo "[*] aggr+vuln - aggr scan. Also fires off NSE on discovered services searching for known exploits"
		echo "[*] reg+vuln - reg scan. Also firing off NSE on discovered services searching for known exploits"
    echo "[*] top 1k+vuln - runs the top 1k scans and vuln scan"
    echo "[*] top 10k+vuln - runs the top 10k scans and vuln scan"
    echo "[*] udp - checks for udp ports"
		echo -e
		tput bold;echo "[~] Auxiliary - These scans can be run standalone, do not enumerate beyond";tput sgr0
		echo "[*] quick - scans with scripts enabled for quick script enumeration"
		echo "[*] vuln - searches for services and checks for known exploits"
		echo -e;sleep 0.5
#		tput smul;echo "Standalone Tools:";tput rmul
#		echo "[*] amass - invokes the OWASP amass tool, highly configurable"
#		echo -e
#		tput smul;echo "Module Commands:";tput rmul
#		echo "[*] list modules - prints list of availiable modules"
#		echo "[*] set module - opens prompt to move into or change modules
}
```

```
#!/bin/bash

#source functions/menu/web.sh
```

```
#source functions/menu/smb.sh
#source functions/menu/dns.sh
#source functions/menu/fingerprint.sh
#source functions/menu/validate.sh
#source functions/menu/amass.sh

menu (){

WHITE='\033[01;37m'
CLEAR='\033[0m'
# https://medium.com/bugbountywriteup/fasten-your-recon-process-using-shell-scripting-359800905d2a

if [[  "$module" == "" ]];then
		cli="Autoenum($IP) > "
fi

tput bold;tput setaf 1;echo -en "$cli";tput sgr0;read arg
while true && [[ ! "$IP" == " " ]];do
		# add more color
		# add more banners (?)...grimmie want more banners :(

		mkbasedirs (){
		echo "[+] Checking for base dirs..."
		if [[ ! -d "$IP/autoenum" ]];then mkdir -p $IP/autoenum;fi
		if [[ ! -d "$IP/autoenum/loot/raw" ]];then mkdir -p $IP/autoenum/loot/raw; loot="$IP/autoenum/loot";else loot="$IP/autoenum/loot";fi
		if [[ ! -d "$loot/exploits" ]];then mkdir -p $loot/exploits;fi
		echo "[+] Done!"
		}
		case $arg in
		"")
				menu
				break
				;;
		"home")
				cli="Autoenum($IP) > "
				menu
				break
				;;
		"commands")
				halp_meh
				menu
				break
				;;
		"shell")
				shell_preserve
				menu
				break
				;;
		"reset")
				reset
				menu
				break
				;;
		"upgrade")
				upgrade
				menu
				break
				;;
		"clear")
				clear
				menu
				break
				;;
		"banner")
				banner
				menu
				break
				;;
		"ping")
```

```
                if [[ "$IP" == "dev" ]];then
                        echo "[-] set an IP. use set target to do this"
                else
                        ping $IP -c 1;echo -e
                fi
                menu
               break
               ;;
 "udp")
                echo "[~] SCAN MODE: udp";sleep 2;echo -e
                mkbasedirs
                udp
                menu
                break
                ;;
"vuln")
               echo "[~] SCAN MODE: vuln";sleep 2;echo -e
               mkbasedirs
               vuln
               menu
               break
               ;;
"aggr")
               echo "[~] SCAN MODE: aggr";sleep 2;echo -e
               mkbasedirs
               aggr
               cleanup
               menu
               break
               ;;
"reg")
               echo "[~] SCAN MODE: reg";sleep 2;echo -e
               mkbasedirs
               reg
               cleanup
               menu
               break
               ;;
"quick")
               echo "[~] SCAN MODE: quick";sleep 2;echo -e
               nmap -sC -sV -T4 -Pn $IP
               menu
               break
               ;;
 "top 1k" | "top1k")
                echo "[~] SCAN MODE: top 1k";sleep 2;echo -e
                mkbasedirs
                top_1k
                cleanup
                menu
                break
                ;;
 "top 10k" | "top10k")
                echo "[~] SCAN MODE: top 10k";sleep 2;echo -e
                mkbasedirs
                top_10k
                cleanup
                menu
                break
                ;;
 "top 1k+vuln" | "top1k+vuln")
                echo "[~] SCAN MODE: top 1k+vuln";sleep 2;echo -e
                mkbasedirs
                top_1k
                vuln
                cleanup
                menu
                break
                ;;
```

```
		"top 10k+vuln" | "top10k+vuln")
			echo "[~] SCAN MODE: top 10k+vuln";sleep 2;echo -e
			mkbasedirs
			top_10k
			vuln
			cleanup
			menu
			break
			;;
		"aggr+vuln")
			echo "[~] SCAN MODE: aggr+vuln";sleep 2;echo -e
			mkbasedirs
			aggr
			vuln
			cleanup
			menu
			break
			;;
		"reg+vuln")
			echo "[~] SCAN MODE: reg+vuln";sleep 2;echo -e
			mkbasedirs
			reg
			vuln
			cleanup
			menu
			break
			;;
		"help")
			halp_meh_pws
			menu
			break
			;;
		"set target")
			echo -en "Enter IP/hostname > ";read unchecked_IP
			if [[ $unchecked_IP =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]];then
				cwd=$(pwd);ping -c 1 $unchecked_IP | head -n2 | tail -n1 > $cwd/tmp
				if ! grep -q "64 bytes" "tmp";then
					echo "[-] IP failed to resolve"
				else
					IP="$unchecked_IP";tput setaf 4;echo -e "[+] IP set to $IP";tput sgr0;echo -e
				fi
				rm $cwd/tmp
			elif [[ $unchecked_IP =~ [a-z,A-Z,0-9].[a-z]$ ]] || [[ $unchecked_IP =~ [a-z].[a-z,A-Z,0-9].[a-z]$ ]];then
				IP=$(host $unchecked_IP | head -n1 | awk '{print($4)}')
				tput setaf 4;echo -e "$unchecked_IP resolved to $IP\n";tput sgr0
			elif [[ $unchecked_IP == "*" ]];then
				IP="dev"
			 else
			echo "[-] Invalid IP detected."
			echo "[-] Example: 192.168.1.5"
			fi
			echo "[*] IP changed to $IP"
			menu
			break
			;;
#		"use amass")
#			echo "[*] OWASP amass set to use"
#			OWASP_amass
#			break
#			;;
#		"list modules")
			# while base autoenum runs nmap an analysis based on services discovered, this module tatgets and deeply analyses target services while base autoenum glosses over services found
#			 echo "[*] Validate"
#			 echo "[*] Fingerprinting"
#			 echo "[*] Web"
#			 echo "[*] Samba"
#			 echo "[*] DNS"
#			 echo "[*] AD"
```

```
#                       menu
#                       break
#                       ;;
#           "set module")
#                       echo -en "module > ";read module
#                        if [[ "$module" == "Validate" ]];then
#                                   module="Validate";cli="Autoenum($IP)$WHITE [$module]$CLEAR > "
#                                   mkbasedirs
#                                   mkdir -p $loot/Modules/$module
#                                   validate_dir="$loot/Modules/$module"
#                                   echo "[+] Entering module: $module";sleep 1.5
#                                   validate
#                        elif [[ "$module" == "Fingerprinting" ]];then
#                                   module="Fingerprinting";cli="Autoenum($IP)$WHITE [$module]$CLEAR > "
#                       mkbasedirs
#                       mkdir -p $loot/Modules/$module
#                       fprint_dir="$loot/Modules/$module"
#                       echo "[+] Entering module: $module";sleep 1.5
#                                   fingerprint
#                       elif [[ "$module" == "Web" ]];then
#                                   module="Web";cli="Autoenum($IP)$WHITE [$module]$CLEAR > "
#                       mkbasedirs
#                       mkdir -p $loot/Modules/$module
#                       web_dir="$loot/Modules/$module"
#                       echo "[+] Entering module: $module";sleep 1.5
#                       Web
#                       elif [[ "$module" == "DNS" ]];then
#                                   module="DNS";cli="Autoenum($IP)$WHITE [$module]$CLEAR > "
#                       mkbasedirs
#                       mkdir -p $loot/Modules/$module
#                       DNS_dir="$loot/Modules/$module"
#                       echo "[+] Entering module: $module";sleep 1.5
#                       DNS
#                       elif [[ "$module" == "AD" ]];then
#                                   module="AD";cli="Autoenum($IP)$WHITE [$module]$CLEAR > "
#                       mkbasedirs
#                       mkdir -p $loot/Modules/$module
#                       AD_dir="$loot/Modules/$module"
#                       echo "[+] Entering module: $module";sleep 1.5
#                       AD
#                       elif [[ "$module" == "Samba" ]];then
#                                   module="Samba";cli="Autoenum($IP)$WHITE [$module]$CLEAR > "
#                       mkbasedirs
#                       mkdir -p $loot/Modules/$module
#                       samba_dir="$loot/Modules/$module"
#                       echo "[+] Entering module: $module";sleep 1.5
#                       Samba
#                       else
#                       echo "[-] Invalid module selected"
#                       fi
#                       menu
#                       break
#                       ;;
            "exit")
                        tput setaf 8;echo "[-] Terminating session..."
                        tput sgr0
                        sleep 1.5
                         exit 1
                        ;;
            *)
                        tput setaf 8;echo "[-] Invalid input detected"
                        tput sgr0
                        menu
                        break
                        ;;
            esac
done
}
```

```
#!/bin/bash

OS_guess (){
   guess=$(ping -c 1 -W 3 $IP | grep '64' | awk '{print($6)}' | cut -d '=' -f2)
   if [[ "$guess" == 127 ]] || [[ "$guess" == 128 ]];then
             tput setaf 2;echo "[*] This machine is probably running Windows";tput sgr0
   elif [[ "$guess" == 255 ]] || [[ "$guess" == 254 ]];then
             tput setaf 2;echo "[*] This machine is probably running Cisco/Solaris/OpenBSD";tput sgr0
   elif [[ "$guess" == 63 ]] || [[ "$guess" == 64 ]];then
             tput setaf 2;echo "[*] This machine is probably running Linux";tput sgr0
   else
             echo "[-] Could not determine OS"
   fi
   sleep 1.5
}

enum_goto (){
            if [[ -s "$loot/raw/redis_found" ]];then redis_enum;fi
            if [[ -s "$loot/raw/snmp_found" ]];then snmp_enum;fi
#           if [[ -s "$loot/raw/rpc_found" ]];then rpc_enum;fi
            if [[ -s "$loot/raw/pop3_found" ]];then pop3_enum;fi
            if [[ -s "$loot/raw/imap_found" ]];then imap_enum;fi
#           if [[ -s "$loot/raw/dns_found" ]];then dns_enum;fi
            if [[ -s "$loot/raw/ftp_found" ]];then ftp_enum;fi
            if [[ -s "$loot/raw/ldap_found" ]];then ldap_enum;fi
            if [[ -s "$loot/raw/smtp_found" ]];then smtp_enum;fi
            if [[ -s "$loot/raw/oracle_found" ]];then oracle_enum;fi
            if [[ -s "$loot/raw/smb_found" ]];then smb_enum;fi
            if [[ -s "$loot/raw/http_found" ]];then http_enum;fi

            if [[ -s "$loot/raw/windows_found" ]];then windows_enum;fi
            if [[ -s "$loot/raw/linux_found" ]];then linux_enum;fi

}

reg (){
            banner
            upgrade
   OS_guess
            nmap_reg="nmap -p- -O -T4 -Pn -v $IP"
            if [[ ! -d "$IP/autoenum/reg_scan/raw" ]];then mkdir -p $IP/autoenum/reg_scan/raw; fi
            if [[ ! -d "$IP/autoenum/reg_scan/ports_and_services" ]];then  mkdir -p $IP/autoenum/reg_scan/ports_and_services; fi
            tput setaf 6;echo "Checking top 1k ports...";tput sgr0
            nmap --top-ports 1000 -sV $IP | tee -a $IP/autoenum/reg_scan/top_1k
            tput setaf 6;echo -e "Scan complete. View 1k scan at $IP/autoenum/aggr_scan/top_1k\nStarting more comprehensive scan...";tput sgr0
            nmap -sV $IP -oX $IP/autoenum/reg_scan/raw/xml_out & $nmap_reg | tee $IP/autoenum/reg_scan/raw/full_scan;searchsploit -j --nmap
$IP/autoenum/reg_scan/raw/xml_out >> $loot/exploits/searchsploit_nmap
            searchsploit --nmap $IP/autoenum/reg_scan/raw/xml_out
            cat $loot/exploits/searchsploit_nmap | jq >> $loot/exploits/searchsploit_nmap.json
            rm $loot/exploits/searchsploit_nmap

            cat $IP/autoenum/reg_scan/raw/full_scan | grep "open" | awk -F 'Discovered' '{print $1}' | sed '/^$/d' | sed '/|/,+1 d' >>
$IP/autoenum/reg_scan/ports_and_services/services_running
            cat $IP/autoenum/reg_scan/raw/full_scan | grep 'OS' | sed '1d' | sed '$d' | cut -d '|' -f 1 | sed '/^$/d' >>
$IP/autoenum/reg_scan/ports_and_services/OS_detection
            cat $IP/autoenum/reg_scan/raw/full_scan | sed -n '/PORT/,/exact/p' | sed '$d' >>
$IP/autoenum/reg_scan/ports_and_services/script_output

            cat $IP/autoenum/reg_scan/ports_and_services/services_running | grep "http" | sort -u >> $loot/raw/http_found.tmp
            for line in $(cat $loot/raw/http_found.tmp | tr ' ' '-');do echo $line | cut -d '/' -f 1;done >  $loot/raw/http_found;rm $loot/raw/http_found.tmp
            cat $IP/autoenum/reg_scan/ports_and_services/services_running | sort -u | grep "smb" > $loot/raw/smb_found
            cat $IP/autoenum/reg_scan/ports_and_services/services_running | sort -u | grep "snmp" > $loot/raw/snmp_found
#           cat $IP/autoenum/reg_scan/ports_and_services/services_running | sort -u | grep "dns" > $loot/raw/dns_found
            cat $IP/autoenum/reg_scan/ports_and_services/services_running | sort -u | grep "ftp" > $loot/raw/ftp_found
            cat $IP/autoenum/reg_scan/ports_and_services/services_running | sort -u | grep "ldap" > $loot/raw/ldap_found
            cat $IP/autoenum/reg_scan/ports_and_services/services_running | sort -u | grep "smtp" > $loot/raw/smtp_found
            cat $IP/autoenum/reg_scan/ports_and_services/services_running | sort -u | grep "imap" > $loot/raw/imap_found
            cat $IP/autoenum/reg_scan/ports_and_services/services_running | sort -u | grep "pop3" > $loot/raw/pop3_found
            cat $IP/autoenum/reg_scan/ports_and_services/services_running | sort -u | grep "oracle" > $loot/raw/oracle_found
#           cat $IP/autoenum/aggr_scan/ports_and_services/services_running | sort -u | grep "rpc" > $loot/raw/rpc_found
            cat $IP/autoenum/aggr_scan/ports_and_services/services_running | sort -u | grep "redis" > $loot/raw/redis_found

   enum_goto
}

aggr (){
            banner
            upgrade
   OS_guess
            nmap_aggr="nmap -n -A -T4 -p- --max-retries 1 -Pn -v $IP"
            if [[ ! -d "$IP/autoenum/aggr_scan/raw" ]];then mkdir -p $IP/autoenum/aggr_scan/raw; fi
            if [[ ! -d "$IP/autoenum/aggr_scan/ports_and_services" ]];then  mkdir -p $IP/autoenum/aggr_scan/ports_and_services; fi
   tput setaf 6;echo "Checking top 1k ports...";tput sgr0
   nmap --top-ports 1000 -sV $IP | tee -a $IP/autoenum/aggr_scan/top_1k
            tput setaf 6;echo -e "Scan complete. View 1k scan at $IP/autoenum/aggr_scan/top_1k\nStarting more comprehensive scan...";tput sgr0
            nmap -sV $IP -oX $IP/autoenum/aggr_scan/raw/xml_out & $nmap_aggr | tee $IP/autoenum/aggr_scan/raw/full_scan;searchsploit -j
--nmap $IP/autoenum/aggr_scan/raw/xml_out >> $loot/exploits/aggr_searchsploit_nmap
            searchsploit --nmap $IP/autoenum/aggr_scan/raw/xml_out
            cat $loot/exploits/aggr_searchsploit_nmap | jq >> $loot/exploits/aggr_searchsploit_nmap.json;rm
$loot/exploits/aggr_searchsploit_nmap

            cat $IP/autoenum/aggr_scan/raw/full_scan | grep "open" | awk -F 'Discovered' '{print $1}' | sed '/^$/d' | sed '/|/,+1 d' >>
$IP/autoenum/aggr_scan/ports_and_services/services_running
            cat $IP/autoenum/aggr_scan/raw/full_scan | grep 'OS' | sed '1d' | sed '$d' | cut -d '|' -f 1 | sed '/^$/d' >>
$IP/autoenum/aggr_scan/ports_and_services/OS_detection
            cat $IP/autoenum/aggr_scan/raw/full_scan | sed -n '/PORT/,/exact/p' | sed '$d' >>
$IP/autoenum/aggr_scan/ports_and_services/script_output

            cat $IP/autoenum/aggr_scan/ports_and_services/services_running | grep "http" | sort -u >> $IP/autoenum/loot/raw/http_found.tmp
            for line in $(cat $loot/raw/http_found.tmp | tr ' ' '-');do echo $line | cut -d '/' -f 1 ;done > $loot/raw/http_found;rm $loot/raw/http_found.tmp
            cat $IP/autoenum/aggr_scan/ports_and_services/services_running | sort -u | grep "smb" > $loot/raw/smb_found
            cat $IP/autoenum/aggr_scan/ports_and_services/services_running | sort -u | grep "snmp" > $loot/raw/snmp_found
#           cat $IP/autoenum/aggr_scan/ports_and_services/services_running | sort -u | grep "dns" > $loot/raw/dns_found
            cat $IP/autoenum/aggr_scan/ports_and_services/services_running | sort -u | grep "ftp" > $loot/raw/ftp_found
            cat $IP/autoenum/aggr_scan/ports_and_services/services_running | sort -u | grep "ldap" > $loot/raw/ldap_found
            cat $IP/autoenum/aggr_scan/ports_and_services/services_running | sort -u | grep "smtp" > $loot/raw/smtp_found
            cat $IP/autoenum/aggr_scan/ports_and_services/services_running | sort -u | grep "oracle" > $loot/raw/oracle_found
            cat $IP/autoenum/aggr_scan/ports_and_services/services_running | sort -u | grep "pop3" > $loot/raw/pop3_found
            cat $IP/autoenum/aggr_scan/ports_and_services/services_running | sort -u | grep "imap" > $loot/raw/imap_found
#           cat $IP/autoenum/aggr_scan/ports_and_services/services_running | sort -u | grep "rpc" > $loot/raw/rpc_found
            cat $IP/autoenum/aggr_scan/ports_and_services/services_running | sort -u | grep "redis" > $loot/raw/redis_found

   enum_goto
}

top_1k (){
   banner
   upgrade
   OS_guess
            if [[ ! -d "$IP/autoenum/top_1k/raw" ]];then mkdir -p $IP/autoenum/top_1k/raw; fi
            if [[ ! -d "$IP/autoenum/top_1k/ports_and_services" ]];then  mkdir -p $IP/autoenum/top_1k/ports_and_services; fi
   t1k="$IP/autoenum/top_1k"
   nmap --top-ports 1000 -sV -Pn $IP | tee -a $t1k/ports_and_services/services & nmap --top-ports 1000 -sC -Pn $IP >>
$t1k/ports_and_services/scripts
   nmap --top-ports 1000 -sV $IP -oX $t1k/raw/xml_out &
   wait
   searchsploit -j --nmap $t1k/raw/xml_out >> $loot/exploits/top_1k_searchsploit_nmap;searchsploit --nmap $t1k/raw/xml_out
            cat $loot/exploits/top_1k_searchsploit_nmap | jq >> $loot/exploits/top_1k_searchsploit_nmap.json

            cat $t1k/ports_and_services/services | grep "open" |grep "http" | sort -u >> $IP/autoenum/loot/raw/http_found.tmp
            for line in $(cat $loot/raw/http_found.tmp | tr ' ' '-');do echo $line | cut -d '/' -f 1;done >  $loot/raw/http_found;rm $loot/raw/http_found.tmp
            cat $t1k/ports_and_services/services | sort -u | grep "smb" > $loot/raw/smb_found
            cat $t1k/ports_and_services/services | sort -u | grep "snmp" > $loot/raw/snmp_found
#           cat $IP/autoenum/aggr_scan/ports_and_services/services_running | sort -u | grep "dns" > $loot/raw/dns_found
            cat $t1k/ports_and_services/services | sort -u | grep "ftp" > $loot/raw/ftp_found
            cat $t1k/ports_and_services/services | sort -u | grep "ldap" > $loot/raw/ldap_found
            cat $t1k/ports_and_services/services | sort -u | grep "smtp" > $loot/raw/smtp_found
            cat $t1k/ports_and_services/services | sort -u | grep "oracle" > $loot/raw/oracle_found
            cat $t1k/ports_and_services/services | sort -u | grep "pop3" > $loot/raw/pop3_found
            cat $t1k/ports_and_services/services | sort -u | grep "imap" > $loot/raw/imap_found
#           cat $IP/autoenum/aggr_scan/ports_and_services/services_running | sort -u | grep "rpc" > $loot/raw/rpc_found
            cat $t1k/ports_and_services/services | sort -u | grep "redis" > $loot/raw/redis_found

   enum_goto
}

top_10k (){
   banner
   upgrade
   OS_guess
            if [[ ! -d "$IP/autoenum/top_10k/raw" ]];then mkdir -p $IP/autoenum/top_10k/raw; fi
            if [[ ! -d "$IP/autoenum/top_10k/ports_and_services" ]];then  mkdir -p $IP/autoenum/top_10k/ports_and_services; fi
   t10k="$IP/autoenum/top_10k"
   nmap --top-ports 10000 -sV -Pn --max-retries 1 $IP | tee -a $t10k/raw/services & nmap --top-ports 10000 --max-retries 1 -sC -Pn $IP >>
$t10k/raw/scripts
   nmap --top-ports 10000 ---max-retries 1 sV $IP -oX $t10k/raw/xml_out &
   wait
   searchsploit -j --nmap $t10k/raw/xml_out >> $loot/exploits/top_10k_searchsploit_nmap;searchsploit --nmap $t10k/raw/xml_out
            cat $loot/exploits/top_10k_searchsploit_nmap | jq >> $loot/exploits/top_10k_searchsploit_nmap.json
   cat $t10k/raw/services | grep 'open' >> $t10k/ports_and_services/services

            cat $t10k/ports_and_services/services | grep "http" | sort -u >> $loot/raw/http_found.tmp
            for line in $(cat $loot/raw/http_found.tmp | tr ' ' '-');do echo $line | cut -d '/' -f1;done > $loot/raw/http_found;rm $loot/raw/http_found.tmp
            cat $t10k/ports_and_services/services | sort -u | grep "smb" > $loot/raw/smb_found
            cat $t10k/ports_and_services/services | sort -u | grep "snmp" > $loot/raw/snmp_found
            cat $t10k/ports_and_services/services_running | sort -u | grep "dns" > $loot/raw/dns_found
            cat $t10k/ports_and_services/services | sort -u | grep "ftp" > $loot/raw/ftp_found
            cat $t10k/ports_and_services/services | sort -u | grep "ldap" > $loot/raw/ldap_found
            cat $t10k/ports_and_services/services | sort -u | grep "smtp" > $loot/raw/smtp_found
            cat $t10k/ports_and_services/services | sort -u | grep "oracle" > $loot/raw/oracle_found
            cat $t10k/ports_and_services/services | sort -u | grep "pop3" > $loot/raw/pop3_found
            cat $t10k/ports_and_services/services | sort -u | grep "imap" > $loot/raw/imap_found
            cat $t10k/ports_and_services/services_running | sort -u | grep "rpc" > $loot/raw/rpc_found
            cat $t10k/ports_and_services/services | sort -u | grep "redis" > $loot/raw/redis_found

   enum_goto
}

udp (){
   banner
   upgrade
   OS_guess
            if [[ ! -d "$IP/autoenum/udp/raw" ]];then mkdir -p $IP/autoenum/udp/raw; fi
            if [[ ! -d "$IP/autoenum/udp/ports_and_services" ]];then  mkdir -p $IP/autoenum/udp/ports_and_services; fi
            udp="$IP/autoenum/udp"
   nmap -sU --max-retries 1 --open $IP | tee -a $udp/scan

}

vuln (){
            mkdir -p $loot/exploits/vulns
            vulns="$loot/exploits/vulns"
            cwd=$(pwd)

            if [[ ! -d "/usr/share/nmap/scripts/vulscan" ]];then
            cd
            git clone https://github.com/scipag/vulscan scipag_vulscan
            ln -s `pwd`/scipag_vulscan /usr/share/nmap/scripts/vulscan
            cd $cwd
```

```
        fi

        nmap -sV --script=vulscan/vulscan.nse $IP | tee -a $vulns/vulscan
        nmap -Pn --script vuln $IP | tee -a $vulns/vuln
}
```

Output

```
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:            65.61.137.117
+ Target Hostname:      65.61.137.117
+ Target Port:          443
---------------------------------------------------------------------------
+ SSL Info:     Subject:  /CN=demo.testfire.net
                Ciphers:  ECDHE-RSA-AES256-GCM-SHA384
                Issuer:   /C=GB/ST=Greater Manchester/L=Salford/O=Sectigo Limited/CN=Sectigo RSA Domain Validation Secure
Server CA
+ Start Time:           2022-07-17 14:36:46 (GMT5.5)
---------------------------------------------------------------------------
+ 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 user agent to protect against some forms of XSS
+ The site uses SSL and the Strict-Transport-Security HTTP header is not defined.
+ The site uses SSL and Expect-CT header is not present.
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different
fashion to the MIME type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
```

```
Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-17 14:35 IST
Nmap scan report for 65.61.137.117
Host is up (0.31s latency).
Not shown: 996 filtered tcp ports (no-response)
PORT STATE  SERVICE
80/tcp   open   http
|_http-title: Altoro Mutual
443/tcp  open   https
|_ssl-date: 2022-07-17T09:06:44+00:00; +59s from scanner time.
|_http-title: Altoro Mutual
| ssl-cert: Subject: commonName=demo.testfire.net
| Subject Alternative Name: DNS:demo.testfire.net, DNS:altoromutual.com
| Not valid before: 2022-06-15T00:00:00
|_Not valid after:  2023-07-16T23:59:59
8080/tcp open   http-proxy
|_http-title: Altoro Mutual
|_http-open-proxy: Proxy might be redirecting requests
8443/tcp closed https-alt

Host script results:
|_clock-skew: 58s

Nmap done: 1 IP address (1 host up) scanned in 51.71 seconds
```

```
Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-17 14:35 IST
Nmap scan report for 65.61.137.117
Host is up (0.32s latency).
```

```
Not shown: 996 filtered tcp ports (no-response)
PORT     STATE  SERVICE   VERSION
80/tcp   open   http      Apache Tomcat/Coyote JSP engine 1.1
443/tcp  open   ssl/http  Apache Tomcat/Coyote JSP engine 1.1
8080/tcp open   http      Apache Tomcat/Coyote JSP engine 1.1
8443/tcp closed https-alt

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 42.50 seconds
```

References

- Wikipedia (https://www.wikipedia.org/)
- NMAP cheat sheet
- DNS recon studyguide
- Searchsploit modules
- Chuck Keith on bash scripting
- Hackerone
- Hackthebox testing labs