

Network Research Project: Anonymous Enumeration Tool

Project by: Reut Abergel

Table of Contents

1. Introduction
2. Target Audience
3. requirements
4. Operational Guide

Step 1: Execution

Step 2: Automated Initialization

Step 3: Anonymity Enforcement

Step 4: Target Configuration

Step 5: Completion & Reporting

5. Purpose & Security
6. Technical Analysis

Function: ROOT

Function: INSTALL

Function: ANON

Function: RMSCAN

Function: log_message

7. Technical Skills DGained
8. Summary

1. Introduction

This document serves as a technical guide for the **Anonymous Enumeration Tool** script. The primary function of this tool is to perform Nmap and Whois network scans against a user-specified IP target.

The script is distinct in its adherence to two critical security protocols:

1. **Anonymity Verification:** It automatically verifies that the host machine is not exposing an Israeli IP address. If an Israeli IP is detected, the script utilizes the **Nipe** tool to mask the address before proceeding.
2. **Privilege Enforcement:** The script enforces a root privilege check, requiring execution via sudo to ensure full functionality.

Data Collection

The tool aggregates the following data from the remote server, saving it to local files:

- **Whois Scan:** Domain and IP registration details (saved to whois_data.txt).
- **Nmap Scan:** Network discovery and security auditing data (saved to nmap_data.txt).
- **Audit Log:** A comprehensive documentation of all actions taken during the session (saved to scan_log.txt).
-

2. Target Audience

This guide and tool are designed for:

- **Cybersecurity Students:** Individuals enrolled in cyber courses requiring network analysis tools.
- **Security Enthusiasts:** Users seeking to gather information or gain experience in network scanning environments.
- **Automation Seekers:** Users looking to automate the setup of security tools, anonymity checks, and remote server scanning.

3. Requirements

To successfully execute this script, the following requirements must be met:

1. **Operating System:** Kali Linux (VM or native OS) to support apt package management.
2. **Permissions:** Root/Sudo privileges are mandatory for tool installation and network operations.
3. **Connectivity:** An active internet connection is required for downloading dependencies (Nipe, etc.) and contacting remote servers.
4. **Target Environment:** A secondary machine or remote server (e.g., VPS or Kali VM) with valid SSH credentials (IP, Username, Password).

4. Operational Guide

Step 1: Execution: The script must be initiated with elevated privileges

```
(kali㉿kali)-[/var/run/vmblock-fuse/blockdir/o6YTVI]
$ sudo ./Network_Research_Anonymous_Enumeration.sh
```

Step 2: Automated Initialization: Upon execution, the script prompts for the sudo password and begins the initialization process. It automatically verifies the presence of required dependencies: Nipe, nmap, geoip-bin, sshpass, and figlet. Missing tools are installed automatically.



```
:: Automated Enumeration & Anonymity Tool ::

[+] User is root
— Checking Dependencies —
[+] Nipe directory found.
[+] Package nmap is installed.
[+] Package geoip-bin is installed.
[+] Package sshpass is installed.
[+] Package figlet is installed.
```

Step 3: Anonymity Enforcement: The script analyzes the user's public IP. If the origin is identified as Israel ("IL"), the system triggers an alert and activates **Nipe** to reroute traffic. It recursively checks the IP until anonymity is confirmed.

```
— Network Anonymity Check —
[!!!] ALERT: IP is from IL. Connecting to Nipe...

— Network Anonymity Check —
[+] Status: Anonymous | Country: AT
```

Step 4: Target Configuration: Once the environment is secure, the user is prompted to input the target parameters:

1. **Username:** The SSH username for the remote server.
2. **Password:** The SSH password.
3. **Target IP:** The public IP address of the server to be scanned.

```
— Remote Target Configuration —
[*] Enter remote ssh username: kali
[*] Enter remote server password:
[*] Enter remote server IP: 192.168.80.130
[*] Target locked: 192.168.80.130
```

Step 5: Completion & Reporting: The script establishes a connection to the target, executes the scans, and saves the outputs locally. Upon completion, the user is given the option to display the generated file contents directly in the terminal.

```
— Remote Server Recon —
IP Address : 5.29.16.91
Location   : Israel
Uptime     : 06:46:17 up 39 min, 2 users, load average: 0.00, 0.00, 0.00

[*] Starting Whois scan on 192.168.80.130 ...
[+] Whois data saved to: whois_data.txt
[*] Starting Nmap scan on 192.168.80.130 (Please wait ... )
[+] Nmap results saved to: nmap_data.txt

✓ Scan Process Complete. Log updated.

[!] Display created files? (y/n)
^_
```

5. Purpose & Security

The tool transforms a complex, manual workflow into a single automated execution, adhering to two main security principles:

1. **Attacker Anonymity:** Ensures the operator's true public IP is masked via Nipe before any engagement.
2. **Audit Trail & Data Integrity:** Generates a reliable audit trail (scan_log.txt) alongside raw data files (whois_data.txt, nmap_data.txt), which is essential for after-action review.

6. Technical Analysis

The script relies on five core functions to manage the workflow:

1. Function: ROOT Validates execution privileges: Checks if the current user is "root". If not, the script terminates immediately. High privileges are required for package installation and network interface manipulation.

```
#1. Installations and Anonymity Check
function ROOT()
{
    if [ "$(whoami)" == "root" ]
    then
        echo -e "${OK} User is ${BOLD}root${NC}"
    else
        echo -e "${ERR} You are not root, exiting..."
        exit
    fi
}

# Banner in Cyan
echo -e "${CYAN}$(figlet "Anonymous Scan")${NC}"
echo -e "${BLUE}:: Automated Enumeration & Anonymity Tool ::${NC}"
echo ""

#3.2 Create a log
LOG_FILE="scan_log.txt"

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

2. Function: INSTALL

Purpose: Ensures all essential tools are installed:

- **Nipe Check:** Searches for the nipe directory using find. If absent, it clones the repository from GitHub and installs dependencies.
- **Package Loop:** Iterates through a list of tools (nmap, geoip-bin, etc.). It utilizes dpkg to check installation status. If a package is missing, it runs apt install silently, redirecting standard output and errors to /dev/null to maintain a clean interface.

```

function INSTALL()
{
    echo -e "${BOLD}--- Checking Dependencies ---${NC}"

    if [ ! -z "$(find / -type d -name "nipe" 2>/dev/null | head -n 1)" ]
    then
        echo -e "${OK} Nipe directory found."
    else
        echo -e "${WARN} Nipe is not installed. Starting installation..."

        echo -e "${INFO} Installing tools: ${WHITE}git, perl${NC}"
        sudo apt-get update -y > /dev/null 2>&1
        sudo apt-get install -y git cpanminus libwww-perl libjson-perl > /dev/null 2>&1

        echo -e "${INFO} Cloning Nipe from GitHub..."
        git clone https://github.com/htrgouvea/nipe.git "$HOME/nipe"

        if [ -d "$HOME/nipe" ]
        then
            cd "$HOME/nipe"
            echo -e "${INFO} Installing Nipe dependencies..."
            sudo cpanm --notest Switch JSON LWP::UserAgent Config::Simple

            echo -e "${INFO} Running Nipe installer..."
            sudo perl nipe.pl install

            if [ -f "$HOME/nipe/nipe.pl" ]
            then
                echo -e "${OK} Nipe installation ${GREEN}SUCCESSFUL${NC}."
            else
                echo -e "${ERR} Nipe installation ${RED}FAILED${NC}."
                exit 1
            fi
            cd - > /dev/null
        else
            echo -e "${ERR} FAILED to clone Nipe."
            exit 1
        fi
    fi

    for pkg in nmap geoip-bin sshpass figlet
    do
        if dpkg -l "$pkg" >/dev/null 2>&1
        then
            echo -e "${OK} Package ${WHITE}$pkg${NC} is installed."
            sleep 0.5
        else
            echo -e "${WARN} Package ${WHITE}$pkg${NC} is missing. Installing..."
            sudo apt update -y >/dev/null 2>&1
            sudo apt install -y "$pkg"
        fi
        if dpkg -s "$pkg" >/dev/null 2>&1
        then
            echo -e "${OK} ${WHITE}$pkg${NC} installation ${GREEN}SUCCESS${NC}"
        else
            echo -e "${ERR} ${WHITE}$pkg${NC} installation ${RED}FAILED${NC}"
        fi
    done
}

```

3. Function: ANON

Purpose: Guarantees network anonymity:

1. Retrieves the public IP and parses the country code using command chaining with curl, geoiplookup, and awk.
2. **Conditional Check:** If the country code is "IL", it locates the Nipe directory and executes a restart command to rotate the IP.
3. **Recursion:** The function calls itself again to re-verify the new IP until a non-Israeli address is confirmed.

```
#1.3/1.4 Anonymity Check
function ANON()
{
    echo -e "\n${BOLD}--- Network Anonymity Check ---${NC}"
    IP=$(curl -s ifconfig.co)
    if [ -z "$IP" ]
        then
            echo -e "${ERR} Could not fetch IP. Retrying..."
            sleep 5
            ANON
            return
    fi

    CN=$(geoiplookup $IP| awk '{print $4}' |tr -d ",")
    if [ "$CN" == "IL" ]
        then
            echo -e "${RED}[!!!] ALERT:${NC} IP is from ${RED}IL${NC}. Connecting to Nipe..."
            NIPE_PATH=$(find / -type d -name "nipe" 2>/dev/null | head -n 1)
            cd $NIPE_PATH
            perl nipe.pl restart
            perl nipe.pl restart
            perl nipe.pl restart
            ANON
        else
            echo -e "${OK} Status: ${GREEN}Anonymous${NC} | Country: ${WHITE}$CN${NC}"
    fi
}
```

4. Function: RMSCAN

Purpose: The main function of the tool, remote scanning, and data logging:

- **Connection:** Uses sshpass to automate password entry, allowing non-interactive SSH connections.
- **StrictHostKeyChecking:** Disabled to prevent the script from hanging on SSH verification prompts.
- **Data Redirection:** Executes remote commands (whois, nmap) and appends the output to local files using the >> (wohis/nmap.txt).

```

function RMSCAN()
{
    echo -e "\n${BOLD}--- Remote Target Configuration ---${NC}"
    # We use echo -n for input to keep the cursor on the same line nicely
    echo -e -n "${INFO} Enter remote ssh username: "
    read SSH_USER

    echo -e -n "${INFO} Enter remote server password: "
    read -s SSH_PASS # -s hides the password typing for security
    echo ""

    echo -e -n "${INFO} Enter remote server IP: "
    read SSH_IP

    echo -e "${INFO} Target locked: ${CYAN}$SSH_IP${NC}"

#2.1 Display Details
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 -e "\n${BOLD}--- Remote Server Recon ---${NC}"
echo -e "IP Address : ${CYAN}$REMOTE_IP${NC}"
sleep 1
echo -e "Location : ${CYAN}$REMOTE_COUNTRY${NC}"
sleep 1
echo -e "Uptime : ${WHITE}$REMOTE_UPTIME${NC}"
sleep 1

log_message "Scan started."
log_message "Remote used: $REMOTE_IP ($REMOTE_COUNTRY)"

#2.2 Whois
echo -e "\n${INFO} Starting ${BOLD}Whois${NC} scan on ${CYAN}$SSH_IP${NC}..."
sshpass -p "$SSH_PASS" ssh -o StrictHostKeyChecking=no $SSH_USER@$SSH_IP "whois $SSH_IP" >> whois_data.txt
echo -e "${OK} Whois data saved to: ${WHITE}whois_data.txt${NC}"
log_message "Whois saved."

#2.3 Nmap
echo -e "${INFO} Starting ${BOLD}Nmap${NC} scan on ${CYAN}$SSH_IP${NC} (Please wait...)"
sshpass -p "$SSH_PASS" ssh $SSH_USER@$SSH_IP "nmap -A $SSH_IP" >> nmap_data.txt
echo -e "${OK} Nmap results saved to: ${WHITE}nmap_data.txt${NC}"
log_message "Nmap saved."

echo -e "\n${GREEN} Scan Process Complete.${NC} Log updated."
log_message "Scan finished."

```

5. Function: log_message

Purpose: Maintains a standardized event log.: Accepts a text argument and appends it to scan_log.txt, prefixed with a precise timestamp generated by \$(date ...). This ensures a clean, chronological record of all automated actions.

```

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

```

7. Technical Skills Gained

Purpose: Through this project, I developed the following technical skills:

- **Automated Environment Setup:** Scripting the detection and installation of software packages.
- **Network Intelligence:** Programmatically retrieving and parsing public IP and geolocation data.
- **SSH Automation:** Utilizing sshpass for handling remote authentication and command execution.
- **Data Management:** Implementing structured logging with timestamps and managing file I/O streams.
- **User Interaction:** Capturing and processing user input effectively.

8. Summary

This project resulted in a fully automated, user-friendly tool that handles dependencies, enforces anonymity, connects to remote targets, and performs detailed reconnaissance. It successfully documents all activities in an organized log while protecting the operator's identity.