30% project proposal

UIAS

Unauthorized Internet Access System

Inbar Koursh

School: Ironi H, TLV

Mentors: Arie Klubaner, David Vinogradov

25th November, 2020

Introduction And Legal

This project proposal addresses an issue concerning the repetitiveness of running routine wireless testing attacks against a network. The program will be capable of deauthing network clients, cracking wpa passwords, and bypassing captive portals.

It will accomplish this using the aircrack-ng suite of tools (licenced under the GNU General Public License, version 2) and the macchanger tool (licenced under the GNU General Public License v3.0) , along with iwconfig (part of the Berkeley Software Distribution and licenced as such) and pkexec (I was unfortunately unable to find the license for, but because it is included natively in ubuntu, I can assume it is under the GNU license or a similar licence) to a lesser extent.

Note that this project (PUIAS) is not licensed under the GNU license but rather under the MIT license. Because it is licensed as such along with the nature of the GNU license, distribution of this software is **strictly forbidden** until such time that this repository becomes public or if the code is bundled with the source code (as done in this project to it's contributors).

Attack Options

This system will boast a total of three attack modes:

1. Captive portal bypass:

   In this mode the system will attempt to bypass the captive portal authentication (as can be found in airports, hotels, cafes etc).

2. Deauth mode:

   In this mode the system will attempt to deauth a target of your choosing or attempt to deauth all clients of a specific network.

3. Password cracking mode:

   In this mode the system will attempt to crack a network password.

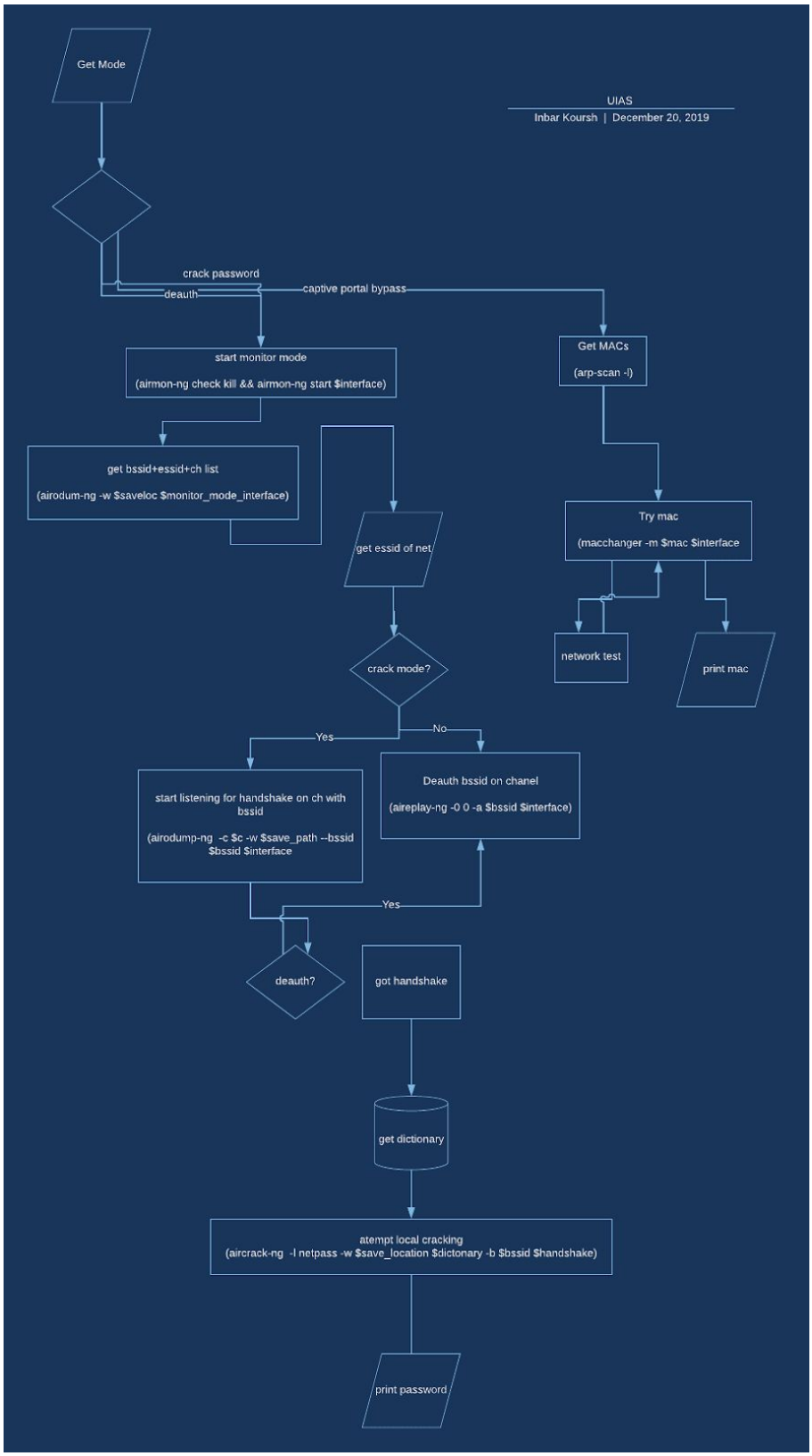
Attack execution

The aforementioned attacks will be carried out in the following ways:

1. The captive portal attack will work by spoofing an authenticated user. It will do so by spoofing the metric that is used commonly by captive portals to verify a users identity: their MAC address, this address that is inherent in all network devices can be both scanned and spoofed. The attack will proceed as follows:
    a. Scan the network for client mac addresses
    b. Spoof mac address
    c. Attempt to connect to the internet
    d. If failed goto step B
    e. You are now connected - quit program

2. The deauth attack will work by spoofing a deauthentication packet from the router, a packet that tells devices to disconnect from the router (**NOTE**: this attack is **illegal** to use without **permission** as it counts as a denial of service attack) the attack will work via the aireplay-ng tool

3. The password cracking option will operate by:
    a. Waiting for a client to connect (or deauth a client if you have permission)
    b. Capture the 4-way handshake exchanged by the connection
    c. The attempted cracking of said handshake using a pre-generated password list or a unique password list created by the network name as a seed


Get Mode
UIAS
Inbar Koursh | December 20, 2019
crack password
deauth
captive portal bypass
start monitor mode
(airmon-ng check kill && airmon-ng start $interface)
Get MACs
(arp-scan -l)
get bssid+essid+ch list
(airodum-ng -w $saveloc $monitor_mode_interface)
get essid of net
Try mac
(macchanger -m $mac $interface
network test
print mac
crack mode?
Yes
No
start listening for handshake on ch with bssid
(airodump-ng -c $c -w $save_path --bssid $bssid $interface
Deauth bssid on chanel
(aireplay-ng -0 0 -a $bssid $interface)
Yes
deauth?
got handshake
get dictionary
atempt local cracking
(aircrack-ng -l netpass -w $save_location $dictonary -b $bssid $handshake)
print password

Functions

The system uses a variety of functions:

1. static Arraylist<String> execute(String command, boolean sudo):

   Executes a command in linux (sudo or not) and returns an arraylist of the response (each entry is a line)

2. static boolean checkpackage(String packageS):

   Executes “which” command on the package name and checks the result to see if the program is installed

3. static ArrayList<String> change_mac(String mac):

   Changes the computer's MAC address to the requested mac by: taking down the wireless card, running “macchanger -m” on the requested address, and finally bringing the wireless card back up.

4. static boolean trymac(String mac):

   Runs change mac on the requested mac, trys 5 times to assert weather connection to network has been established, finally pings 8.8.8.8 (google dns server) and monitors packet loss, if packet loss is less than 100% returns true, if not or if ping failed, returns false

Data types

The program uses the following data types:

1. int
2. String
3. boolean
4. Arraylist<String>
5. Process
6. ProcessBuilder
7. InputStreamReader
8. BufferedReader
9. Exception
10. IOException
11. System
12. File
13. BufferedWriter
14. FileWriter
15. Object

Java Docs

This Github Repo also contains a javadoc file that goes into much further detail.