



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	2
I Tools	3
1 Introduction	4
2 Wi-Fi access	5
3 Memory analysis	6
Bibliography	6
A Scripts	8

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 be 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 [6], 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 [6] this work is based it was suggested to use the Fern wifi cracker [3] to gain access to the target network. This 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
- Brute-force 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.

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

- Reaver [4]: a tool that exploits a flaw in the WPS implementation of some routers using the WPS Pixie-Dust and other brute-force attacks
- Wifite [5]: 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 [2]: 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.

3 Memory analysis

Bibliography

- [1] Aircrack-ng. <https://www.aircrack-ng.org/>. accessed 15/06/2023.
- [2] Bully. <https://github.com/aanarchy/bully>. accessed 15/06/2023.
- [3] Fern wifi cracker. <https://github.com/savio-code/fern-wifi-cracker>. accessed 15/06/2023.
- [4] Reaver. <https://code.google.com/archive/p/reaver-wps/>. accessed 15/06/2023.
- [5] Wifite. <https://github.com/derv82/wifite2>. accessed 15/06/2023.
- [6] Bringhenti Alessandro. A framework for the security analysis of iot devices, 2022.

Appendix A Scripts

```
1  #!/bin/bash
2
3  # Colors
4  RED='\033[0;31m'
5  GREEN='\033[0;32m'
6  NORMAL_COLOR='\033[0m'
7  OUTPUT_FILE="output"
8  IP=""
9  FULL=false
10 UDP=false
11 TCP=false
12 OS=false
13 MODE=""
14 DISCOVERY=false
15 ATTACK=false
16
17 # Helper functions
18 error(){
19     printf "${RED}$1"
20     exit 2
21 }
22
23 print_header(){
24     printf "${GREEN}-----\n"
25     printf "${GREEN}$1${NORMAL_COLOR}\n"
26 }
27
28 run_command(){
29     printf "\n\n" >> $OUTPUT_FILE
30     printf "${GREEN}-----\n"
31     printf "${GREEN}$1\n"
32     printf "${GREEN}***Executing command: $2 ${NORMAL_COLOR}\n"
33     $2 | tee -a $OUTPUT_FILE || error "Failed executing $1"
34 }
35
36 print_usage(){
37     printf "Usage: scan [OPTIONS]\n"
38     printf "\t-u\t\tperform udp scan\n"
39     printf "\t-a\t\tperform attack with default scripts\n"
40     printf "\t-t\t\tperform tcp scan\n"
41     printf "\t-f\t\tfull scans\n"
42     printf "\t-d\t\tdiscovery mode (only -i and -o relevant)\n"
43     printf "\t-v\t\tscan for OS version\n"
44     printf "\t-o FILENAME\tredirect output to FILENAME\n"
45     printf "\t-i IP\t\tscan specified IP\n"
46     printf "\t-m MODE\t\ttcp scan mode\n"
47 }
48
49
```

```

50 # Perform TCP scan
51 tcp_scan(){
52     if [ $FULL = true ]
53     then
54         # Full scan
55         run_command "Full TCP scan" "nmap -p- -s${MODE} ${IP}"
56     else
57         # Default scan
58         run_command "Default TCP scan" "nmap -s${MODE} ${IP}"
59     fi
60 }
61
62 # Perform UDP scan
63 udp_scan(){
64     if [ $FULL = true ]
65     then
66         # Perform UDP scan
67         run_command "Full UDP scan" "nmap -sU -p- $IP --max-rtt-timeout 20ms --max-retries
68             0"
69     else
70         # Perform UDP scan
71         run_command "Default UDP scan" "nmap -sU $IP --max-rtt-timeout 20ms --max-retries 0"
72     fi
73 }
74
75 # Discover hosts in the network
76 discovery(){
77     run_command "Scanning network" "nmap -sn $IP"
78     exit 0
79 }
80
81 attack(){
82     print_header "Starting to attack discovered ports"
83     PORTS=$(cat $OUTPUT_FILE | grep open | cut -d " " -f 1,2,3,4 | grep tcp | sort | uniq |
84         cut -d "/" -f 1)
85
86     for p in $PORTS
87     do
88         run_command "Testing port $p" "nmap -sC -p$p $IP"
89     done
90 }
91
92 while getopts 'avdhufm:o:i:' flag; do
93     case "${flag}" in
94         u) UDP=true ;;
95         t) TCP=true ;;
96         f) FULL=true ;;
97         o) OUTPUT_FILE="${OPTARG}" ;;
98         i) IP="${OPTARG}" ;;
99         v) OS=true ;;
100        a) ATTACK=true ;;
101        d) DISCOVERY=true ;;
102        m) MODE="${OPTARG}"; TCP=true ;;
103        h|*) print_usage && exit 1 ;;
104    esac
105 done
106
107 [ -z $IP ] && error "You must specify an ip address"
108
109 echo "" > $OUTPUT_FILE

```

```
109 [ $DISCOVERY = true ] && discovery
110
111
112 [ $OS = true ] && run_command "First scan and os version" "nmap -O -sV $IP"
113
114 [ $TCP = true ] && tcp_scan
115
116 [ $UDP = true ] && udp_scan
117
118 print_header "Scanning results:\n"
119 cat $OUTPUT_FILE | grep open | cut -d " " -f 1,2,3,4 | sort | uniq
120
121 [ $ATTACK = true ] && attack
```
