Network Research Project

TMagen773637.S11.NX201

Ron Nisinboym

July 8, 2025

Overview

Project Description:

This project aims to automate the process of network research by connecting to remote Linux machines while ensuring anonymity. It involves gathering key network-related information such as Nmap scans, Whois data, and cURL responses. The entire connection process should be anonymized using the Nipe tool to route all traffic through the Tor network, providing a layer of privacy and security before performing the necessary network investigation tasks over SSH.

Technologies Used

- **Bash Scripting**: Used for automating the execution of commands, managing file paths, and automating the report generation.
- **Nipe**: A Tor-based anonymity tool used to ensure traffic is routed anonymously.
- **TOR:** Routes internet traffic through a series of different nodes, to anonymize the communication.
- **cURL**: For interacting with web servers and retrieving HTTP response data such as external IP.
- **GeoIPLookup:** Used to look up the geographical location(Country) based on the IP.
- **CPANMinus:** A more script-friendly alternative to CPAN, for installing Perl modules.
- **SSH**: The secure shell protocol for establishing remote connections with the target machines.
- **SCP**: A secure file transfer protocol used to transfer collected files between two machines over SSH.
- **Nmap**: For performing network scans and identifying open ports and services.
- **Whois**: For domain registration information retrieval.
- **Uptime**: To display how long the system has been running since its last reboot.

Detailed Workflow

1. **Tools installation:** The script starts by checking for root user, creating an empty folder for the script to save data into. It then checks for the installation of **Nipe** and required dependencies, installing any missing components to run this script.
2. **Anonymity:** Nipe is started to ensure anonymity by routing traffic through Tor. It then checks the anonymity status of the local machine. If anonymous, it proceeds.
3. **SSH:** The user is prompted for SSH credentials to connect to the remote machine.Once connected, the script tries to install Nmap, Whois, and cURL tools before running them as intended on the remote system.
4. **Data Collection:** The collected data is saved locally on the remote machine, then transferred via SCP to the local machine. When transferred, the collected data, previously saved on the remote machine, gets deleted.
5. **Log File Generation:** On the local machine, generates a log file that consolidates all the data collected by combining the output of different scans and any other relevant data into a single log file.
6. **Disable Anonymity:** Stops the **Nipe** service to terminate the Tor connection and return the local machine to its original, non-anonymous state, before exiting the script.

Design Principles

1. **Simplicity Over Decoration:** By reducing the use of bold colors, fancy fonts, and overly complex layouts, the script avoids visual clutter, providing an **"easy on the eyes" experience**.
2. **Clear, Readable Output**:
    a. The font choice is kept **simple and legible**, ensuring that the user can quickly read and interpret the different output, even over long periods of use.
    b. Colors are used sparingly, with only essential information (e.g. important information, warnings, errors and successes) highlighted using subtle color changes. This reduces eye strain while still providing the necessary emphasis on important results.

Credits

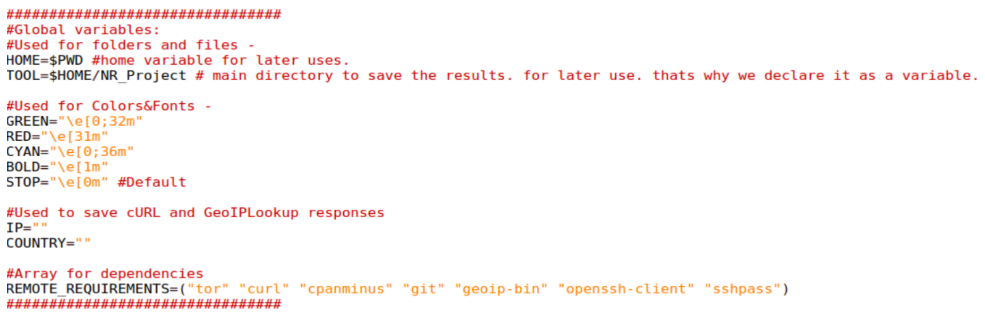
1. **Erel Regev (lecturer) –** for kickstarting the bash script, providing the initial framework and guidance by supplying the foundational script structure and key concepts which was later modified and expanded to my own vision of this project.
2. **ChatGPT** – The use of AI was made exclusively as a helping tool. Used to fix some of the errors and bugs occurring while writing the script, consulting in regards of the script's logic when more difficult steps were used(e.g. Installing tools on the remote host, SSH options), asking for general information(e.g. “Heredoc”, “”Press any key” logic, Arrays), and things I struggled to remember(e.g. certain flags). The choice of ChatGPT over Google, forums or other sources was made mostly to save time and get straight forward explanations for each piece of advice to keep learning from it.

Detailed Walkthrough Of The Script

This section provides a breakdown of the script’s functionality, including detailed explanations of key code snippets, along with screenshots of the code and its practical execution.

Global Variables -

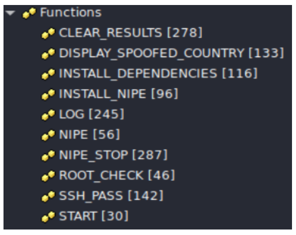
In this screenshot we can see all the global variables that are being used by the script.
These variables **are not** nested in any function, to be available for any function that needs to call them.



```
################################
#Global variables:
#Used for folders and files -
HOME=$PWD #home variable for later uses.
TOOL=$HOME/NR_Project # main directory to save the results. for later use. thats why we declare it as a variable.

#Used for Colors&Fonts -
GREEN="\e[0;32m"
RED="\e[31m"
CYAN="\e[0;36m"
BOLD="\e[1m"
STOP="\e[0m" #Default

#Used to save cURL and GeoIPLookup responses
IP=""
COUNTRY=""

#Array for dependencies
REMOTE_REQUIREMENTS=("tor" "curl" "cpanminus" "git" "geoip-bin" "openssh-client" "sshpass")
################################
```

Functions in general -

In this screenshot we can see a list of all functions that are being used by the script.
The idea is to encapsulate repetitive chunks of code or any code that has a unique behavior, to achieve ease of access and further modifying of specific functionality.
Therefore, all pieces of code in this script, **except** global variables, are nested inside functions.



```
Functions
    CLEAR_RESULTS [278]
    DISPLAY_SPOOFED_COUNTRY [133]
    INSTALL_DEPENDENCIES [116]
    INSTALL_NIPE [96]
    LOG [245]
    NIPE [56]
    NIPE_STOP [287]
    ROOT_CHECK [46]
    SSH_PASS [142]
    START [30]
```

Starting the script -

This function starts the script once the user tries to run the ".sh" file, immediately calling the "ROOT_CHECK" function to prevent any user who is not root from continuing to run it.

It uses the $TOOL variable to create the parent folder which will store Nipe and different results.

The use of the –p flag with the "mkdir" command, used in different parts of the script to avoid errors if a folder already exists.

It then calls the "CLEAR_RESULTS" and "NIPE" functions to continue with the script.

```
#function that start the script
function START() {
    ROOT_CHECK
    figlet -f small "Network Research"
    mkdir -p $TOOL

    CLEAR_RESULTS

    NIPE
}
```

Checking for root user -

This function uses an *"if"* statement to check if the current user is a root user.

It is done to prevent a user who is not root to continue running the script, and to have elevated permissions for certain actions taking place(e.g. creation of folders and files).

```
#function that checks if user is root
function ROOT_CHECK(){
    USER=$(whoami)
    if [ "$USER" != "root" ] #if NOT root.
    then
        echo "You are not root. Exiting.."
        exit
    fi
}
```

Clearing previous results -

This function uses an *"if"* statement with the "*-d*" flag, to check if the folder for the result files exists.
If so, it uses *"rm -rf"* along with a wildcard character inside that folder to remove all previous gathered result files.

Utilizing the *"read -n*" command, the user gets the option to exit the script if needed, prior certain tasks will take place. **Notice! -** This method slows the execution of this script by requesting an additional output each time it runs if the OUTPUT_FILES folder already exists, consider marking it out in the code to disable it.

The use of the –e flag with the "echo" command, used in many parts of the script to be able to use different color\font variables, and new-line logic as intended.

```
#function that clears previous OUTPUT_RESULTS (doesn't include LOG file)
function CLEAR_RESULTS (){
   if [ -d "$TOOL/OUTPUT_FILES" ]; then #If the OUTPUT_FILES directory exists
        echo -e "\n${CYAN}***${BOLD}ATTENTION!${STOP} ${CYAN}Files inside${STOP} $TOOL/OUTPUT_FILES ${CYAN}will be removed. ${BOLD}Consider a backup!!!***${STOP}\n"
        echo -e "Use CTRL+C to exit the script, or press any key to continue... "
        read -n 1 -s #Waits for 1 character as user input to proceed. silents the input.
        rm -rf "$TOOL/OUTPUT_FILES/*"  #Deletes all files inside OUTPUT_FILES
        echo -e "${CYAN}\nOUTPUT_FILES folder has been cleared!${STOP}\n"
    else
        echo -e "${CYAN}Creating OUTPUT_FILES directory for the results in:${STOP}" $TOOL
        echo -e "${CYAN}To initiate dependecies installation, remove nipe folder if already exists in this folder.${STOP}\n"
        mkdir -p $TOOL/OUTPUT_FILES #Create folder if doesn't exist
    fi
}
```

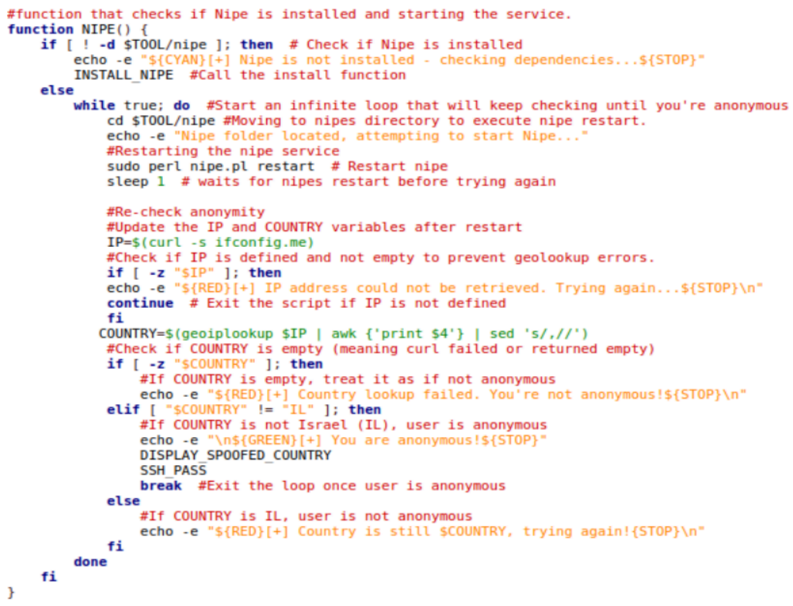
Attempt to start Nipe -

This Function checks if Nipe is installed in the correct folder. If not, it calls the “INSTALL_NIPE” function.

It then uses a “*While True*” loop, to continually check whether the user’s public IP address and visible location is no longer located in Israel, to achieve anonymity, and to restart it if it isn’t achieved.
The reason for this loop is that the Nipe tool has difficulties to start as intended on the first run.
In this loop, different *“if”* statements are being used to prevent the function from continuing in case that important variables have returned empty.

“GeoIPLookup” is being used with some text-manipulation to get the country code, based on the “IP” variable that gets defined earlier. As it is still being run on the local machine, “GeoIPLookup” is used at this stage, instead of “cURL”, for geographical information.

It also calls the “DISPLAY_SPOOFED_COUNTRY” and “SSH_PASS” functions when anonymity has been achieved.

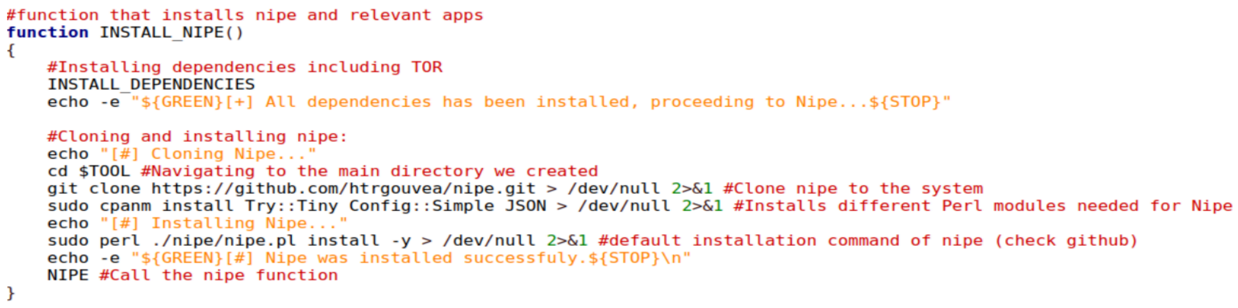


```
#function that checks if Nipe is installed and starting the service.
function NIPE() {
    if [ ! -d $TOOL/nipe ]; then  # Check if Nipe is installed
        echo -e "${CYAN}[+] Nipe is not installed - checking dependencies...${STOP}"
        INSTALL_NIPE  #Call the install function
    else
        while true; do  #Start an infinite loop that will keep checking until you're anonymous
            cd $TOOL/nipe #Moving to nipes directory to execute nipe restart.
            echo -e "Nipe folder located, attempting to start Nipe..."
            #Restarting the nipe service
            sudo perl nipe.pl restart  # Restart nipe
            sleep 1  # waits for nipes restart before trying again

            #Re-check anonymity
            #Update the IP and COUNTRY variables after restart
            IP=$(curl -s ifconfig.me)
            #Check if IP is defined and not empty to prevent geolookup errors.
            if [ -z "$IP" ]; then
            echo -e "${RED}[+] IP address could not be retrieved. Trying again...${STOP}\n"
            continue  # Exit the script if IP is not defined
            fi
           COUNTRY=$(geoiplookup $IP | awk {'print $4'} | sed 's/,//')
            #Check if COUNTRY is empty (meaning curl failed or returned empty)
            if [ -z "$COUNTRY" ]; then
                #If COUNTRY is empty, treat it as if not anonymous
                echo -e "${RED}[+] Country lookup failed. You're not anonymous!${STOP}\n"
            elif [ "$COUNTRY" != "IL" ]; then
                #If COUNTRY is not Israel (IL), user is anonymous
                echo -e "\n${GREEN}[+] You are anonymous!${STOP}"
                DISPLAY_SPOOFED_COUNTRY
                SSH_PASS
                break  #Exit the loop once user is anonymous
            else
                #If COUNTRY is IL, user is not anonymous
                echo -e "${RED}[+] Country is still $COUNTRY, trying again!{STOP}\n"
            fi
        done
    fi
}
```

Installing Nipe -

This function downloads a git package to later install Nipe, after installing Perl modules along with other dependencies, that are being called by the “INSTALL_DEPENDENCIES” function.

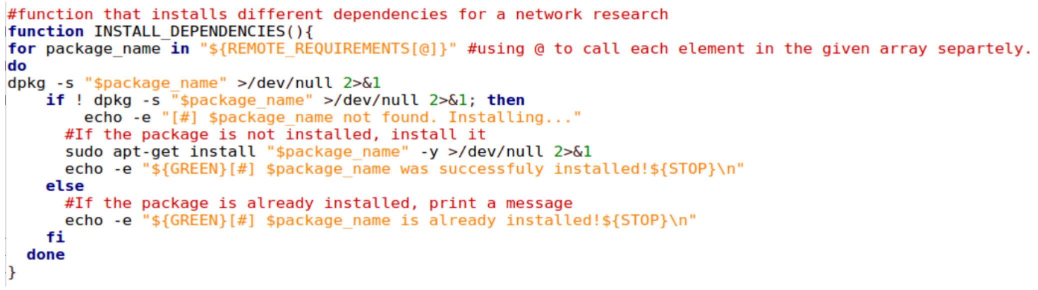
“cpanm” is being used instead of *“cpan”*, to automate the process by skipping the user's, otherwise required, input.



```
#function that installs nipe and relevant apps
function INSTALL_NIPE()
{
    #Installing dependencies including TOR
    INSTALL_DEPENDENCIES
    echo -e "${GREEN}[+] All dependencies has been installed, proceeding to Nipe...${STOP}"

    #Cloning and installing nipe:
    echo "[#] Cloning Nipe..."
    cd $TOOL #Navigating to the main directory we created
    git clone https://github.com/htrgouvea/nipe.git > /dev/null 2>&1 #Clone nipe to the system
    sudo cpanm install Try::Tiny Config::Simple JSON > /dev/null 2>&1 #Installs different Perl modules needed for Nipe
    echo "[#] Installing Nipe..."
    sudo perl ./nipe/nipe.pl install -y > /dev/null 2>&1 #default installation command of nipe (check github)
    echo -e "${GREEN}[#] Nipe was installed successfuly.${STOP}\n"
    NIPE #Call the nipe function
}
```

Other Nipe Dependencies installation -

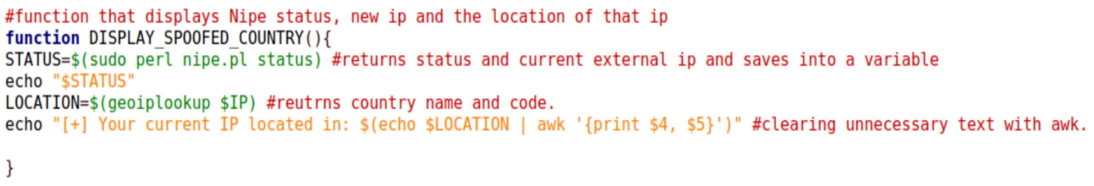
This function uses a *"For Loop"* with a global variable of an array which contains names of different tools that are required for Nipe and for executing different parts of this script (e.g. *"TOR"*, *"cURL"*, *"sshpass"*)



```
#function that installs different dependencies for a network research
function INSTALL_DEPENDENCIES(){
for package_name in "${REMOTE_REQUIREMENTS[@]}" #using @ to call each element in the given array separtely.
do
dpkg -s "$package_name" >/dev/null 2>&1
    if ! dpkg -s "$package_name" >/dev/null 2>&1; then
        echo -e "[#] $package_name not found. Installing..."
      #If the package is not installed, install it
      sudo apt-get install "$package_name" -y >/dev/null 2>&1
      echo -e "${GREEN}[#] $package_name was successfuly installed!${STOP}\n"
    else
      #If the package is already installed, print a message
      echo -e "${GREEN}[#] $package_name is already installed!${STOP}\n"
    fi
  done
}
```

Showing user's public IP address and corresponding geolocation -

This function is using Nipe to save its "status" output into a variable which is then used with "GeoIPLookup" along with some text-manipulation to provide a clear and informative output to the user.



```
#function that displays Nipe status, new ip and the location of that ip
function DISPLAY_SPOOFED_COUNTRY(){
STATUS=$(sudo perl nipe.pl status) #returns status and current external ip and saves into a variable
echo "$STATUS"
LOCATION=$(geoiplookup $IP) #reutrns country name and code.
echo "[+] Your current IP located in: $(echo $LOCATION | awk '{print $4, $5}')" #clearing unnecessary text with awk.

}
```

Connecting to a remote host and running tools -

This function uses "*sshpass"* with credentials that are defined by the user's input and saved as variables, including *"TARGET_ADDRESS"* which holds a full *USER@IP* SSH address.
It uses different SSH options such as:

- **StrictHostKeyChecking=no -** Disables host key verification, which in turn disables a prompt when connecting for the first time into a remote host.
- **UserKnownHostsFile=/dev/null -** Prevents storing the key in /.ssh/known_hosts - to avoid host key mismatch warnings.
- **IdentitiesOnly yes -** Makes sure that SSH uses only the specified method(sshpass), to prevent the trying of other keys that might be loaded.
- **LogLevel=QUIET -** This suppresses SSH-related output. The option controls the verbosity of SSH client logs. Setting it to "QUIET" reduces some unnecessary messages.

Collectively, these options bypass common SSH security checks and warnings for an automated login and makes the script less dependent on user input and reduces text clutter made by warnings.
The reason for choosing this practice was to avoid different failures that occurred during testing the script at the *sshpass* connection stage, on different Linux systems(e.g. "Remote host identification has changed!")
The **disadvantage** of this method is that it is less secure, making it easier to impersonate the remote host for an attacker, and weakening the authentication, as SSH keys are considered safer than a password, but this **shouldn't** be a problem in our case.

To try to install the research tools on the remote host, this function takes advantage of this - "*echo \$ssh_pass | sudo -S*". It is done to pass the password that was saved in that variable previously, to use *sudo* without additional user input.

When a connection is established, it proceeds to use a "heredoc"(here document) - *"<<SSH_COMMANDS"* to pass multiple lines of commands, followed by an *"if"* statement along with *"$?"* to check if the connection was indeed established.
Some variables get defined in the "heredoc" as it is unable to reach the global variables of this script.

Uses *"curl -s "http://ip-api.com/json/$REMOTE_EXTERNAL_IP?fields=countryCode""* at that stage to fetch country code, to avoid installing an additional tool(GeoIPLookup) on the remote host, while also getting more accurate data via the API.

It also uses *"SCP"* to collect the generated result files from the remote host prior to deleting them from it.
Finally, it calls the "LOG" function.

```
#function that connects user to a remote host via SSH and runs different commands
function SSH_PASS (){
echo ""
echo -e "${CYAN}PAY ATTENTION! Proceeding with this stage will execute the following steps using SSH:\n [#]Attempt connection to remote host\n [#]Attempt installing research tools if missing\n [#]Attempt to gather information${STOP}\n"
read -p "Please type the username for the remote server:" ssh_user
read -s -p "Please provide the password for the user of the remote server:" ssh_pass
echo ""
read -p "Please provide the IP address of the remote server:" ssh_ip
TARGET_ADDRESS="$ssh_user"@"$ssh_ip"


#SSH into the remote server and execute commands:
#When connected to SSH, automatically try to install nmap\whois\curl on the remote host and hide output
#using different options to bypass secuirty checks
#Using -t to be able to run sudo
sshpass -p "$ssh_pass" ssh -o "StrictHostKeyChecking=no" -o "UserKnownHostsFile=/dev/null" -o "IdentitiesOnly yes" -o "LogLevel=QUIET" -t "$TARGET_ADDRESS" "export ssh_pass='$ssh_pass'; echo \$ssh_pass | sudo -S apt-get install -y nmap whois curl > /dev/null 2>&1"
#Connects to SSH and starts a heredoc for running commands
sshpass -p "$ssh_pass" ssh -o "StrictHostKeyChecking=no" -o "UserKnownHostsFile=/dev/null" -o "IdentitiesOnly yes" -o "LogLevel=QUIET" "$TARGET_ADDRESS" << 'SSH_COMMANDS'
#Global Variables for sshpass:
GREEN="\e[0;32m" # color code for green
STOP="\e[0m" # color code to end the "painting".
REMOTE_LOCAL_IP=$(hostname -I | awk '{print $1}')
REMOTE_EXTERNAL_IP=$(curl -s https://ifconfig.me)

#Creating output folder on remote host(will be deleted after copied to local machine)
mkdir -p /home/$USER/Desktop/OUTPUT_FILES

#Variables for each output file:
CURL_FILE="/home/$USER/Desktop/OUTPUT_FILES/curl_output.txt"
UPTIME_FILE="/home/$USER/Desktop/OUTPUT_FILES/uptime_output.txt"
COUNTRY_FILE="/home/$USER/Desktop/OUTPUT_FILES/country_output.txt"
WHOIS_FILE="/home/$USER/Desktop/OUTPUT_FILES/whois_output.txt"
NMAP_FILE="/home/$USER/Desktop/OUTPUT_FILES/nmap_output.txt"


#Successfully connected message
echo ""
echo -e "${GREEN}Successfully connected! Gathering information...${STOP}"

#Gather info:

#1 Fetch the local IP address of the server and save it to a file
echo -e "\n# Remote Server Local IP" | tee -a $CURL_FILE
echo $REMOTE_LOCAL_IP | tee -a $CURL_FILE

#2 Fetch the External IP address of the server and save it to a file
echo -e "\n# Remote Server External IP" | tee -a $CURL_FILE
echo $REMOTE_EXTERNAL_IP | tee -a $CURL_FILE

#3 Get country information based on the remote server's external IP and save it to its own file
COUNTRY=$(curl -s "http://ip-api.com/json/$REMOTE_EXTERNAL_IP?fields=countryCode" | awk -F'"countryCode":"' '{print $2}' | awk -F'"' '{print $1}')
echo -e "\n# Remote Server Country" | tee $COUNTRY_FILE
echo $COUNTRY | tee -a $COUNTRY_FILE

#4 Get uptime of the remote server and save it to its own file
UPTIME=$(uptime -p)
echo -e "\n# Remote Server Uptime" | tee $UPTIME_FILE
echo $UPTIME | tee -a $UPTIME_FILE

#5 Perform a Whois lookup using the local IP and save it to its own file
echo -e "\n# Whois lookup for local IP $REMOTE_EXTERNAL_IP" >> $WHOIS_FILE
whois $REMOTE_EXTERNAL_IP >> $WHOIS_FILE 2>&1

#6 Scan open ports on the local IP of the server and save the result to its own file
echo -e "\n# Scanning open ports on the remote server ($REMOTE_LOCAL_IP)" >> $NMAP_FILE
nmap $REMOTE_LOCAL_IP >> $NMAP_FILE 2>&1

SSH_COMMANDS

#Checks If SSH connection was successful based on the resault of the last command before the "heredoc" starts.
if [ $? -eq 0 ]; then

#Notifying user prior to copying the files via scp.
echo ""
echo "[+] Copying the output files to your local machine..."

#Copying the files from the remote server to local machine with scp. Using -r to recursively copy all the files.
sshpass -p "$ssh_pass" scp -o "StrictHostKeyChecking=no" -o "UserKnownHostsFile=/dev/null" -o "IdentitiesOnly yes" -o "LogLevel=QUIET" -r "$TARGET_ADDRESS":/home/"$ssh_user"/Desktop/OUTPUT_FILES "$TOOL/"

#Deleting the files from the remote machine.
sshpass -p "$ssh_pass" ssh -o "StrictHostKeyChecking=no" -o "UserKnownHostsFile=/dev/null" -o "IdentitiesOnly yes" -o "LogLevel=QUIET" "$TARGET_ADDRESS" "rm -rf /home/$ssh_user/Desktop/OUTPUT_FILES"



#Showing messages to indicate files are saved
echo -e "\n${GREEN}[+] All information saved to individual files on the local machine:"
echo -e "[+] curl info saved to $TOOL/OUTPUT_FILES/curl_output.txt"
echo -e "[+] country info saved to $TOOL/OUTPUT_FILES/country_output.txt"
echo -e "[+] uptime results saved to $TOOL/OUTPUT_FILES/uptime_output.txt"
echo -e "[+] whois results saved to $TOOL/OUTPUT_FILES/whois_output.txt"
echo -e "[+] nmap results saved to $TOOL/OUTPUT_FILES/nmap_output.txt${STOP}"

#Calling the LOG function
LOG

else
#If the SSH connection failed, display a failure message.
echo -e "${RED}\nSSH connection failed! Please check your credentials or server status.${STOP}"
fi

}
```

Creating a log file -

This function uses a timestamp embedded into the filename to make its title distinguished if new log files are created later, to prevent overwriting older logs and to make it easier for the user to find a specific log for further investigation.

It also uses a local variable with the *"basename"* and *"sed"* command to be able to "catch" each results filename, and add it to its corresponding position in the log file, while removing the path to it and the ".txt" extension. Done for aesthetic reasons, making the reading of the log easier.

Proceeds with calling the "NIPE_STOP" function.

```
#function that makes a single file that summarizes all "output" files information.
function LOG(){
    #Defining log file name and location + adding timestamp.
    AUDIT_LOG="$TOOL/audit_log_$(date +%F_%H-%M-%S).log"

    #Array of output files on the local machine
    OUTPUT_FILES=(
        "$TOOL/OUTPUT_FILES/curl_output.txt"
        "$TOOL/OUTPUT_FILES/country_output.txt"
        "$TOOL/OUTPUT_FILES/uptime_output.txt"
        "$TOOL/OUTPUT_FILES/whois_output.txt"
        "$TOOL/OUTPUT_FILES/nmap_output.txt"
    )
    #Message that starts the log file
    echo "Audit log created on: $(date)" > "$AUDIT_LOG"

    #Loop through each output file and write to the log if it exists
    for FILE in "${OUTPUT_FILES[@]}"; do
        if [ -f "$FILE" ]; then
            #Extract just the filename from the full path and delete .txt extention.
            FILENAME=$(basename "$FILE" | sed 's/.txt//')
            #Write the file's output to the log
            echo -e "\n========== $FILENAME ==========\n" >> "$AUDIT_LOG"
            cat "$FILE" >> "$AUDIT_LOG"
        fi
    done

    #Completion message in terminal
    echo -e "\n${GREEN}Log file generation has been completed, and saved to $AUDIT_LOG${STOP}\n"

    NIPE_STOP
}
```

Back to real external IP address and location -

This function updates the relevant global variables before they are being used in a *"if"* statement to check and confirm that the user's public IP address, and in turn the geolocation, has been restored back to default - "IL" in this case prior to exiting the script.

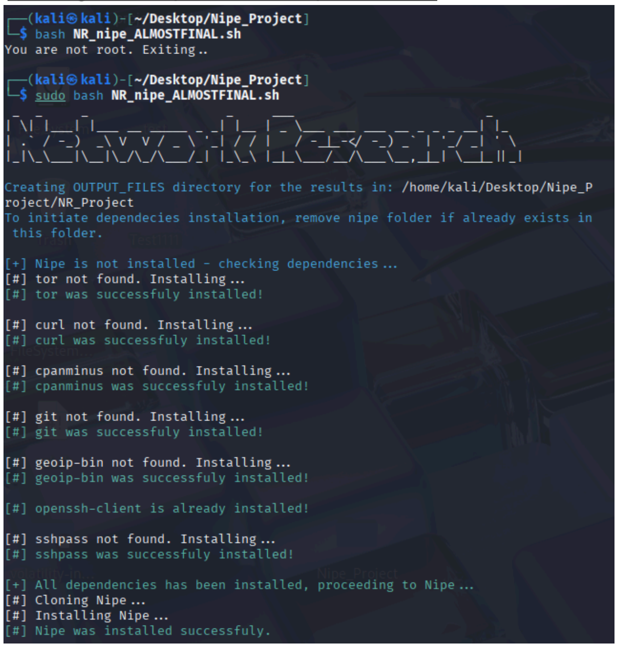
It calls the "DISPLAY_SPOOFED_COUNTRY" function to show the user Nipe's status with their current IP address and geolocation after Nipe has been allegedly stopped.

```
#function that stops nipe and switches the ip back to normal.
function NIPE_STOP(){
echo "Stopping Nipe..."
#Moving to nipe folder to execute nipe.pl stop
cd $TOOL/nipe
sudo perl nipe.pl stop
#Updating IP variable
IP=$(curl -s ifconfig.me)
#Updating COUNTRY variable with current info
COUNTRY=$(geoiplookup $IP | awk {'print $4'} | sed 's/,//')
#Showing user their current ip and location
DISPLAY_SPOOFED_COUNTRY
#Checking if no longer anonymous
if [ "$COUNTRY" == "IL" ];then #If country IS Israel
echo -e "${GREEN}You are back to NOT being anonymous${STOP}\n"
echo -e "${CYAN}Hasta la vista, baby! ^_^"
else
echo -e "${RED}You are still anonymous!${STOP}"
fi
exit 0 #Exiting script
}
```

Demonstrating Script Performance -

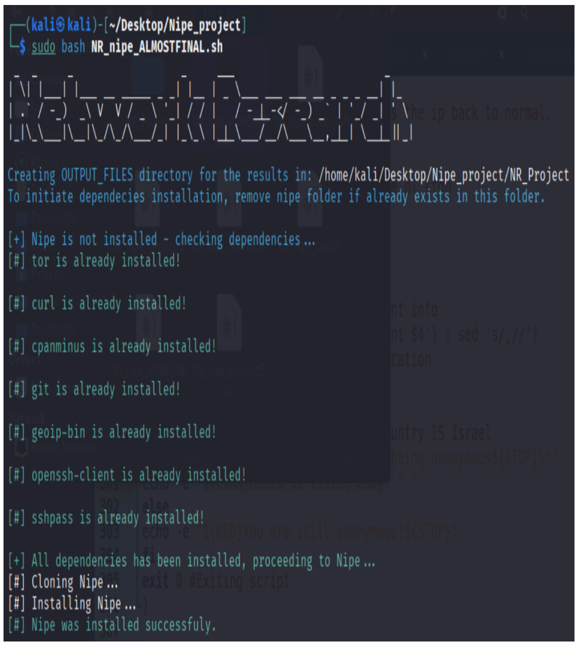
This part will show **some** of the script functionality in work.

Starting the script + install dependencies:



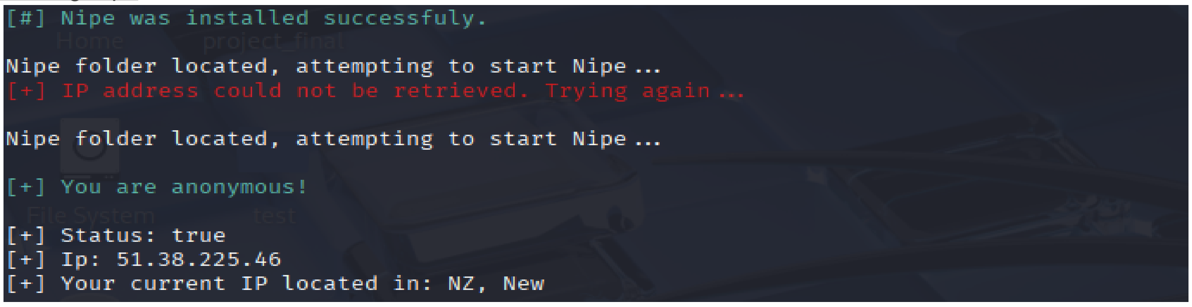
```
┌──(kali㉿kali)-[~/Desktop/Nipe_Project]
└─$ bash NR_nipe_ALMOSTFINAL.sh
You are not root. Exiting..

┌──(kali㉿kali)-[~/Desktop/Nipe_Project]
└─$ sudo bash NR_nipe_ALMOSTFINAL.sh
 _  _     _                   _     ___                          _    
| \| |___| |___ __ _____ _ _| |__ | _ \___ ___ ___ __ _ _ _ __| |_  
| .` / -_)  _\ V  V / _ \ '_| / / |   / -_|_-</ -_) _` | '_/ _| ' \ 
|_|\_\___|\__|\_/\_/\___/_| |_\_\ |_|_\___/__/\___\__,_|_| \__|_||_|

Creating OUTPUT_FILES directory for the results in: /home/kali/Desktop/Nipe_P
roject/NR_Project
To initiate dependecies installation, remove nipe folder if already exists in
 this folder.

[+] Nipe is not installed - checking dependencies...
[#] tor not found. Installing...
[#] tor was successfuly installed!

[#] curl not found. Installing...
[#] curl was successfuly installed!

[#] cpanminus not found. Installing...
[#] cpanminus was successfuly installed!

[#] git not found. Installing...
[#] git was successfuly installed!

[#] geoip-bin not found. Installing...
[#] geoip-bin was successfuly installed!

[#] openssh-client is already installed!

[#] sshpass not found. Installing...
[#] sshpass was successfuly installed!

[+] All dependencies has been installed, proceeding to Nipe...
[#] Cloning Nipe...
[#] Installing Nipe...
[#] Nipe was installed successfuly.
```

Starting the script(dependencies already installed):



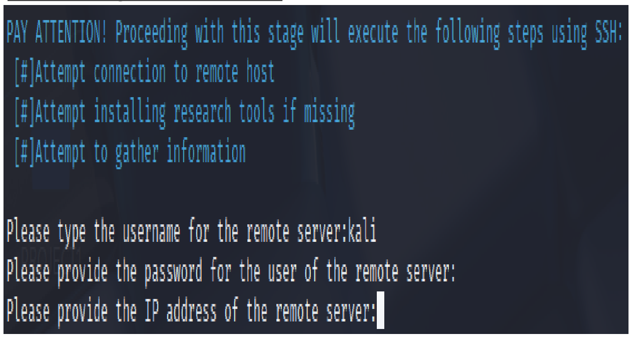
```
┌──(kali㉿kali)-[~/Desktop/Nipe_project]
└─$ sudo bash NR_nipe_ALMOSTFINAL.sh
 _  _     _                   _     ___                          _    
| \| |___| |___ __ _____ _ _| |__ | _ \___ ___ ___ __ _ _ _ __| |_  
| .` / -_)  _\ V  V / _ \ '_| / / |   / -_|_-</ -_) _` | '_/ _| ' \ 
|_|\_\___|\__|\_/\_/\___/_| |_\_\ |_|_\___/__/\___\__,_|_| \__|_||_|

Creating OUTPUT_FILES directory for the results in: /home/kali/Desktop/Nipe_project/NR_Project
To initiate dependecies installation, remove nipe folder if already exists in this folder.

[+] Nipe is not installed - checking dependencies...
[#] tor is already installed!

[#] curl is already installed!

[#] cpanminus is already installed!

[#] git is already installed!

[#] geoip-bin is already installed!

[#] openssh-client is already installed!

[#] sshpass is already installed!

[+] All dependencies has been installed, proceeding to Nipe...
[#] Cloning Nipe...
[#] Installing Nipe...
[#] Nipe was installed successfuly.
```

Starting Nipe



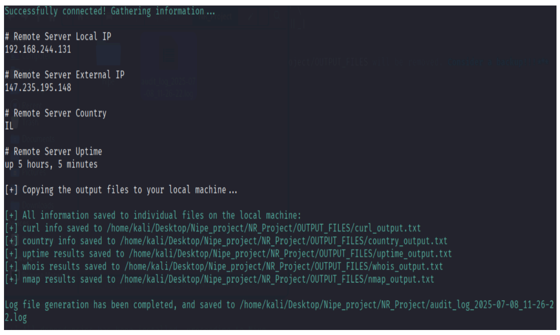
```
[#] Nipe was installed successfuly.

Nipe folder located, attempting to start Nipe ...
[+] IP address could not be retrieved. Trying again ...

Nipe folder located, attempting to start Nipe ...

[+] You are anonymous!

[+] Status: true
[+] Ip: 51.38.225.46
[+] Your current IP located in: NZ, New
```

Connecting via SSHPASS:



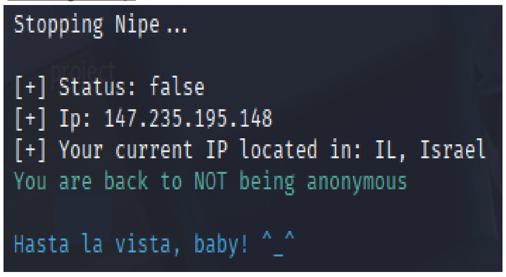
```
PAY ATTENTION! Proceeding with this stage will execute the following steps using SSH:
 [#]Attempt connection to remote host
 [#]Attempt installing research tools if missing
 [#]Attempt to gather information

Please type the username for the remote server:kali
Please provide the password for the user of the remote server:
Please provide the IP address of the remote server:
```

Information Gathering:



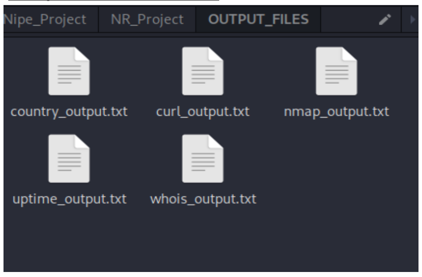
```
Successfully connected! Gathering information ...

# Remote Server Local IP
192.168.244.131

# Remote Server External IP
147.235.195.148

# Remote Server Country
IL

# Remote Server Uptime
up 5 hours, 5 minutes

[+] Copying the output files to your local machine ...

[+] All information saved to individual files on the local machine:
[+] curl info saved to /home/kali/Desktop/Nipe_project/NR_Project/OUTPUT_FILES/curl_output.txt
[+] country info saved to /home/kali/Desktop/Nipe_project/NR_Project/OUTPUT_FILES/country_output.txt
[+] uptime results saved to /home/kali/Desktop/Nipe_project/NR_Project/OUTPUT_FILES/uptime_output.txt
[+] whois results saved to /home/kali/Desktop/Nipe_project/NR_Project/OUTPUT_FILES/whois_output.txt
[+] nmap results saved to /home/kali/Desktop/Nipe_project/NR_Project/OUTPUT_FILES/nmap_output.txt

Log file generation has been completed, and saved to /home/kali/Desktop/Nipe_project/NR_Project/audit_log_2025-07-08_11-26-2
2.log
```

Exiting script:

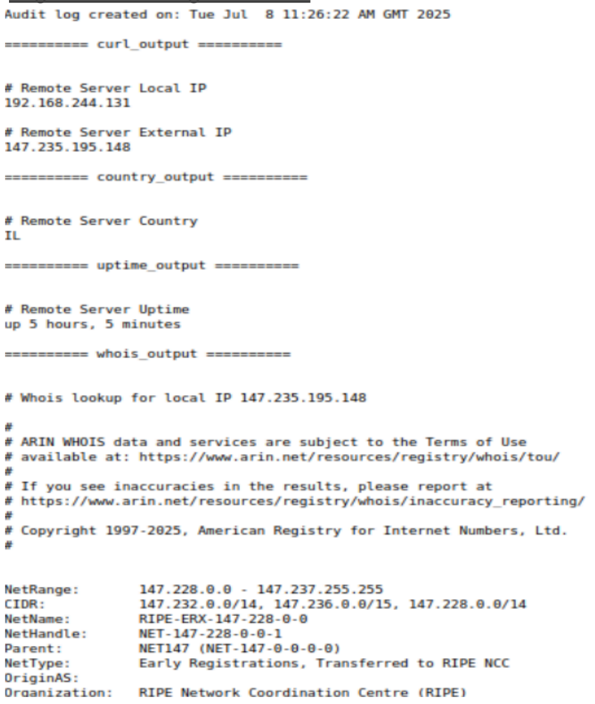


```
Stopping Nipe ...

[+] Status: false
[+] Ip: 147.235.195.148
[+] Your current IP located in: IL, Israel
You are back to NOT being anonymous

Hasta la vista, baby! ^_^
```

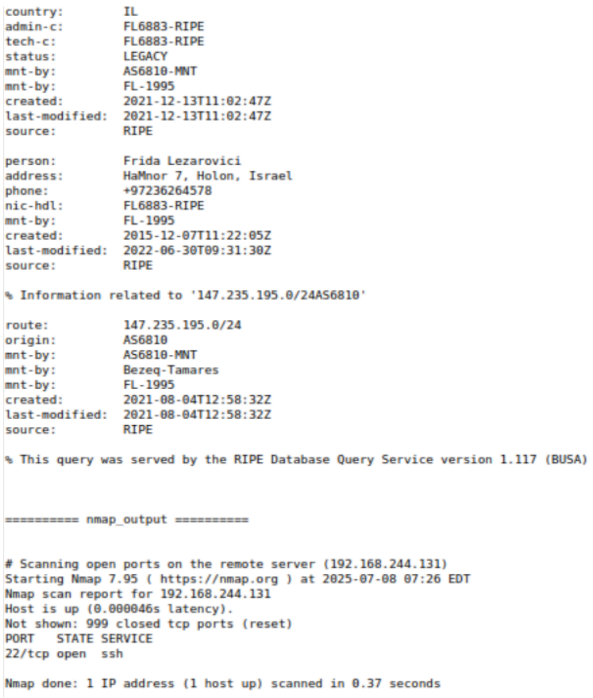
Output for each tool:



Log file combining all results:



```
Audit log created on: Tue Jul  8 11:26:22 AM GMT 2025

========== curl_output ==========


# Remote Server Local IP
192.168.244.131

# Remote Server External IP
147.235.195.148

========== country_output ==========


# Remote Server Country
IL

========== uptime_output ==========


# Remote Server Uptime
up 5 hours, 5 minutes

========== whois_output ==========


# Whois lookup for local IP 147.235.195.148

#
# ARIN WHOIS data and services are subject to the Terms of Use
# available at: https://www.arin.net/resources/registry/whois/tou/
#
# If you see inaccuracies in the results, please report at
# https://www.arin.net/resources/registry/whois/inaccuracy_reporting/
#
# Copyright 1997-2025, American Registry for Internet Numbers, Ltd.
#


NetRange:       147.228.0.0 - 147.237.255.255
CIDR:           147.232.0.0/14, 147.236.0.0/15, 147.228.0.0/14
NetName:        RIPE-ERX-147-228-0-0
NetHandle:      NET-147-228-0-0-1
Parent:         NET147 (NET-147-0-0-0-0)
NetType:        Early Registrations, Transferred to RIPE NCC
OriginAS:
Organization:   RIPE Network Coordination Centre (RIPE)
```



```
country:        IL
admin-c:        FL6883-RIPE
tech-c:         FL6883-RIPE
status:         LEGACY
mnt-by:         AS6810-MNT
mnt-by:         FL-1995
created:        2021-12-13T11:02:47Z
last-modified:  2021-12-13T11:02:47Z
source:         RIPE

person:         Frida Lezarovici
address:        HaMnor 7, Holon, Israel
phone:          +97236264578
nic-hdl:        FL6883-RIPE
mnt-by:         FL-1995
created:        2015-12-07T11:22:05Z
last-modified:  2022-06-30T09:31:30Z
source:         RIPE

% Information related to '147.235.195.0/24AS6810'

route:          147.235.195.0/24
origin:         AS6810
mnt-by:         AS6810-MNT
mnt-by:         Bezeq-Tamares
mnt-by:         FL-1995
created:        2021-08-04T12:58:32Z
last-modified:  2021-08-04T12:58:32Z
source:         RIPE

% This query was served by the RIPE Database Query Service version 1.117 (BUSA)



========== nmap_output ==========


# Scanning open ports on the remote server (192.168.244.131)
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-08 07:26 EDT
Nmap scan report for 192.168.244.131
Host is up (0.000046s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE
22/tcp open  ssh

Nmap done: 1 IP address (1 host up) scanned in 0.37 seconds
```