UNIVERSITÀ DI TRENTO

Department of Information Engineering and Computer Science

Bachelor's Degree in
Computer Science

Final Dissertation

Automated framework for IoT security testing

Supervisor

Crispo Bruno

Student

Mariotti Matteo

Academic year 2022/2023

Acknowledgements

...thanks to... hello

Contents

Abstract

In the last few years, the security of IoT devices has become a major concern in the field of Computer Science. In fact we are surrounded by always new devices, such as cameras, smart speakers, and domotic devices in general, that have become an integral part of our lives.
This means we are more and more exposed to attack that can compromise our privacy and security.
The focus of this thesis is to automate the process of finding vulnerabilities in such devices, by the means of finding the right tools and scripting the various phases of the process.
This work will divided into two parts: a first more theoretical part, where we will analyze the proposed tools and compare them to the used in previous works. We will also analyze the scripts created to help the automation and explain their capabilities.
In a second, more practical part, we will test these tools on a real device, focusing on the actions that have been successfully automated, and hence require little to no user interaction.

Part I

Tools

1 Introduction

In this part of the dissertation we will discuss the improvements over previous works [7], both in terms of alternative tools with respect to the ones presented in that thesis, and in terms of additional areas where we can search for vulnerabilities.
In particular we will discuss the following topics:

- Gaining Wi-Fi access to the target network and exploring its weak points
- Attacks to the target network from the inside
- Attacks to the exported services of the target device, in particular web services
- An extension to the previous work, which allows to retrieve the memory image of the target device and analyze it offline

2 Wi-Fi access

In the thesis on which [7] this work is based it was suggested to use the Fern wifi cracker [4] to gain access to the target newtwork. Tis tool is basically a GUI for the aircrack-ng suite [1] and exposes all the features of the command line tool. It can in fact perform a lot of other attacks, aside from the WPA/WPA2 cracking, such as:

- Automatic attacks on the access point
- MITM attacks
- Bruteforce attacks using protocols such as FTP, HTTP/HTTPS and TELNET
- Deauthentication, access point spoofing and replay attacks

Nevertheless, as the goal of this work is to automate the process, the command line tool may prove to be more useful, as it can be easily integrated in a script.
The Aircrack-ng suite is the most portable of the tools presented here, as it can be used from virtually every operating system broadly used today, such as Windows, MacOS, a lot of Linux distributions and even from BSD operating systems.
There is also a Docker container for running it without installing it on the main system. It can be installed using the right packaged version for the host OS or by compiling it from source [2].

Other tools that can be employed instead of aircrack-ng are:

- Reaver [5]: a tool that exploits a flaw in the WPS implementation of some routers using the WPS Pixie-Dust and other brute-force attacks
- Wifite [6]: a python script that is a frontend for a lot of tools, including aircrack-ng, reaver, cowpatty, pyrit, tshark, etc. It is therefore perfect to automate the process of vulnerability assessment of a wifi network
- Bully [3]: similar to Reaver

The suggested tool is Wifite, as it is the most complete and it is a frontend to all other tools presented here. It is also a command line program, so it can be easily used in a script and it is also very easy to use because of the almost complete automation of the process. Unfortunately it needs many dependencies to reach its full potential, but many of them are optional. It can be downloaded using git in the following way:

```
git clone https://github.com/derv82/wifite2.git
```

It can then be run opening the terminal in the cloned directory and running:

```
sudo ./Wifite.py
```

3 Memory analysis

Bibliography

[1] Aircrack-ng. https://www.aircrack-ng.org/. accessed 15/06/2023.

[2] Aircrack-ng git repository. https://github.com/aircrack-ng/aircrack-ng. accessed 15/06/2023.

[3] Bully. https://github.com/aanarchyy/bully. accessed 15/06/2023.

[4] Fern wifi cracker. https://github.com/savio-code/fern-wifi-cracker. accessed 15/06/2023.

[5] Reaver. https://code.google.com/archive/p/reaver-wps/. accessed 15/06/2023.

[6] Wifite. https://github.com/derv82/wifite2. accessed 15/06/2023.

[7] Bringhenti Alessandro. A framework for the security analysis of iot devices, 2022.

Appendix A Scripts

```
#!/bin/bash

# Colors
RED='\033[0;31m'
GREEN='\033[0;32m'
NORMAL_COLOR='\033[0m'
OUTPUT_FILE="output"
IP=""
FULL=false
UDP=false
TCP=false
OS=false
MODE=""
DISCOVERY=false
ATTACK=false

# Helper functions
error(){
    printf "${RED}$1"
    exit 2
}

print_header(){
    printf "${GREEN}--------------------\n"
    printf "${GREEN}$1${NORMAL_COLOR}\n"
}

run_command(){
    printf "\n\n" >> $OUTPUT_FILE
    printf "${GREEN}--------------------\n"
    printf "${GREEN}$1\n"
    printf "${GREEN}***Executing command: $2 ${NORMAL_COLOR}\n"
    $2 | tee -a $OUTPUT_FILE || error "Failed executing $1"
}

print_usage(){
    printf "Usage: scan [OPTIONS]\n"
    printf "\t-u\t\tperform udp scan\n"
    printf "\t-u\t\tperform attack with default scripts\n"
    printf "\t-t\t\tperform tcp scan\n"
    printf "\t-f\t\tfull scans\n"
    printf "\t-d\t\tdiscovery mode (only -i and -o relevant)\n"
    printf "\t-v\t\tscan for OS version\n"
    printf "\t-o FILENAME\tredirect output to FILENAME\n"
    printf "\t-i IP\t\tscan specified IP\n"
    printf "\t-m MODE\t\ttcp scan mode\n"
}


# Perform TCP scan
tcp_scan(){
    if [ $FULL = true ]
    then
        # Full scan
        run_command "Full TCP scan" "nmap -p- -s${MODE} ${IP}"
    else
        # Default scan
        run_command "Default TCP scan" "nmap -s${MODE} ${IP}"
    fi
}

# Perform UDP scan
udp_scan(){
    if [ $FULL = true ]
    then
        # Perform UDP scan
        run_command "Full UDP scan" "nmap -sU -p- $IP --max-rtt-timeout 20ms --max-retries
            0"
    else
        # Perform UDP scan
        run_command "Default UDP scan" "nmap -sU $IP --max-rtt-timeout 20ms --max-retries 0"
    fi
}

# Discover hosts in the network
discovery(){
    run_command "Scanning network" "nmap -sn $IP"
    exit 0
}

attack(){
    print_header "Starting to attack discovered ports"
    PORTS=$(cat $OUTPUT_FILE | grep open | cut -d " " -f 1,2,3,4 | grep tcp | sort | uniq |
        cut -d "/" -f 1)

    for p in $PORTS
    do
        run_command "Testing port $p" "nmap -sC -p$p $IP"
    done
}

while getopts 'avdhuftm:o:i:' flag; do
    case "${flag}" in
        u) UDP=true ;;
        t) TCP=true ;;
        f) FULL=true ;;
        o) OUTPUT_FILE="${OPTARG}" ;;
        i) IP="${OPTARG}" ;;
        v) OS=true ;;
        a) ATTACK=true ;;
        d) DISCOVERY=true ;;
        m) MODE="${OPTARG}"; TCP=true ;;
        h|*) print_usage && exit 1 ;;
    esac
done

[ -z $IP ] && error "You must specify an ip address"

echo "" > $OUTPUT_FILE

```

```
[ $DISCOVERY = true ] && discovery


[ $OS = true ] && run_command "First scan and os version" "nmap -O -sV $IP"

[ $TCP = true ] && tcp_scan

[ $UDP = true ] && udp_scan

print_header "Scanning results:\n"
cat $OUTPUT_FILE | grep open | cut -d " " -f 1,2,3,4 | sort | uniq

[ $ATTACK = true ] && attack
```