Name: reut abergel
SID :s8
Unit: TMagen7736.38
Program code: NX201

Network Research project

Table of contents

1.Introduction

Hi! This is a friendly guide for my script,The script's role is to perform an Nmap and Whois network scan against an IP target provided by the user.

The uniqueness of the script is that it operates in two key security stages:

1. Anonymity check: It verifies that the machine running it is not exposing an Israeli IP, and if so, it uses the Nipe tool to mask the address.
2. Root check: You must add the Sudo command when running the script to provide permissions.
3. The script collects the following information from the remote server and saves it to local files:

Whois scan of the target (saved in whois_data.txt).
Nmap scan of the target (saved in nmap_data.txt).
A full log documentation of the actions (saved in scan_log.txt).

?Who is this guide for

1. Students in a cyber course.
2. for anyone who wants to gather information or get expirene in the network scaning invirment
3. Users who want to automate the process of installing tools, checking anonymity, and performing a scan from a remote server.

How the guide helps:

1. Explains the tool's purpose and the security logic behind it.
2. Shows exactly how to run the script step-by-step.
3. Describes what each main function in the script does (ROOT, INSTALL, ANON, RMSCAN) and analyzes the commands within it.

what is needed to run this script:

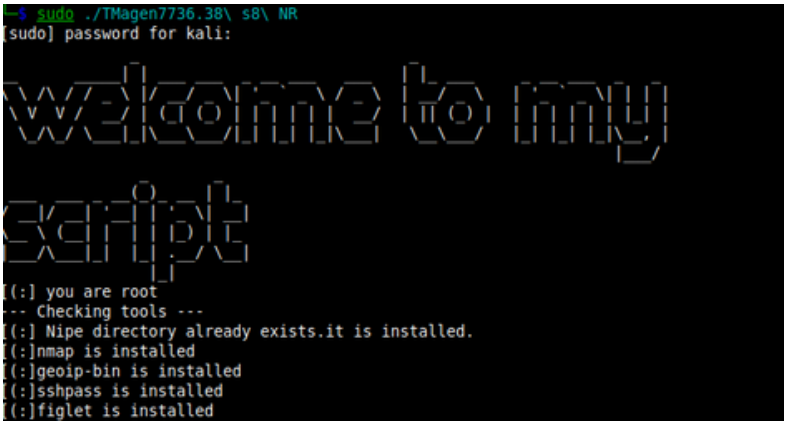
1. **kali linux:** vm or operating system to use apt for installations.
2. **Root permissions**: The script checks this automatically and will require sudo permissions to install tools.
3. **Internet connection:** Required for downloading tools, cloning Nipe from GitHub, and contacting the remote server.
4. **other vm mechine or remote server:** any kali Linux machine (or VPS) that you have access to with a username, password, and IP address.

How to run

1. **Execution**: You must run the script with root or sudo permissionss

```
└─$ sudo ./TMagen7736.38\ s8\ NR
```

Installations Automatic Process: The script will ask you the password for the sudo permmisions and then will start to run. It will automatically check if all the required tools are installed: Nipe, nmap, geoip-bin, sshpass, figlet. If a tool is missing, the script will install it automatically.



Automatic Process (Step 2 - Anonymity): The script will check your public IP. If it identifies that you are from Israel ("IL"), it will display an alert, automatically activate Nipe to change the IP, and check again.

```
[!!!]NOT anonymous - IP is from IL taking action,connecting to nipe....
you are anonymous Country: NL
```

User Input (Step 3 - Scan): After everything is ready, the script will ask you to enter 3 details for the remote scan:

Username for the target or remote server.

```
[*] enter remote ssh username:
```

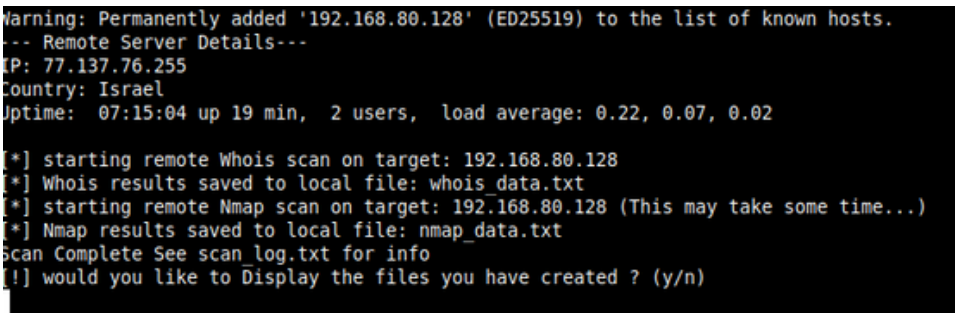
The SSH password

```
[*] Enter the remote server password:
```

The target's IP address (or server you want to scan).

```
[*] Enter the remote server public IP address:
```

Finish and Files: The script will connect to the remote server, perform the scans, and save the output in 3 files in the local directory. When finished, you will be asked if you want to display the content of these files in the terminal.



Purpose of the Tool

- The goal is to perform a network scan (Nmap and Whois) against a target, while maintaining two key security principles:

1. **Attacker Anonymity**: Ensuring the user's public IP is masked (using Nipe) before any action begins.
2. **Audit Trail & Data** : saving output to whois_data.txt, nmap_data.txt, and a scan_log.txt embodies this principle. This creates a reliable audit trail for your engagement, which is essential for analysis and writing a final report.

The tool turns a complex manual process (installations, IP check, Nipe activation, connecting to a target, running commands) into a single automated action.

Summary

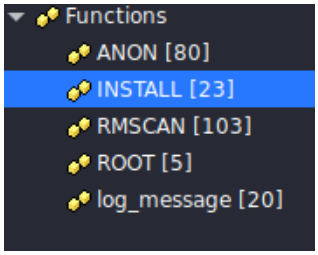
At the end of the process, you have a fully automated easy to use script that takes care of the necessary installations, checks and ensures anonymity, connects to a target, performs Whois and Nmap scans from it, and documents everything in an organized log file (scan_log.txt). All of this without exposing your original IP.

What I learned from building the script

- How to check if programs are installed and perform their automatic installation.
- How to clone a project from GitHub and run its installation.
- How to check a public IP and get its country name.
- How to use sshpass to automatically pass a password to SSH to execute remote commands.
- How to write organized logs with a timestamp.
- How to receive input from the user
- How to create a log based on the scans results and automat it to use diffrent files for each command

Analysis of functions and complex commands

The script is divided into 5 main functions:



Here we will explain what each function does and detail the complex commands found within it.

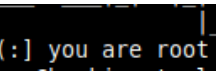
Function: ROOT

```
function ROOT()
{
    if [ "$(whoami)" == "root" ]
    then
        echo "[(:] you are root"
    else
        echo "[!] you are not root,exiting..."
        exit
    fi
}
```

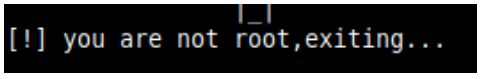
What it does: Checks if the user running the script is root.

Why: If the user is not root, the script prints an error message and exits. This is necessary because installing tools, using Nipe, and network operations require high privileges.

Topic: Checking Root User [Insert screenshot of the command for checking Root]



OR



Function: INSTALL

What it does: Verifies that all tools required for the script's operation are installed.

Process:Nipe check: Checks if the nipe directory exists. If not, it installs base tools, clones Nipe from GitHub, and installs it.

Additional tools check: Loops through a list of tools (nmap, geoip-bin, sshpass, figlet) and checks if they are installed. If not, it installs them automatically.

Complex commands in this function:

- Checking if Nipe is installed

```
function INSTALL() {
    echo "--- Checking tools ---"
  if [ ! -z "$(find / -type d -name "nipe" 2>/dev/null | head -n 1)" ]
```

Explanation: This is a complex line designed to check if the "nipe" directory exists somewhere on the system.

find / -type d -name "nipe": Searches the entire system (/) for a file of type directory (-type d) named "nipe".

2>/dev/null: Hides all error messages that might appear during the search.

if [! -z "..."]: The if condition checks: ! (the opposite of) -z (is the string empty?). Meaning: Is it not true that the result is empty?. In simple terms: Was something found? If a path was found, the condition is met, and the script assumes Nipe is installed.

- Checking package installation (with dpkg)

```
for pkg in  nmap geoip-bin sshpass figlet
    do
if dpkg -l "$pkg" >/dev/null 2>&1
    then
     echo "[(:]$pkg is installed"
     sleep 2
    else
     echo "[!]$pkg is not installed now installing"
      sudo apt update -y >/dev/null 2>&1
  sudo apt install -y "$pkg"
if dpkg -s "$pkg" >/dev/null 2>&1
    then
     echo "[(:]$pkg installation SUCCESS"
    else
     echo "[):]$pkg installation FAILED"
    fi
fi
```

Explanation: The line checks if a specific package (stored in the $pkg variable) is already installed.

dpkg -l "$pkg": This is the command that checks. If the package is installed, the command succeeds. If not, it fails.>/dev/null 2>&1: This is an important part that "silences" the command. >/dev/null sends the regular output to the “trash” (doesn't print to the screen), and 2>&1 sends the error messages to the same place.

Function: ANON

```
function ANON() {

    IP=$(curl -s ifconfig.co) # public IP
    CN=$(geoiplookup $IP| awk '{print $4}' |tr -d ",")  # country code of IP

    if [ "$CN" == "IL" ]
    then
        echo "[!!!]NOT anonymous - IP is from IL taking action,connecting to nipe...."
        #finding nipe
        NIPE_PATH=$(find / -type d -name "nipe" 2>/dev/null | head -n 1)
        cd $NIPE_PATH
         perl nipe.pl restart
         perl nipe.pl restart
         perl nipe.pl restart
         ANON
    else
         echo "you are anonymous Country: $CN"
    fi
```

[Insert screenshot of the output "NOT anonymous - IP is from IL taking action..."]

What it does: Ensures the network connection is anonymous (not from Israel) before starting the scan.

Process:

Saves the current public IP and its country code then Checks if the country code is "IL"

If yes (IP is Israeli): Prints an alert, locates Nipe, restarts it to change the IP, and calls itself again (recursion) to check the new IP.

If no (IP is not Israeli): Prints that you are anonymous and the fake country name.

Complex commands in this function:

Getting country code

```
IP=$(curl -s ifconfig.co) # public IP
CN=$(geoiplookup $IP| awk '{print $4}' |tr -d ",")  # country code of IP
```

Explanation: This line performs command chaining (piping):

geoiplookup $IP: Checks the IP we received from the command curl -s ifconfig.co and prints information about the country.

awk '{print $4}': This output is passed to awk, which tells it "take only the fourth word in the line" (which in this case is the country code, e.g., "IL,").

| tr -d ",": This output (e.g., "IL,") is passed to tr which tells it "delete (-d) the comma (,) character".

The final result ("IL") is saved in the CN variable.

Function: RMSCAN

```
function RMSCAN()
{
    echo "you are all set Starting Remote Scan process..."
    read -p "[*] enter remote ssh username:" SSH_USER
    read -p "[*] Enter the remote server password:" SSH_PASS
    read -p "[*] Enter the remote server public IP address:" SSH_IP
    echo "Target ip/domain is : $SSH_IP"
    #2.1 Display the details of the remote server (country, IP, and Uptime).
     REMOTE_IP=$(sshpass -p "$SSH_PASS"  ssh -o StrictHostKeyChecking=no $SSH_USER@$SSH_IP "curl -s ifconfig.co")
    REMOTE_COUNTRY=$(sshpass -p "$SSH_PASS" ssh -o StrictHostKeyChecking=no $SSH_USER@$SSH_IP "geoiplookup $REMOTE_IP" | awk '{print $5}')
    REMOTE_UPTIME=$(sshpass -p "$SSH_PASS" ssh -o StrictHostKeyChecking=no $SSH_USER@$SSH_IP "uptime")
    echo "--- Remote Server Details---"
    echo "IP: $REMOTE_IP"
    sleep 2
    echo "Country: $REMOTE_COUNTRY"
    sleep 2
    echo "Uptime: $REMOTE_UPTIME"
    sleep 2
    echo
    sleep 2
     # --- Start Logging ---
    log_message "Scan started ."
    log_message "Remote server used: $REMOTE_IP ($REMOTE_COUNTRY)"
    log_message "Target address: $SSH_IP"
    log_message "remote uptime is: $REMOTE_UPTIME"
```

What it does: This is the main function that manages the remote scan and creates the logs.

Process:

User input: Asks the user to enter the targets username password and ip addrres

Connects to the target and collects info like Country and Uptime of the target

Writes all these details to the scan_log.txt file.

Perform Whois scan: Connects to the target, runs whois,and redirects the output to the local whois_data.txt file.

Perform Nmap scan: Connects to the target, runs nmap -A , and redirects the output to the local nmap_data.txt file.

Display files: Asks the user if they want to display all the files that were created.

Complex commands in this function:

Running commands via SSH and saving output

```
    #2.1 Display the details of the remote server (country, IP, and Uptime).
     REMOTE_IP=$(sshpass -p "$SSH_PASS"  ssh -o StrictHostKeyChecking=no $SSH_USER@$SSH_IP "curl -s ifconfig.co")
    REMOTE_COUNTRY=$(sshpass -p "$SSH_PASS" ssh -o StrictHostKeyChecking=no $SSH_USER@$SSH_IP "geoiplookup $REMOTE_IP" | awk '{print $5}')
    REMOTE_UPTIME=$(sshpass -p "$SSH_PASS" ssh -o StrictHostKeyChecking=no $SSH_USER@$SSH_IP "uptime")
    echo "--- Remote Server Details---"
    echo "IP: $REMOTE_IP"
    sleep 2
    echo "Country: $REMOTE_COUNTRY"
    sleep 2
    echo "Uptime: $REMOTE_UPTIME"
    sleep 2
    echo
    sleep 2
     # --- Start Logging ---
    log_message "Scan started ."
    log_message "Remote server used: $REMOTE_IP ($REMOTE_COUNTRY)"
    log_message "Target address: $SSH_IP"
    log_message "remote uptime is: $REMOTE_UPTIME"
```

Explanation: This is the line that performs the remote scan and saves its output directly to a local file.

Let's analyze it: sshpass -p "$SSH_PASS": This is the command that allows password automation. It receives the password (-p) from the $SSH_PASS variable.

ssh -o StrictHostKeyChecking=no $SSH_USER@$SSH_IP: This is the regular SSH command. The -o=no part is important: it tells SSH "just connect." This prevents the script from getting stuck on a security question.

whois $SSH_IP: This is the specific command that will run on the remote server. In this case, a whois scan against the target (which is stored in $SSH_IP).

>> whois_data.txt: This is the last and most important part. The >> symbol is called "Append Redirect." It takes all the output that the sshpass command printed to the screen and adds it to the end of a local file named whois_data.txt. If the file doesn't exist, it creates it.

Function: log_message()

What it does: Its only job is to take a text message (an argument) and write it to the $LOG_FILE (scan_log.txt) with a precise timestamp in front of it.

Why: Instead of writing the long echo "$(date...) ... $LOG_FILE" command every time you want to log an action, you can now just write log_message "Starting Nmap scan...". It makes the script much cleaner, easier to read, and ensures all your log entries have the exact same format.

Complex commands in this function:

Writing a Formatted Log Entry

```
log_message() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] - $1" >> "$LOG_FILE"
```

Explanation: This line creates the log entry.

$(date '+%Y-%m-%d %H:%M:%S'): This command runs first and generates the current date and time in a standard format (2025-10-13 16:00:10).

$1: This is a special variable that holds the first argument you pass to the function.

>> "$LOG_FILE": This appends (adds) that single line of text to the end of the log file, without erasing what's already there.