Network Research Project

Author : Adir Salinas (code S18)

Class code : 7736

Lecturer : Natali Erez

This document will talk about the Network Research project:

First, we will talk about the purpose of this project and will display a short explanation and few details about the structure of the project.

Secondly, we will display Topology – a diagram that demonstrate the project in a visual way and in a more convenient way to understand the project.

Finally, The last part will give an instructions of how to use the script and the different sections, and after explain in details about each section of the project, and will break down the entire script and each command separately and the purpose of them.

What the project does and what the purpose of it ?

In the internet world, like everything in the world there is a good side but there are also things that are not always good and allowed to be done and might be illegal to do.
In such cases the experienced people who do these kind of things, will always try to hide themselves and be anonymous as possible so they can't be tracked down.

An example of one of the things that is illegal to do is “unauthorized scanning”.

So what does that mean?
Unauthorized scanning involves probing or examining computer networks or domains without proper permission or authorization, often for assessing vulnerabilities or identifying open ports without the explicit consent of the network owner.

There are several ways to perform these kind of scans and we will discuss later on about them in the script.

So what will the script perform?

The script will focus on unauthorized scanning of a chosen Domains or IP addresses.
Before any scan being executed, the script was built to make sure that your internet connection is completely unassociated with your location and you are totally anonymous before moving on to perform the scans.

The second part of the script will connect remotely to another server that will perform these scans for you in order to hide yourself more and make it even more difficult to tracked you, if they will try to catch whoever did these scan, the server you connected to will be tracked.

The last part of the script will save all the data, details and information that the server scan in files and will transfer them into your computer safely without you being exposed.

The work environment and goal of the script

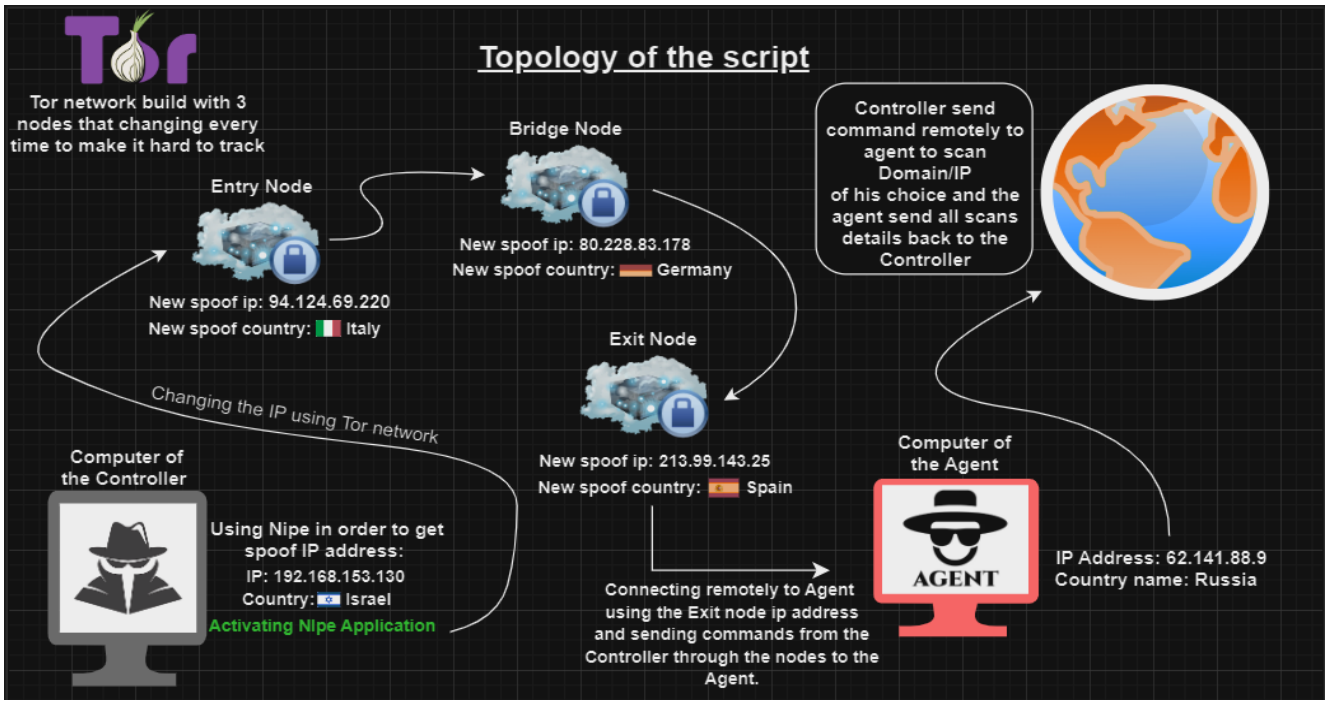
The goal of the script is to create a system for reconnaissance purposes in which the Controller manages the Agent automatically and send him instructions to scan sources for the Controller which is performed automatically and transfer the information of the scans automatically.

- Controller (Client)

- Agent (Server)

Topology of the script

Note: all the IP addresses and countries are just for demonstration.



The first step on the topology is activating the Nipe application.

So what is Nipe application?

Nipe is an application that in the script we will installed for you in your computer. The application purpose is to change the ip address and country to a different then yours, the Nipe using Tor network in order to make this changes.

So what is Tor network?

Tor network is an open-source privacy network that enables anonymous web browsing, Tor will make anonymously by routing your traffic through a network comprised of thousands of volunteer-run servers called nodes.
Every time you connect to Tor, it will choose three nodes that every node connects you to a different network which changes your ip and country every certain period of time. (like shown in the topology).

The next step after you connected successfully through Tor network using Nipe application that was been installed, the script will connect you remotely to an another server (agent) that will executed the commands that the script send him for you according to the Domains/IP addresses you chosen to scan.

And finally will save all the information and data scan into files and send them safely to your computer.

Script structure and operating instructions

Setting up a server (agent):

First of all, before starting the script make sure you have a working and functional server that we can use as our Agent, In order to enable remote control please make sure that the server is ready and set to establish a connection via ssh.
If not, please make sure ssh service is installed on it in order to connect to it remotely.

how to install ssh service?

For every command you instruct to execute first you need to open Linux if you don't have download it, then click in the desktop **"ctrl+alt+T"** to open terminal.

First execute the command **"sudo apt update"** by typing in the terminal to update your system and make sure all the latest links and packages are updated.
Then run he command **"sudo apt install openssh-server"** to install the ssh service, after you installed the service make sure the service is running by executing the command **"sudo service ssh start"** after you started the service run the command **"sudo netstat -tlp"** this command will provide you with information about the status of the ssh service and the port it is using, so make sure the server State is on "LISTEN" mode this state means that ssh is installed, running and ready to connect.

In order to to send the Agent server a commands and order him to scans domains and ip addresses for us the server need to have the ability to scan.
The applications we will use in this project is **Whois, Nmap and Geoiplookup.**
If your Agent server already set up and those applications are already running on the server you good to go and can move on to the script instructions.
If you need to install this applications follow the following steps.

First make sure your system is updated by using the command: "**sudo apt update"**

To install Whois application:
Open the terminal and use the command: **"sudo apt-get -y install whois".**

To install Nmap application:
Open the terminal and use the command: **"sudo apt-get -y install nmap".**

To install Geoiplookup application:
Open the terminal and use the command: **"sudo apt-get -y install geoip-bin".**

After the server (Agent) is set up we can continue to the script instructions

Script sections and instructions:

To run the script download the script file to your Linux and type in the terminal **"bash nr.project.sh"**

The first section of the script will display text of the start of the script, will show the purpose of the script in a few words.

The second section of the script will update your system before installing the needed application in order to make sure that all the links and packages are the latest.
You might need to insert the password of your system when the script asks you to.

The third section of the script will run tests on the system to check if the necessary applications is installed, If not all the necessary applications are installed, it will install for you all the missing applications automatically.

The fourth section of the script will check whether your network connection anonymous before we remotely connect to the server.
It will ask you to insert your country in **lower case** when needed, after the test over and your network connection if not anonymous, it will restart Nipe application again and will reveal your new spoof country and ip address of the anonymous connection.

The fifth section of the script after the connection is fully anonymous and not associated with your details, this section will preform remote control to the server that will scan for you the domains/ip addresses.
In order to connect successfully you will need to insert the following details:

- IP address of the server: open the terminal in the server and type **" hostname -I "**.
- Username of the server: open the terminal in the server and type **" whoami "**.
- Password of the server: insert the password you used when logged into the server.

After you insert the credentials of the server (Agent) it will check if the details are correct and can and managed to connect to it successfully it will display the the IP address of the server, the country associated with the server and displaying the uptime of the server.
Uptime - indicating how long a system has been operational since its last restart or reboot, and how many user are connected to the server and will display the average system load.

The sixth section of the script will be related to scanning a domains and ip addresses.
After you connected successfully to the server it will ask you to insert a Domain/ip of your choice and will check whether your Domain/ip is valid or invalid.
If its valid will run scans on the Domain/ip and display the ip address associated the country address and display few known ports that are open on the Domain/ip (open ports can be an indicator and show vulnerability of the Domain/ip you chosen).
After you sucssuflly scanned the Domain/ip it will ask you if you want to scan another one by input the text **'y'** (for yes) and to scan another one, or type **'n'** (for no) and you will move on.

The last part of the script will show text of the end of the script and will save all your files of the scans in your Desktop under a directory called **"Project.files"** you would have a case choice to choose if you want to display on screen the files that was created or exit the script and view them at your on time.

Explaining the whole script and each command

The last part of this document will explain everything about command, script and more in order to understand everything, and will break down each part of the script and the purpose of the different commands. **So Let's Start…**

What is a command?

In computing, a "command" refers to a specific instruction or directive given to a computer or software to perform a particular action. Commands are typically issued through a command-line interface (CLI) or a terminal, where users enter text-based instructions to interact with the operating system or software applications to do an action. It can be a single word, a line of code, or a series of instructions that tell the computer what to do.
Later we will talk about different commands and what each command instructs the computer to do.

What is a script?

In computing, a "script" is a set of instructions or number of commands that are written in a programming language to perform a specific task or automate a series of tasks. The script executed the commands or tasks by reading them line by line.
In simple words instead of typing each command every time, you writing all the command you want to execute in a script and when you run the script on a computer then all the commands you written will be executed one after the other.

What is a function?

In programming, a "function" is a set of commands grouped together to perform a specific task. Functions allow you to break down your script into smaller, more manageable pieces. Every function have a name that you set when creating the function and the function will be operate and execute only when we call the name of the function, if we are not calling the function the function will not be operated and the commands inside will not work.

What is a statement?

In programming, a "statement" is a single line of code that performs a specific action. It is the basic building block of a program and represents an executable instruction. Statements are the means by which a programmer gives instructions to a computer to perform tasks, make decisions, or control the flow of the program.
In simple words, a statement It's a line of code that performs a specific action, such as assigning a value to a variable, printing something on the screen, or making a decision based on certain conditions. Each statement is like a step in a set of instructions that the computer follows to carry out a task.

What is a variable?

In programming, a "variable" is a container or placeholder for storing data. It has a name and a value, and the value can change as the program runs. Variables are used to store and manipulate information within a program. in simple terms, think of a "variable" like a box with a name. You can put stuff (information or data) in the box, for example set a variable with a certain name and save inside the name a command output, when you call the variable name later, it will print in screen the output or data.

Lets talk about few important commands that will be executed in the script :

"echo" - Making space between commands or displaying text on screen.

"clear" - clearing the output terminal screen.

"sleep" - this command used to introduce a delay or pause in the script for a certain period of time. Most of the time there is a number after the command to wait in seconds.

" cd " - Short for "change directory," is used in the script and allows you to navigate between directories and move to a different location.

"grep" - searching for word or sentence inside an output and print them on screen.

"awk" - have a lot of used most of them in the script is to separate certain words or location.

"read" - read command reads the user's input and assigns it to the variable, using the flag (-p) after the command for printing text on screen before the input.

"nmap" - is an open-source network scanning tool used for discovering devices, services, and vulnerabilities on computer networks.

"whois" - also a scanning tool that provides information and details about a domain names, including details about the domain's owner, registrar, and registration dates.

“sudo” - command is used to execute commands with elevated privileges. and it allows a permitted user to execute a command as the superuser.

“sudo apt-get install" - command to install diffrent packages or applications.

"> /dev/null 2>&1" - this command used in addition to another command to execute the command quietly without displaying any output or errors.

Functions names, their order and short description:

Functions for update the system to make sure we have the needed links and recent packages: **UPDATE**

Functions for installing applications : **NMAP, GEO, SSHPASS, WHOISAPP, NIPE.**

Functions that display text animation for installing the application : **INSTALL**

Functions to check if the application is installed: **NMAPINFO, GEOINFO, SSHPASSINFO, WHOISINSTAL, NIPEINFO.**

Functions to verify whether your network connection is anonymous **: ANON , ANONCNRY.**

Functions to check anonymity and if Nipe is not running:
FALSE - (restarting the Nipe to start it).
TRUE - (verify the user country does not match the Nipe country).

Functions to extract the information about the attempts to connect to the server:
LOG - (commands to check if the connection was successful).
SCANLOG - (the output to the file according to the LOG function).

Functions to create a file for the different scans output that was executed:
NMAPSCAN - (nmap scan for the domain).
WHOIS - (whois scan for the domain).

Function that allows the user to enter details of the ssh server, after the commands checks that the details are correct, displays information about the ssh server: **VPS**

Function that allows the user to enter a Domain/ip , after that run checks that the Domain/ip are correct and valid then displays information about the Domain/ip: **DMN**

Function that finishes the script and display text where the log and scans file are locate and case options to display them on screen: **END**

Braking down the script to parts and explain each part in details

(Basic command that was explained before will not have description about them)
Inside the functions there will be other functions names that getting called when needed!

first part before the START function there is a install command for application.

```
clear
echo "[!] You may need to enter a password for the script:"
echo
sleep 1.5
sudo apt-get install -y figlet > /dev/null 2>&1 # command t
```

Display text on screen using echo to enter a password for the installation command.
Then installing '**figlet'** package: "sudo apt-get install -y figlet > /dev/null 2>&1"
The figlet command generates normal text into ASCII art text banners using various font styles. It's for visual purpose only. (The flag -y is for automatically answer "yes" for questions)

The next function is START the first function of the script that display short description about the purpose of the script and few details.

```
function START () # Function for the start of the script and display short description about the purpose of the script and few details.
{
    clear
    figlet -t -c -f standard "Welcome To My Sciprt !" # this command convert text to ascii(more artistic way),"-t" for expanding to the entire screen,"-c" for centering
    sleep 3
    echo
    echo
    echo " The Purpose Of The Script Is To Scan Different Domains And IP Addresses While Being Completely Anonymous And Collect Data Scans To Your Computer Safely. "
    sleep 8
    echo
    echo " It will be performed as following: "
    echo
    sleep 2
    echo " - Master "
    sleep 1
    echo " - Agent"
    echo
    sleep 2
    echo "        You Will Be Operated As The Master, Firstly You Gonna Perform Remote Control On Your Agent After We Make Sure You Completely Anonymous. "
    sleep 7.5
    echo
    echo "                Then You Will Send Commands To The Agent That Will Performed Several Scans For Domains Or IP Addresses That You Wanted.  "
    sleep 7
    echo
    echo "                        And Finally Will Transfer All The Scan Data To A File On Your Computer In The Most Safest Way Possible. "
    sleep 6
    echo
    echo "                                                                So Lets Start..."
    sleep 2.5
}
START
```

The command " figlet -t -c -f standard "Welcome To My Sciprt ! " :
convert text to ascii (more artistic way), "-t " for expanding to the entire screen, "-c" for centering the text, and "-f " for specify which font to use in this case its "standard" font.

Next function is UPDATE this function is to update the system in order to make sure to have the recent packages and links of the applications.

```
function UPDATE () # Function is to updating the system in order to make sure to have the recent packages and links of the applications.
{
    echo
    echo
    echo
    echo "[*] Before We Start Please Wait While We Updating Your System...  "
    sleep 1
    echo "                (It may take a few minutes) "
    sleep 1.5
    echo
    echo
    sudo apt update -y &>/dev/null # commmand to update your system
    echo
    echo
    echo "[+] Your System Has Been Updated Successfully."
    echo
    sleep 2
    echo "[?] Progressing To Check If The Necessary Applications Are Installed. "
    echo
    echo
    sleep 2
}
UPDATE
```

The command “sudo apt update -y &>/dev/null” :
This command responsible for updating your system.

The next part for installing small needed applications:

```
sudo apt-get install -y lolcat > /dev/null 2>&1
sudo apt-get install -y cmatrix > /dev/null 2>&1
```

The application lolcat used to output text with rainbow-colored letters and changing the color of the text, for visual purpose only.

The application cmatrix used to create a visually dynamic and animated display of characters on the terminal screen. for visual purposes only.

The following functions is for downloading the necessary applications if they are not installed.

```
function NMAP () # Script to download nmap.
{
    sudo apt-get -qq -y install nmap 1> /dev/null  # apt-get -y said say "yes" to all que
}
#NMAP


function GEO () # Script to download Geoiplookup.
{
    sudo apt-get -qq -y install geoip-bin 1> /dev/null # apt-get -y said say "yes" to all
}
#GEO

function SSHPASS () # Script to download Sshpass.
{
    sudo apt-get install -y sshpass > /dev/null 2>&1 # apt-get -y said say "yes" to all c
}
#SSHPASS

function WHOISAPP () # Script to download whois.
{
    sudo apt-get -qq -y install whois 1> /dev/null  # apt-get -y said say "yes" to all qu
}
#WHOISAPP

function NIPE () # Script to download Nipe.
{
    cd ~ > /dev/null 2>&1  # Navigate back to the the home directory to make sure Nipe is
    git clone https://github.com/GouveaHeitor/nipe > /dev/null 2>&1 && cd nipe # Git clor
    sudo apt-get install -y cpanminus > /dev/null 2>&1  # installing "cpanminus" package
    sudo cpanm install Switch JSON LWP::UserAgent Config::Simple > /dev/null 2>&1 # "cpar
    sudo perl nipe.pl install > /dev/null 2>&1 # "perl" is used to run Perl scripts, and
    sudo perl nipe.pl start > /dev/null 2>&1 # and then specify the word start for the sc
}
#NIPE
```

When using the command “sudo apt get -qq -y install “:
apt-get ‘’-y’’ said say "yes" to all questions, and "-qq" means "do things quietly" suppresses most of the output.

Explaining the Nipe installation commands:

cd ~ > /dev/null 2>&1 :
Navigate back to the home directory to make sure Nipe will be installed there.

git clone https://github.com/GouveaHeitor/nipe > /dev/null 2>&1 && cd nipe :
Git clone command will download the entire content of the URL given and create a copy directory named "nipe", then entering the directory.

sudo apt-get install -y cpanminus > /dev/null 2>&1:
installing "cpanminus" package allowing you to easily manage Perl modules.

sudo cpanm install Switch JSON LWP::UserAgent Config::Simple > /dev/null 2>&1:
"cpanm" is used to install Perl modules, and "Switch JSON LWP::UserAgent Config::Simple" are the name of the perl modules.

sudo perl nipe.pl install > /dev/null 2>&1:
"perl" is used to run Perl scripts, and then specify the name of the script "nipe.pl".

sudo perl nipe.pl start > /dev/null 2>&1:
"perl" is used to run Perl scripts, and then specify the word start for the script "nipe.pl to executed the according commands in the script to start Nipe.

Next function called INSTALL the function used to show animated text, that getting called when installing an application or package.

```
function INSTALL () # Function to show animated text, that getting called when installing an application or package.
{
        echo
        echo "[!] Your Application Is Not Installed, Progressing To Download... "
        echo
        sleep 1.5
        echo "[*] Continuing To Install The Application."
        sleep 3
        clear
        echo "> Installing the application. "
        sleep 0.5
        clear
        echo "> Installing the application.. "
        sleep 0.5
        clear
        echo "> Installing the application... "
        sleep 0.5
        clear
        echo "> Installing the application. "
        sleep 0.5
        clear
        echo "> Installing the application.. "
        sleep 0.5
        clear
        echo "> Installing the application... "
        sleep 2
}
```

Function to check if Nmap application is installed.

```
function NMAPINFO () # Function to check if Nmap application is installed.
{
    if [ -f /usr/bin/nmap ] # For the following syntax: "-f" is used to chec
    then
        echo "[?] Checking If NMAP Is Already Installed... "
        sleep 3
        echo
        echo "[*] The Application Is Installed."
        sleep 2
    else
        clear
        echo "[?] Checking If NMAP Is Already Installed... "
        sleep 3
        echo
        INSTALL
        echo
        NMAP
        clear
        echo "[+] The Appliction Has Been Successfully Downloaded!"
        sleep 2
    fi
}
NMAPINFO
```

In the function there is a if statement that said the following condition:
"-f " is used to check if the file exist in the following path. if the file exist went to "then", if the file not existing went to "else". The following path that was given is the location of where the application should be located. So if its not existing in the path calling the function that install.

Function to check if Geoiplookup application is installed.

```
function GEOINFO () # Function to check if Geoiplookup application is installed.
{
    if [  -f /usr/bin/geoiplookup ]  # For the following syntax: "-f" is used to che
    then
        echo
        echo
        echo "[?] Checking If GeoIpLookup Is Already Installed... "
        sleep 3
        echo
        echo "[*] The Application Is Installed."
        sleep 2
    else
        echo
        echo
        echo "[?] Checking If GeoIpLookup Is Already Installed... "
        sleep 3
        echo
        INSTALL
        echo
        GEO
        clear
        echo "[+] The Appliction Has Been Successfully Downloaded!"
        sleep 2
    fi
}
GEOINFO
```

In the function there is a if statement that said the following condition:
"-f " is used to check if the file exist in the following path. if the file exist went to "then", if the file not existing went to "else". The following path that was given is the location of where the application should be located. So if its not existing in the path calling the function that install.

Function to check if Sshpass application is installed.

```
function SSHPASSINFO () # Function to check if Sshpass application is installed.
{
    if [ -f /usr/bin/sshpass ] # For the following syntax: "-f" is used to check i
        then
        echo
        echo
        echo "[?] Checking If SshPass Is Already Installed... "
        sleep 3
        echo
        echo "[*] The Application Is Installed."
        sleep 2
    else
        echo
        echo
        echo "[?] Checking If SshPass Is Already Installed... "
        sleep 3
        echo
        INSTALL
        echo
        SSHPASS
        clear
        echo "[+] The Appliction Has Been Successfully Downloaded!"
        sleep 2
    fi
}
SSHPASSINFO
```

In the function there is a if statement that said the following condition:
"-f " is used to check if the file exist in the following path. if the file exist went to "then", if the file not existing went to "else". The following path that was given is the location of where the application should be located. So if its not existing in the path calling the function that install.

Function to check if whois application is installed.

```
function WHOISINSTAL () # Function to check if whois application is installed.
{
    if [ -f /usr/bin/whois ] # For the following syntax: "-f" is used to check if
        then
        echo
        echo
        echo "[?] Checking If Whois Is Already Installed... "
        sleep 3
        echo
        echo "[*] The Application Is Installed."
        sleep 2
    else
        echo
        echo
        echo "[?] Checking If Whois Is Already Installed... "
        sleep 3
        echo
        INSTALL
        echo
        WHOISAPP
        clear
        echo "[+] The Appliction Has Been Successfully Downloaded!"
        sleep 2
    fi
}
WHOISINSTAL
```

In the function there is a if statement that said the following condition:
"-f " is used to check if the file exist in the following path. if the file exist went to "then", if the file not existing went to "else". The following path that was given is the location of where the application should be located. So if its not existing in the path calling the function that install.

Function to check if Nipe application is installed.

```
function NIPEINFO () # Function to check if Nipe application is installed.
{
    cd ~ && cd nipe > /dev/null 2>&1 # Change the location to home directory and then entering nipe di
    path=$(pwd)  # executing 'pwd' command (outputs the current directory path) and insert it into a v
    NIPEPATH=$path/nipe.pl  # adding "nipe.pl" to the "pwd" command to make a full path to the nipe.pl
    if [ -f $NIPEPATH ]  # For the following syntax: "-f" is used to check if the file exist in the fo
    then
        echo
        echo
        echo "[?] Checking If NIPE Is Already Installed... "
        sleep 3
        echo
        echo "[*] The Application Is Installed."
        sleep 2
    else
        echo
        echo
        echo "[?] Checking If NIPE Is Already Installed... "
        sleep 3
        echo
        INSTALL # calling the function called INSTALL and executes the code inside the function.
        NIPE # calling the function called NIPE and executes the code inside the function.
        clear
        echo "[+] The Appliction Has Been Successfully Downloaded!"
        sleep 2
    fi

}
NIPEINFO
```

Explaining the command before the if statement:

“cd ~ && cd nipe” : Change the location to home directory and then entering nipe directory.
“path=$(pwd)” : executing 'pwd' command (outputs the current directory path) and insert it into a variable named "path".
“NIPEPATH=$path/nipe.pl” : adding "nipe.pl" to the variable “path” ("pwd" command) to make a full path to the nipe.pl file, and insert it into a variable named "NIPEPATH".

In the function there is a if statement that said the following condition:
"-f " is used to check if the file exist in the following path. if the file exist went to "then", if the file not existing went to "else". The following path that was given is the location of where the application should be located. So if its not existing in the path calling the function that install.

Function to check if the network connection is anonymous and ask the user to type his country.

```
function TRUE () # Function to check if the network connection is anonymous and ask the user to type his country.
{
    clear
    echo
    echo "[*] To Perform Remote Control And Scaning Other Domains, You Need To Be Anonymous."
    sleep 3
    echo
    echo "[?] Checking If Your Network Connection Is Anonymous... "
    sleep 2
    # IP Address:
    IPNIPE=$(sudo perl nipe.pl status | grep Ip | awk '{print $3}') # getting status from Nipe about the spoof IP
    # Country name:
    NIPECOUNTRY=$(geoiplookup $IPNIPE |awk '{print $5}'|tr '[:upper:]' '[:lower:]') # using geoiplookup on the va
    echo
    echo
    sleep 1
    echo "[*] To display If You Are Anonymous Please Insert Your Country. "
    sleep 1
    echo
    read -p " >  Write Your Country In Lower Case Only! : " USERCOUNTRY # read command used to read the input wor
}
#TRUE
```

Explaining the commands in the following function :

" IPNIPE=$(sudo perl nipe.pl status | grep Ip | awk '{print $3}') " :
getting status from Nipe about the spoof IP, after that using grep command to grep Ip row and using awk command to separate the ip address, and insert it into a variable named "IPNIPE".

" NIPECOUNTRY=$(geoiplookup $IPNIPE |awk '{print $5}'|tr '[:upper:]' '[:lower:]') " :
using geoiplookup on the variable "IPNIPE", then using the command awk to separate the spoof country name, after we have the name using " tr " command to make the country in lower case only, and insert it into a variable named "NIPECOUNTRY".

" read -p " > Write Your Country In Lower Case Only! : " USERCOUNTRY " :
read command used to read the input word the user typing, and insert it into a variable named "USERCOUNTRY", (the flag -p used for printing text on screen before the input).

Function to restart and starting the Nipe application, in case the nipe didn't work and didn't display the new spoof IP.

```
function FALSE () # Function to restarting and starting the Nipe applicatio
{
    sudo perl nipe.pl restart # inserting "restart" variable inside the nip
    sudo perl nipe.pl start # inserting "start" variable inside the nipe.pl
    sudo perl nipe.pl status # inserting "status" variable inside the nipe.
}
#FALSE
```

" sudo perl nipe.pl restart " : inserting "restart" variable inside the nipe.pl script that executed the commands inside the script to restart the Nipe application

" sudo perl nipe.pl start " : inserting "start" variable inside the nipe.pl script that executed the commands inside the script to start the Nipe application

" sudo perl nipe.pl status " : inserting "status" variable inside the nipe.pl script that executed the commands inside the script to display the IP status.

Function to check whether the status of the Nipe is working and change the ip address, if not call the function to restart Nipe in order to be anonymous.

```
function ANON() # Function to check whether the status of the Nipe is working and change the i
{
    cd ~ && cd nipe > /dev/null 2>&1  # Change the location to home directory and then enterir
    sudo perl nipe.pl stop > /dev/null 2>&1  # inserting "stop" variable inside the nipe.pl sc
    sudo perl nipe.pl restart > /dev/null 2>&1  # inserting "restart" variable inside the nipe
    sudo perl nipe.pl start > /dev/null 2>&1  # inserting "start" variable inside the nipe.pl
    echo

    ifon=$(sudo perl nipe.pl status | grep -io true|tr '[:upper:]' '[:lower:]')  # getting the
    ifoff=$(sudo perl nipe.pl status | grep -io false|tr '[:upper:]' '[:lower:]') # getting th

    if [ $ifon = "true" ] # For the following syntax: if the variable "ifon" equal to "true" c
    then
        TRUE  # calling the Function called TRUE
    elif [ $ifoff = "false" ] # the 'elif' statment is alternative action if the previous conc
    then
        FALSE  # calling the Function called FALSE
        TRUE  # calling the Function called TRUE
    fi
}
ANON
```

“ cd ~ && cd nipe “ : Change the location to home directory and then entering nipe directory.

“ sudo perl nipe.pl stop “ : inserting "stop" variable inside the nipe.pl script that executed the commands inside the script to stop the Nipe application.

“ sudo perl nipe.pl restart “ : inserting "restart" variable inside the nipe.pl script that executed the commands inside the script to restart the Nipe application

“ sudo perl nipe.pl start “ : inserting "start" variable inside the nipe.pl script that executed the commands inside the script to start the Nipe application

“ ifon=$(sudo perl nipe.pl status | grep -io true|tr '[:upper:]' '[:lower:]') “ :
Getting the status output of Nipe, using the command grep the word "true" then using 'tr' command to make the “true” word lower case only, and insert it into a variable named "ifon ".

“ ifoff=$(sudo perl nipe.pl status | grep -io false|tr '[:upper:]' '[:lower:]') “ :
Getting the status output of Nipe, , using the command grep the word "false" then using 'tr' command to make “false” word in lower case only, and insert it into a variable named " ifoff ".

In the function there is a if statement that said the following condition:
The following statement said the if the variable "ifon" equal to "true" go continue to "then" statement.In the statement there is a statement called “ elif ” the elif statement is alternative action if the previous conditions are not met. So the next condition said that if the variable "ifoff" equal to "false" so continue to "then" statement.

Function to check if network connection is anonymous if it is displaying details about the new spoof ip and country, and if not restarting Nipe to make connection anonymous and then displaying the new spoof details.

```
function ANONCNTRY () # Function to check if network connection is anonymous if it is display detail
{
    if [ $NIPECOUNTRY = $USERCOUNTRY ] # If statment that if NIPECOUNTRY equal to the USERCOUNTRY, m
    then
        sleep 2
        clear
        echo "[?] Checking If Your Network Connection Is Anonymous... "
        sleep 3
        echo "[*] Your Network Connection Is Not Anonymity! "
        sleep 2
        echo "[-] Progressing To Try Again To Make You Anonymous..."
        sleep 4
        ANON  # Calling ANON function for restarting Nipe in order to make connection anonymous
        ANONCNTRY  # Calling ANONCNTRY again to check if both of the country is equal that verify an
    else
        sleep 3
        clear
        echo "[!] Your Network Connection Is Completely Anonymous And Safe To Continue. "
        echo
        sleep 3
        echo " > Your New Spoofed Country name: $NIPECOUNTRY"
        sleep 1
        echo
        echo " > Your New Spoofed IP Address: $IPNIPE"
        echo
        sleep 2.5
        echo "[*] Connecting To A Remote Server ... "
        sleep 3
        echo
    fi
}
ANONCNTRY
```

In the function there is a if statement that said the following condition:

The If statement said that if NIPECOUNTRY variable equal to the USERCOUNTRY variable, means that if the country the user input is different then the Nipe spoof country the connection is anonymous and can move to 'else' statement.
if they are equal, restart Nipe by moving to “then” statement.
Inside the statement the script called to the different functions that responsible for starting Nipe and making connection fully anonymous.

Function to create a log file structure and extract all the needed details through the ssh connection to make the log file. (this function being called after the user enters the ssh server details).

```
function LOG () # Function to create a log file stracture and extracting all the needed details through the ssh connection to make the t
{

    function ACCFAIL () # This Function will try to connect to an ssh server according to the details given by the user, if the connecti
    {
    sshpass -p $PASS ssh -o StrictHostKeyChecking=no $USR@$IP exit # "sshpass" used for entering ssh server password (-p flag used to en

    if [ $? -eq 0 ]; # is a conditional expression that checks whether the last command was successful. "$?"- holds the exit status of t
    then
        echo "Accepted Password"
    else
        echo "Failed Password"
    fi
    }

accfail=$(ACCFAIL) # getting the output of the echo of ACCFAIL function and insert it into a variable named "accfail".

# Display The list all processes that run on the device and cut the number of PID of sshd and display on screen to add inside the logfil
pid=$(sshpass -p "$PASS" ssh -o StrictHostKeyChecking=no "$USR@$IP" 'ps -e | grep -i sshd | head -n1' 2>/dev/null | awk '{print $1}') #

if [ -z "$pid" ];  # The flag -z is to check if "pid" variable is empty if so move to 'then' statment and print Unknown when using the '
then
    pid="Unknown"
fi

# "netstat" command is used to provides information about network connections, routing tables, interface statistics and more.
# Send netstat command to the device to display all the ESTABLISHED connections (netstat -t flag display information about TCP connectio
portnum=$(sshpass -p $PASS ssh -o StrictHostKeyChecking=no $USR@$IP "netstat -t |grep -i ESTABLISHED|grep -i ssh") # after execute the c
PORTNUM=$(echo "$portnum" |awk '{print $5}'|awk -F: '{print $2}') # echo the variable "portnum" that contain the all row of the random p

# The command (date +'%b %e %H:%M:%S') will say what to display while using 'date' (%b-format represents month name)(%e-format represent
cd ~ && cd Desktop # Change the location to home directory and then entering Desktop directory.
mkdir Project.files > /dev/null 2>&1 # ("mkdir" command used for creating directories) making new directory inside Desktop named "Projec
cd ~ && cd Desktop && cd Project.files # Change the location to home directory and then entering Desktop directory and entering the new
echo "$(date +'%b %e  %H:%M:%S') sshd[$pid]: $accfail for $USR from $IPNIPE port $PORTNUM ssh2." >> NR.log # making the stracture of the
```

Inside the ‘LOG’ function there is another function called ‘ACCFAIL’ that creates the if statement if the connection to the server was accomplished or failed via the user given details. **(This function getting called only when needed, and after the user gave details.)**

The ‘ACCFAIL’ function will try to connect to an ssh server according to the details given by the user, if the connection was successful or not, the event will be displayed in the logfile.

“ sshpass -p $PASS ssh -o StrictHostKeyChecking=no $USR@$IP exit “ :
"sshpass" used for entering ssh server password (-p flag used to enter single password, -P flag used to enter passwords file), "-o StrictHostKeyChecking=no" This option disables strict host key checking. "exit" for immediately exits the session after ssh server was checked.

The if statement inside ACCFAIL function is a conditional expression that checks whether the last command was successful. " $? " - holds the exit status of the last executed command. A value of 0 usually means success. So, if its equal to 0 then continue to 'then' statement if not so continue to 'else' statement.

“ accfail=$(ACCFAIL) “ : getting the output of the echo inside ACCFAIL function and insert it into a variable named "accfail".

“ pid=$(sshpass -p "$PASS" ssh -o StrictHostKeyChecking=no "$USR@$IP" 'ps -e | grep -i sshd | head -n1' 2>/dev/null | awk '{print $1}') “ :
Connecting to the server using the details the user gave and sending the command **“ps -e “** that display The list all processes that run on the device and cut the number of PID of sshd and display on screen to add inside the logfile. After the command being executed inserting the PID output of the ssh connection into a variable named "pid".

After the command being execute and saved into the variable there is a if statement to check if the pid variable is empty. The flag -z is to check if "pid" variable is empty if so move to 'then' statement and print Unknown when using the 'pid' variable in the log file.

Continue to the rest of the log function…

“ portnum=$(sshpass -p $PASS ssh -o StrictHostKeyChecking=no $USR@$IP "netstat -t |grep -i ESTABLISHED|grep -i ssh"). “

"netstat" command is used to provides information about network connections, routing tables, interface statistics and more.
Send netstat command to the device to display all the ESTABLISHED connections (netstat -t flag display information about TCP connections) then using grep command for ssh and using awk command to cut the random port of the connection to put in the logfile. after execute the command the all row of the random port is displayed, inserting it into a variable named "portnum"

“ PORTNUM=$(echo "$portnum" |awk '{print $5}'|awk -F: '{print $2}') “ :

echo the variable "portnum" that contains the all row of the random port after that using 'awk' command for the port number and using awk to grep the port number only without the **" : "** symbol. and insert it into a variable named "PORTNUM" to use it later in the log file.

“ cd ~ && cd Desktop “ :
Change the location to home directory and then entering Desktop directory.

“ mkdir Project.files “ :
("mkdir" command used for creating directories) making new directory inside Desktop named "Project.files"

“ cd ~ && cd Desktop && cd Project.files “ :
Change the location to home directory and then entering Desktop directory and entering the new created directory named "Project.files".

“ echo "$(date +'%b %e %H:%M:%S') sshd[$pid]: $accfail for $USR from $IPNIPE port $PORTNUM ssh2." >> NR.log “ :

Creating a command to display the text in the log file that contains all the variables created.

The command (date +'%b %e %H:%M:%S') will say what to display while using 'date':
(%b-format represents month name) (%e-format represents day)
(%H-format represents hour)(%M-format represents minute) (%S-format represents second).

making the structure of the log file and using all the necessary variables that was created before and using the " >> " to insert it into a new file named "NR.log"

Inside the LOG function there is a last function named SCANLOG :

```
function SCANLOG () # Function for making a structure of the log file the scaning domains and collecting data while using all the n
{
    cd ~ && cd Desktop && cd Project.files # Change the location to home directory and then entering Desktop then entering the dire
    echo "$(date +'%b %e  %H:%M:%S') sshd[$pid]: $IPNIPE port $PORTNUM scan $DMN using Nmap successfully." >> NR.log
    echo "$(date +'%b %e  %H:%M:%S') sshd[$pid]: Nmap data scan collected successfully inside $IPNIPE for: $DMN" >> NR.log
    echo "$(date +'%b %e  %H:%M:%S') sshd[$pid]: $IPNIPE port $PORTNUM scan $DMN using Whois successfully." >> NR.log
    echo "$(date +'%b %e  %H:%M:%S') sshd[$pid]: Whois data scan collected successfully inside $IPNIPE for: $DMN" >> NR.log
}
```

The function is for making a structure of the log file for the scanning domains and collecting data while using all the necessary variables inside the log. (the ">>" command used to adding text to a file without without overwriting its existing contents.)

Next function for scanning using nmap the user chosen country and creating a file named "Nmap.scan" and send all details of the nmap scan to the new file.
(this function will be execute only when being called when needed).

```
function NMAPSCAN () # Function for scanning using nmap the user chosen country and c
{
    cd ~ && cd Desktop && cd Project.files  # Change the location to home directory a
    echo " "
    echo "[+] Result Of Nmap Scan For: $DMN" >> Nmap.scan
    echo " " >> Nmap.scan
    nmap -p 22,21,80,443,25,3389,53 --open $DMN -Pn >> Nmap.scan # scanning the user
    echo " " >> Nmap.scan
    echo "************************************************************" >> Nmap.scan
}
```

(The " >> " command used to adding text to a file without overwriting its existing contents.)

“ cd ~ && cd Desktop && cd Project.files “ :
Change the location to home directory and then enter Desktop directory and entering the "Project.files" directory.

“ nmap -p 22,21,80,443,25,3389,53 --open $DMN -Pn >> Nmap.scan “ :
scanning the user chosen country (using -p to specify port number) scan for the 7 most known, when using "--open" its displaying only the open ports, and using "-Pn" to Skip host discovery and assumes that the target are online.

Function for scanning using whois the user chosen country and creating a file named "Whois.scan" and send all details of the nmap scan to the new file.
(this function will be execute only when being called when needed).

```
function WHOIS ()  # Function for scanning using whois the user chosen country and creat
{
    cd ~ && cd Desktop && cd Project.files  # Change the location to home directory and
    echo " " >> Whois.scan
    echo "[+] Result Of Whois Scan For: $DMN" >> Whois.scan
    echo " " >> Whois.scan
    whois $DMN >> Whois.scan  # scanning using whois command the user chosen country.
    echo " " >> Whois.scan
    echo "************************************************************" >> Whois.scan
}
```

(The " >> " command used to adding text to a file without overwriting its existing contents.)

“ cd ~ && cd Desktop && cd Project.files “ :
Change the location to home directory and then enter Desktop directory and entering the "Project.files" directory.

“ whois $DMN >> Whois.scan “ :
Scanning using whois command the user chosen country.

This next function Will ask the user for the ssh server details. Then creating a loop statement to check if the details are correct if not ask the user again for correct details, if the details are correct exit the loop and continue to display information about the ssh server.

```
function VPS() # This Function Will ask the user for the ssh server details. then creating a loop to check if the details are co

{
    echo "[*] To Preform Remote Control, You Will Need To Fill In The Following Details... "
    echo
    sleep 3
    # Creating a loop that ask the user for ssh server details then trying to connect until a successful SSH connection is estab
    while true
    do
        read -p " > Please Enter The IP Address Of The SSH Server: " IP  # reading the user input and asking him for the ip addr
        echo
        sleep 1
        read -p " > Please Enter The Username Of The SSH Server: " USR   # reading the user input and asking him for the usernam
        echo
        sleep 1
        read -p " > Please Enter The Password Of The SSH Server: " PASS  # reading the user input and asking him for the passwor
        echo
        echo
        sleep 1
        echo "[?] Please Wait While We Verify The Details Are Correct..."
        echo "               (Might take some time)"
        echo
        echo

        # command to attempt to connect to the ssh server using the credentials the user provided.
        sshpass -p "$PASS" ssh -o StrictHostKeyChecking=no -v "$USR"@"$IP" exit > /dev/null 2>&1 # "sshpass" used for entering s

        if [ $? -eq 0 ]; # is a conditional expression that checks whether the last command was successful. "$?"- holds the exit
        then
            LOG  # calling the LOG function to write the event into the log file
            break #  The break command is used to break out of the loop when the statemant are currect.
        else
            echo
            LOG > /dev/null 2>&1  # calling the LOG function to write the event into the log file
            echo "[!] One Of The Following Details Are Incorrect. Please try again."
            sleep 5
            clear
        fi
    done  # end of the loop.
```

There is a while statement in the function that creating a loop that ask the user for ssh server details then trying to connect until a successful SSH connection is established, in case is not correct going back to the start of the loop and ask the user to input the details again.

“ read -p " > Please Enter The IP Address Of The SSH Server: " IP “ :
reading the user input and asking him for the ip address of the ssh server, insert it into a variable named " IP ".

“ read -p " > Please Enter The Username Of The SSH Server: " USR “ :
reading the user input and asking him for the username of the ssh server, insert it into a variable named " USR ".

“ read -p " > Please Enter The Password Of The SSH Server: " PASS “ :
reading the user input and asking him for the password of the ssh server, insert it into a variable named " PASS ".

“ sshpass -p "$PASS" ssh -o StrictHostKeyChecking=no -v "$USR"@"$IP" exit “ :
"sshpass" used for entering ssh server password, "-o StrictHostKeyChecking=no" This option disables strict host key checking. "-v" flag enables verbose mode, providing more detailed information. " exit " for immediately exits the session after ssh server was checked.

Inside the while statement there is a if statement that is a conditional expression that checks whether the last command was successful. " $? " - holds the exit status of the last executed command. A value of 0 usually means success. so, if its equal to 0 then continue to 'then' statement if the connection was failed so continue to 'else' statement.
Inside the then statement the function LOG being called to write the event into the log file. also, the break command is used to break out of the loop when the statement are correct. And also in the else statement there is the LOG function that being called to write the event.

The end of the VPS function is :

```
    done  # end of the loop.
    sleep 3
    echo "[!] Your Details Are Correct. Progressing... "
    sleep 2
    echo
    echo
    echo "[*] Scaning The Device Of The SSH Server..."
    sleep 2.5
    echo
    TIME=$(sshpass -p $PASS ssh -o StrictHostKeyChecking=no $USR@$IP uptime) # connecting via ssh to the server using the credentials
    DVCIP=$(sshpass -p $PASS ssh -o StrictHostKeyChecking=no $USR@$IP 'hostname -I') # connecting via ssh to the server using the cred
    DVCNTRY=$(sshpass -p $PASS ssh -o StrictHostKeyChecking=no $USR@$IP "geoiplookup \$(curl -s ifconfig.co) | awk '{print \$5}'") # c
    echo " > The IP Address Of The Device: $DVCIP "
    echo
    sleep 2
    echo " > The Country Associated To The Device: $DVCNTRY"
    echo
    sleep 2
    echo " > Uptime Of The Device: $TIME"
    echo
    sleep 2
    echo "[!] The Scan Results Of The Device Are Over. "
    echo
    echo
    sleep 3

}
VPS
```

“ TIME=$(sshpass -p $PASS ssh -o StrictHostKeyChecking=no $USR@$IP uptime) “ :
connecting via ssh to the server using the credentials the user provided and send the command "uptime" ("uptime"-how long a system has been running and its current load). And inserting it into a variable named " TIME ".

“ DVCIP=$(sshpass -p $PASS ssh -o StrictHostKeyChecking=no $USR@$IP 'hostname -I') “ :
Connecting via ssh to the server using the credentials the user provided and send the command "hostname -I" ("hostname -I"- Display the ip of the device).
And inserting it into a variable named "DVCIP".

“ DVCNTRY=$(sshpass -p $PASS ssh -o StrictHostKeyChecking=no $USR@$IP "geoiplookup \$(curl -s ifconfig.co) | awk '{print \$5}'") “ :
Connect to the ssh server the running the command "curl -s ifconfig.co" that reveal the external ip address by curling into the website “ifconfig.co”, using "geoiplookup" command on the external ip to reveal the country, and using 'awk' to separate the country name only and finally inserting it into a variable named "DVCNTRY".

Function that create a loop to ask the user for Domain/IP, after runs check on the input to verify its exist and can be used, after the Domain/IP is verified continue to displaying information associated to the Domain/IP the user chosen.

```
function FUNDMN () # Function that create a loop to ask the user for Domain/IP and runs check on the input to verify its exist and can be used, after the Domai
{

    while true; # The start of the loop to check if the Domain/IP that the user typed are valid, exist and good to continue.
    do
        read -p " > Please Specify A Domain/IP Address To Scan: " DMN  # reading the user input and asking him to specify A Domain/IP to scan, and insert it in

        if [[ -z "$DMN" ]]; # " [[ ]]" we use double bracket for more complex conditional expressions. The flag -z is to check if "DMN" variable is empty if so
        then
            echo
            sleep 1
            echo "[!] Invalid Domain/IP: Empty Text."
            sleep 1
            echo
        elif [[ "$DMN" =~ ^[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+$ ]]; # 'elif' statment is alternative action if the previous conditions are not met. "[[ ]]" we use d
        then
            # Check if it's a valid IP address using dig -x
            if dig -x "$DMN" |grep -iq "ANSWER SECTION";  # This command performs a reverse DNS lookup on the ip to find domain to the ip address. and then gre
            then
                echo
                sleep 1
                echo "[*] IP Address Validation Successful. Progressing..."
                sleep 2
                break  # break the loop if the IP address is valid
            else
                echo
                sleep 1
                echo "[!] Invalid IP Address. Please Try Again."
                sleep 1
                echo
            fi
        else
            # In case its not ip and it's a domain the following output will be executed.
            if nslookup "$DMN" 2>/dev/null | grep -qi 'Non-'; # The nslookup command ask the the DNS to find an IP address that belong with this domain then us
            then
                echo
                sleep 1
                echo "[*] Domain Validation Successful. Progressing..."
                echo
                sleep 2
                break  # break the loop if the domain is valid
            else
                echo

                sleep 1
                echo "[!] Invalid Domain. Please Try Again..."
                sleep 1
                echo
            fi

        fi
    done  # End of the loop.
    echo
    echo
    echo "[?] Scaning The Domain/IP Address Might Take Few Seconds. Please Wait...  "
    SCANLOG  # Calling the function called "SCANLOG"
    NMAPSCAN  # Calling the function called "NMAPSCAN"
    WHOIS  # Calling the function called "WHOIS"
    sleep 3
    SCAN=$(sshpass -p $PASS ssh -o StrictHostKeyChecking=no $USR@$IP "nmap -p 22,21,80,443,25,3389,53 --open $DMN -Pn")  # Connecting via ssh to the server and
    ip=$(sshpass -p $PASS ssh -o StrictHostKeyChecking=no $USR@$IP "nmap -p 22,21,80,443,25,3389,53 --open $DMN -Pn"| grep -oP '(\d+\.){3}\d+' | head -n 1) # u
    LOCATION=$(sshpass -p $PASS ssh -o StrictHostKeyChecking=no $USR@$IP "geoiplookup $DMN "|awk -F, '{print $2}') # connecting to the ssh server and runing th
    echo
    echo " > The IP Address Of The Domain/IP: $ip"
    echo
    sleep 3
    echo " > The Country Associated To The Domain/IP: $LOCATION"
    echo
    sleep 3
    echo "Displaying Few Open Ports That Are Open In The Domain/IP: "
    echo
    echo "Port:     Service:"
    sshpass -p $PASS ssh -o StrictHostKeyChecking=no $USR@$IP "nmap -p 22,21,80,443,25,3389,53 --open $DMN -Pn"| grep open |awk '{print $1"    "$3}'  # connect
    sleep 4
    }
while true # Creating a loop for the user and ask him if he want to scan another domain/ip.
do
    FUNDMN  # Calling the function called "FUNDMN"

    echo
    echo "[!] The Scan Results Are Over. "
    echo
    sleep 1
    read -p " > Would You Like To Scan Another Domain/IP [y/n]: " yn  # after asking the user if he would like to scan another Domain/ip, if the answer is 'y'
    if [ "$yn" != "y" ];  # If the input that the user gave  equal to 'y' going to 'else' statment , if its not equal move to 'then' statment.
    then
        break  # breaking out of the loop if he dont want to scan another Domain/ip.
    else
        sleep 2
        clear
    fi
done
```

In the next page the explanation of the function.

There is a few statements inside the function.

First statement is while loop to check if the Domain/IP that the user typed are valid, exist and good to continue…

“ read -p " > Please Specify A Domain/IP Address To Scan: " DMN “ :
Reading the user input and asking him to specify A Domain/IP to scan, and insert it into a variable named "DMN".

After the while loop there is a if statement **“ if [[-z "$DMN"]] “ :**
" [[ ]]" we use double bracket for more complex conditional expressions. The flag -z is to check if "DMN" variable is empty if so, move to 'then' if it’s not empty move to ‘else' statment.

“ elif [["$DMN" =~ ^[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+$]] “ :
'elif' statment is alternative action if the previous conditions are not met. "[[ ]]" we use double bracket for more complex conditional expressions. " =~ " checking if the string of the left matches the pattern on the right side. the pattern after the symbol representing a valid IPv4 address. So, if the string matches the pattern move on to 'then' statment, if it doesn't continue to 'else' statment.

Inside the “ then “ statement there is an another if statement :
“ if dig -x "$DMN" |grep -iq "ANSWER SECTION" “ :
check if it's a valid IP address using dig -x
This command performs a reverse DNS lookup on the ip to find domain to the ip address. and then using grep the word **"ANSWER SECTION”** that means the ip is belong to a domain and valid ip address. If its valid moving to 'then' statment, inside the then statement there is a **break** command that break the loop if the IP address is valid. But if it didn't grep the word mean no domain belong to the ip and its invalid ip address then moving to 'else' statment.

Inside the “ else statement there is an another if statement :
In case it’s not ip and it's a domain the following output will be executed.
“ if nslookup "$DMN" 2>/dev/null | grep -qi 'Non-' “ :
The nslookup command ask the DNS to find an IP address that belong with this domain then using grep (flag -qi mean in quite mode and Ignore case) for the word "Non-" mean its valid domain. If there is a match mean its vaild and moving to 'then' statment, inside the then statement there is a **break** command that break the loop if the Domain is valid.
But if not matched it’s means that the Domain is invalid and move on to 'else' statment.

The continuation of the function is:

```
    echo
    echo
    echo "[?] Scaning The Domain/IP Address Might Take Few Seconds. Please Wait...  "
    SCANLOG  # Calling the function called "SCANLOG"
    NMAPSCAN  # Calling the function called "NMAPSCAN"
    WHOIS  # Calling the function called "WHOIS"
    sleep 3
    SCAN=$(sshpass -p $PASS ssh -o StrictHostKeyChecking=no $USR@$IP "nmap -p 22,21,80,443,25,3389,53 --open $DMN -Pn")  # Connecting via ssh to the server
    ip=$(sshpass -p $PASS ssh -o StrictHostKeyChecking=no $USR@$IP "nmap -p 22,21,80,443,25,3389,53 --open $DMN -Pn"| grep -oP '(\d+\.){3}\d+' | head -n 1)
    LOCATION=$(sshpass -p $PASS ssh -o StrictHostKeyChecking=no $USR@$IP "geoiplookup $DMN "|awk -F, '{print $2}') # connecting to the ssh server and runing
    echo
    echo " > The IP Address Of The Domain/IP: $ip"
    echo
    sleep 3
    echo " > The Country Associated To The Domain/IP: $LOCATION"
    echo
    sleep 3
    echo "Displaying Few Open Ports That Are Open In The Domain/IP: "
    echo
    echo "Port:     Service:"
    sshpass -p $PASS ssh -o StrictHostKeyChecking=no $USR@$IP "nmap -p 22,21,80,443,25,3389,53 --open $DMN -Pn"| grep open |awk '{print $1"    "$3}'  # conn
    sleep 4
    }
while true # Creating a loop for the user and ask him if he want to scan another domain/ip.
do
    FUNDMN  # Calling the function called "FUNDMN"

    echo
    echo "[!] The Scan Results Are Over. "
    echo
    sleep 1
    read -p " > Would You Like To Scan Another Domain/IP [y/n]: " yn  # after asking the user if he would like to scan another Domain/ip, if the answer is '
    if [ "$yn" != "y" ];  # If the input that the user gave  equal to 'y' going to 'else' statment , if its not equal move to 'then' statment.
    then
        break  # breaking out of the loop if he dont want to scan another Domain/ip.
    else
        sleep 2
        clear
    fi
done
```

SCANLOG : Calling the function to insert the Domain to the log file.
NMAPSCAN : Calling the function to insert the Domain scan of nmap to a file.
WHOIS : Calling the function to insert the Domain scan of whois to a file.

“ SCAN=$(sshpass -p $PASS ssh -o StrictHostKeyChecking=no $USR@$IP "nmap -p 22,21,80,443,25,3389,53 --open $DMN -Pn") “ :
Connecting via ssh to the server and scanning using nmap the user chosen Domain/IP (using -p to specify port number) scan for the 7 most known, when using "--open" its displaying only the open ports, and using "-Pn" to Skip host discovery and assumes that the target are online. and inserting it into a variable named "SCAN".

“ ip=$(sshpass -p $PASS ssh -o StrictHostKeyChecking=no $USR@$IP "nmap -p 22,21,80,443,25,3389,53 --open $DMN -Pn"| grep -oP '(\d+\.){3}\d+' | head -n 1) “ :
Using grep -oP to extract and print only the portions of the text that match the specified regular expression pattern, the pattern after extracts the structure of an IPv4 address from the nmap output. The command “head -n1” make sure to cut the first ip that appears. and inserting it into a variable named "ip".

“ LOCATION=$(sshpass -p $PASS ssh -o StrictHostKeyChecking=no $USR@$IP "geoiplookup $DMN "|awk -F, '{print $2}') “ :
Connecting to the ssh server and running the 'geoiplookup' on the Domain/ip the user choose to reveal its associated country. and inserting it into a variable named "LOCATION".

“ sshpass -p $PASS ssh -o StrictHostKeyChecking=no $USR@$IP "nmap -p 22,21,80,443,25,3389,53 --open $DMN -Pn"| grep open |awk '{print $1"   "$3}' “ :
Connecting to the ssh server executing nmap scan for the according port numbers and grep the word open to display only the open ports, the using awk to print only the port number and service.

The last part of the function is an another while loop that creating a loop for the user and ask him if he want to scan another domain/ip:

Inside the loop the FUNDMN function being called to start the function :

“ read -p " > Would You Like To Scan Another Domain/IP [y/n]: " yn “ :
After asking the user if he would like to scan another Domain/ip, if the answer is 'y' moving to 'else' statment inside the if statement, if it’s not equal to 'y' then moving to 'then' inside the if statment that scan another Domain/IP.

“ if ["$yn" != "y"] “ :
The syntax “ != “ means not equal .So, If the input that the user gave equal to 'y' going to 'else' statment , if its not equal move to 'then' statement , inside the then statement there is a **break** command that break the loop.

Function for the end of the script which concludes the script and writing the files locations of the different scans that been executed in the script and the log file location.

```
function END()  # Function for the end of the script which concludes the script and writing the files locations of the different scans that been executed in the
){
clear
figlet -t -c -f slant This is The End Of The Script! |lolcat # using the command figlet that make the text font diffrent and convert it into ASCII, -t flag to
echo
echo
sleep 0.5
echo "               > [!]  All The Nmap Scans That You Scan On The Domain/IP Are Saved In Your Desktop -> ~/Desktop/Project.files/Nmap.scan " |lolcat -a  # the
echo
echo "               > [!]  All The Whois Scans That You Scan On The Domain/IP Are Saved In Your Desktop -> ~/Desktop/Project.files/Whois.scan " |lolcat -a  # t
echo
echo "               > [!]  Your Scans, Data And Failed/Success Connections To SSH Was Saved Into A Log file -> ~/Desktop/Project.files/NR.log "|lolcat -a  # th
echo
echo
echo
echo
echo -e "                                                  Press Any Key To Exit..." | lolcat -a  #using -e enables the interpretation of backslash ,and
read -n 1 -s -r # This command will exit the script after receiving any key press. "-n 1" flag specifies that only one character should be read. -s  Causes the
cmatrix & sleep 4; kill $! # the command cmatrix make visually dynamic and animated display of characters on the terminal screen. using sleep command to make it
echo
}
END
```

“ figlet -t -c -f slant This is The End Of The Script! |lolcat “ :
Using the command figlet that make the text font diffrent and convert it into ASCII, -t flag to spread on the screen, -c to center the text in the terminal, -f to Specifies the font in the case is 'slant' font. and the " | lolcat -a " to make the text colorful.

“ echo -e “Press Any Key To Exit..." | lolcat -a “ :
using echo command with “-e “ flag enables the interpretation of backslash ,and the "| lolcat -a " command to make the text colorful.

“ read -n 1 -s -r “ :
This command will exit the function after receiving any key press. " -n 1" flag specifies that only one character should be, “read. -s” Causes the input to be silent and not displaying on screen, “ -r “ flag treating backslashes and escape characters as literal characters.

“ cmatrix & sleep 4; kill $! “ :
The command cmatrix make visually dynamic and animated display of characters on the terminal screen. Using sleep command to make it run for 4 seconds, and the "kill $!" command sends a signal to kill the most recently backgrounded process. The '$!' represents the process ID (PID) of the last background command.

And for the end there is a case statement, the script using case statment to display the files the user choose inside the variable "key".

```
case $key in  # Using case statment to display the file the user choose inside the variable "key".
    1)
        echo
        echo
        echo "[*] Displaying The Log File... "
        echo
        echo
        sleep 2.5
        cat ~/Desktop/Project.files/NR.log
        echo
        echo
        ;;
    2)
        echo
        echo
        echo "[*] Displaying The Whois Scan File... "
        echo
        echo
        sleep 2.5
        cat ~/Desktop/Project.files/Whois.scan
        echo
        echo
        ;;
    3)
        echo
        echo
        echo "[*] Displaying The Nmap Scan File... "
        echo
        echo
        sleep 2.5
        cat ~/Desktop/Project.files/Nmap.scan
        echo
        echo
        ;;
    *)
        echo
        echo "Exiting The Script... Goodbye!"
        exit 0 #exiting the scipt
        ;;
esac
```

Displaying case options to display specific file, if the number of case was chosen using cat to display the file or any other option to exit the script.

<u>“ read -p " > Press Any Key To Go Back To The Menu, Or Type 'exit' To Exit: " “</u> : after the user chosen a case asking the user if he want to exit by typing 'exit' or pressing any key to go make to the menu.

Then creating a if statement that if the replay of the last command equal to 'exit' moving to 'then' statment and exiting the script.

```
      read -p " > Press Any Key To Go Back To The Menu, Or Type 'exit' To Exit: "
    # Check if the user typed "exit" and exit the script
    if [[ "$REPLY" == "exit" ]];  #if the replay of the last command equal to 'exi
    then
        echo
        echo "   Exiting The Script... Goodbye!"
        sleep 3
        exit 0  # exiting the script.
    fi
done  # end of the loop.
```

If the user choose any other input and didn’t typed ‘exit’ going back to the case menu.

This Is The End Of The Script…

(Enjoy..:)