Made by: Omer Shor

Vulner.sh: Comprehensive Network Vulnerability Scanner - Detailed Breakdown

1. Script Overview

Vulner.sh is an advanced Bash script designed for network vulnerability assessment. It combines various cybersecurity tools and techniques to provide a comprehensive analysis of target networks or hosts.

2. Initial Setup

Shebang and Comments

```
#!/bin/bash
#Made by Omer Shor
```

The shebang specifies that this is a Bash script. The comment credits the author.

Welcome Message and Root Check

```
figlet Vulner
echo "[#] Welcome to the vulner PTool..."
if [[ ${id -u} != 0 ]]
    then
        echo "[-] Please run the script with root account"
        exit 1
    else
        echo "[+] You will move forward to start scanning your target, Enjoy"
fi
```

This section displays a welcome message using `figlet` and checks if the script is run with root privileges. If not, it exits with an error message.

Directory and File Setup

```
TS=${date +%H:%M}
vuln_dir="vulner_results_$TS"
mkdir -p $vuln_dir
cd $vuln_dir
report_file="$vuln_dir/audit_file.$TS.txt"
```

Creates a timestamped directory for results and sets up a report file.

3. Target Input and Validation

IP Validation Function

```
validate_ip() {
    local ip=$1
    local cidr=32
    local ip_regex="^([0-9]{1,3}\.){3}[0-9]{1,3}$"
    local cidr_regex="^([0-9]{1,3}\.){3}[0-9]{1,3}/([0-9]|[1-2][0-9]|3[0-2])$"
    if [[ $ip =~ $ip_regex ]] || [[ $cidr =~ $cidr_regex ]]; then
        return 0
    else
        return 1
    fi
}
```

This function validates if the input is a valid IP address or CIDR notation using regex.

Target Input Loop

```
while true; do
    read -p "[?] Please Enter a valid IP address for your target [network/host]: " target
    if validate_ip "$target" "$target"; then
        break
    else
        echo -e "${RED}[-]${NC} Your IP address input is NOT valid, please enter a valid IP address
    fi
done
```

Continuously prompts the user for a valid IP address or CIDR notation until a valid input is provided.

Network Target Handling

```
if [[ "$target" == *"/"* ]]; then
    echo "[#] Please select one target from the targets list"
    targets+=($(nmap -sn $target))
    nmap -O --top-ports 1 $target > targets2
    cat ./targets2 | grep -Eo '\b([0-9]{1,3}\.){3}[0-9]{1,3}\b' -e "OS details:" | awk '/Nmap scan
    read -p "[?] Please enter your choice here: " target
    rm -r targets2
fi
```

If a network (CIDR) is provided, this section performs a quick scan and allows the user to select a specific target from the network. It removes the temporary file 'targets2' after use.

4. Scanning Options

Scan Type Selection

```
read -p "[?] Please chose [B]asic scan or [F]ull scan, basic scan is default, full include service
```

Prompts the user to choose between a basic or full scan.

Basic Scan

```
if [ "$scan_type" == B ] || [ "$scan_type" == b ] ;
    then
        echo "[#] You chose to run a Basic scan on the target"
        echo "[#] The script will run a basic scan on the target $target"
        nmap -sV --top-ports=50 $target -oN $vuln_dir.scaning_resulte.$scan_type.$TS.txt -oX $vuln_
        sleep 5
        echo "[#] the nmap scan on target $target is complete."
```

Performs a basic Nmap scan on the top 50 ports with version detection.

Full Scan

```
elif [ "$scan_type" == F ] || [ "$scan_type" == f ] ;
    then
        echo "[#] You chose to run a full scan on the target include service version and vuln and O
        echo "[#] The script will run a full scan on the target $target"
        echo "[#] The scanning will take 2-5 Min, Don't stop the script!"
        nmap -sV -p- -O --script=vuln $target -oN $vuln_dir.scaning_resulte.$scan_type.$TS.txt -oX
        sleep 10
        echo "[#] the nmap scan on target $target is complete."
        echo "[#] starting using searchsploit for Mapping vulnerabilities"
        for x in $(cat $vuln_dir.scaning_resulte.$scan_type.$TS.txt | grep CVE | awk -F / '{print $
```

Performs a full Nmap scan on all ports, with OS detection, version scanning, and vulnerability scripts. It also uses Searchsploit to find potential exploits for detected CVEs.

5. Service Check

service_check() Function

```
function service_check(){
    echo "[#] Checking for the open port available on the target with auth, like SSH, FTP and TELNE
    open_ssh_port=$(cat $vuln_dir.scaning_resulte.$scan_type.$TS.txt | grep open | grep -Eo '[0-9]+
    open_ftp_port=$(cat $vuln_dir.scaning_resulte.$scan_type.$TS.txt | grep open | grep -Eo '[0-9]+
    open_telnet_port=$(cat $vuln_dir.scaning_resulte.$scan_type.$TS.txt | grep open | grep -Eo '[0-
    open_rdp_port=$(cat $vuln_dir.scaning_resulte.$scan_type.$TS.txt | grep open | grep -Eo '[0-9]+

    # Check for each service and print results
}
```

Checks for open authentication services (SSH, FTP, Telnet, RDP) on the target based on the scan results.

6. Brute Force Attack

Brute_force() Function

```
function Brute_force() {
    while true; do
        read -p "[?] Would you like to perform a brute force attack? (Y/N): " user_choice
        user_choice=$(echo "$user_choice" | tr '[:upper:]' '[:lower:]')
        if [ "$user_choice" = "y" ]; then
            echo "[#] You chose to perform a brute force attack."
            break
        elif [ "$user_choice" = "n" ]; then
            echo "[#] You chose not to perform a brute force attack. Exiting."
            exit
        else
            echo "[!] Invalid response. Please enter 'Y' or 'N'."
        fi
    done
}
```

Asks the user if they want to perform a brute force attack and handles the response.

Medusa_install() and Hydra_install() Functions

```
function Medusa_install() {
    if ! command -v medusa &> /dev/null 2>&1; 
        then
            echo "[-] medusa is not installed"
            echo "[*] start installing medusa"
            sudo apt install medusa &> /dev/null 2>&1
    else
        echo "[+] medusa is installed!"
    fi
}

function Hydra_install() {
    # Similar structure to Medusa_install()
}
```

These functions check if Medusa and Hydra are installed, and install them if they're not present.

BAT() Function (Brute Force Attack)

```
function BAT(){
    echo "[#] The tool will Brutforce attack on the target, To check weak passwords"
    # ... (warnings and service selection)

    # Example for SSH attack
    if [ $target_port == 1 ];
        then
            attack_port="ssh"
            # ... (password list selection)
            if [ $pass_choice == 1 ];
                then
                    read -p "[?] Please enter file path (full) for users accounts: " users_list
                    read -p "[?] Please enter file path (full) for passwords: " passwords_list
                    echo "[#] Starting the attack now, findings will be save to a file - found_acco
                    medusa -h $target -U $users_list -P $passwords_list -M $attack_port > audit_BAT
                    cat audit_BAT.txt | grep -B 1 -A 1 -ie "found" -e "login:" > found_accounts.txt
                    cat found_accounts.txt | grep "SUCCESS"
                    echo "[#] The $attack_port password scan is complete"
                elif [ $pass_choice == 2 ];
                    then
                        sudo git clone https://github.com/shawntns/top-100-worst-passwords.git &> /dev/
                        read -p "[?] Please enter file path (full) for users accounts: " users_list
                        echo "[#] Starting the attack now, findings will be save to a file - found_acco
                        medusa -h $target -U $users_list -P ./top-100-worst-passwords/dic.txt -M $attac
                        cat audit_BAT.txt | grep -B 1 -A 1 -ie "found" -e "login:"  > found_accounts.tx
                        cat found_accounts.txt | grep "SUCCESS"
                        echo "[#] The $attack_port password scan is complete"
            fi
    # ... (similar blocks for FTP, Telnet, and RDP)
    fi
}
```

This function performs the brute force attack on the selected service using either Medusa or Hydra, depending on the service. It includes an option to use a predefined list of common weak passwords.

7. Main Execution

Script Execution Flow

```
service_check
sleep 5
Brute_force
Medusa_install
Hydra_install
sleep 5
BAT
chmod 777 *
chmod 777 *
echo "[#] Saving everything into a zip file"
cd ..
zip -r $vuln_dir $vuln_dir &> /dev/null 2>&1
```

This section outlines the main execution flow of the script, calling the defined functions in sequence and finalizing by zipping the results. The script includes two chmod commands to ensure proper permissions.

Conclusion

Vulner.sh is a sophisticated Bash script that automates various aspects of network vulnerability assessment. It combines multiple tools like Nmap, Medusa, and Hydra to provide a comprehensive security analysis. The script demonstrates advanced Bash scripting techniques and a deep understanding of network security principles. It includes features such as IP validation, network scanning, service detection, and brute force attack capabilities, making it a powerful tool for cybersecurity professionals and ethical hackers.