WLAN-AP mit regelmäßigem PSK-Tausch und QR-Code Anmeldung

Luca Asmus
Marius Würstle
Rolf Wiersch, 29837

Zusammenfassung

Das Ziel dieses Projekts war es, die Sicherheit im eigenen Gast-WLAN zu gewährleisten, unter Berücksichtigung der Faulheit und Bequemlichkeit vieler Endnutzer. Der pre-shared Key eines WLANs wird in den meisten Netzwerken einmal oder nie geändert. Dadurch können Gäste dauerhaften Zugang zum Netzwerk behalten, obwohl das nicht erwünscht ist. Ein weiteres Problem ist die Umständlichkeit einen sicheren pre-shared Key zu verwenden. Es führt zu unangenehmen Mehraufwand eine kryptische und lange Zeichenkette auf Endgeräten einzugeben. Gelöst wurde dies durch einen eigenen Access Point für Gäste. Über diesen wird der Zugriff ins Internet geleitet. Der pre-shared Key wird einmal die Woche oder manuell neu erzeugt und auf einem Display ausgegeben. Die Ausgabe erfolgt in Form eines QR - Codes und in Klartext.

December 11, 2020

Inhaltsverzeichnis

1 Motivation

Der Hauptgrund für dieses Projekt war es die Sicherheit im eigenen Gast-WLAN zu gewährleisten, unter Berücksichtigung der Faulheit und Bequemlichkeit vieler Endnutzer.

Wenn Gäste in der heimischen Wohnung auftauchen, ist der Wunsch nach freiem WLAN meist sehr groß. Bedeutet, der pre-shared key muss abgelesen und den Gästen bekannt gemacht werden. Folgend muss dieser umständlich von Hand eingegeben werden. Um hierbei Sicherheit zu gewährleisten, ist dieser meist länger und kryptisch gewählt. Dies führt oft zur falschen Eingabe bzw. Mehrversuchen und darauf folgenden Ärger darüber. Weiterhin ist es vom Gastgeber nicht immer erwünscht, dass die Gäste nach der Verabschiedung den Zugriff zum WLAN behalten.

Da die geschilderten Umstände den Verfassern dieses Dokumentes nicht fremd sind, soll mit diesem Projekt eine eigenständige Lösung erstellt werden. Der Fokus liegt auf einfacher Bedienung und komfortabler Sicherheit. Weiterhin wird für die Sicherheit auf fertige Endprodukte von Drittanbietern verzichtet.

1.1 Fachbegriffe –> kommt ins glossary

host access point daemon = hostapd sed = Stream EDitor, Unix-Werkzeug zum Bearbeiten von Text stdout = Standard Ausgabe, normalerweise mit Monitor verbunden

2 Grundlagen Hard- und Software

In den folgenden Abschnitten werden Hardware und Software genauer erläutert. Im Glossar am Ende der Projektarbeit können weitere Erklärungen bzw. Akronyme entnommen werden.

2.1 Hardware

2.1.1 Raspberry Pi

Der Raspberry Pi wurde für junge Menschen entwickelt, um ihnen eine preisgünstige Möglichkeit zu bieten, sich mit der Informatik zu beschäftigen. Der Einplatinencomputer ist etwa kreditkartengroß und kam Anfang 2012 auf den Markt. Er ermöglicht einen schnellen und praktischen Weg um Wissen in den Bereichen Programmieren und Hardware zu erlangen. Zudem ist er vielseitig einsetzbar.

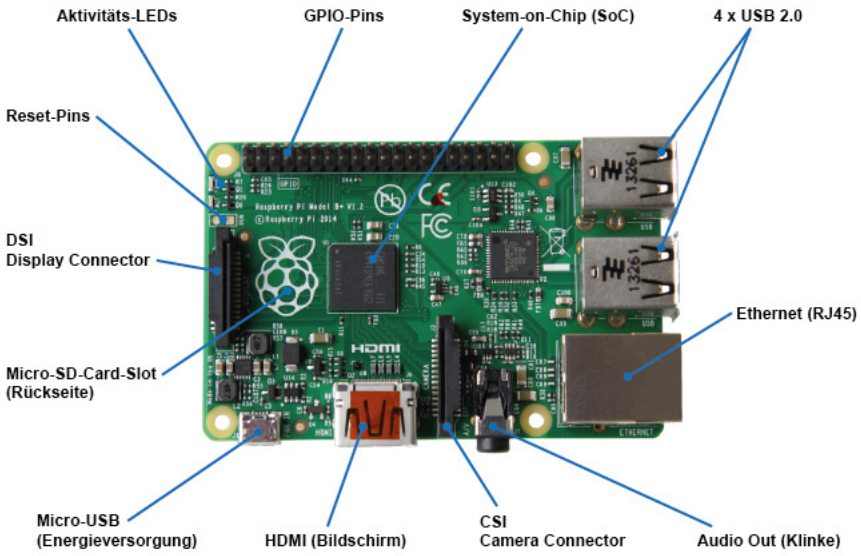

Abbildung 1: Raspberry Pi 3b - Quelle: [3]

Technische Spezifikationen des Raspberry Pi 3b:

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
- 1GB RAM
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
- 100 Base Ethernet
- 40-pin extended GPIO
- 4 USB 2 ports
- 4 Pole stereo output and composite video port
- Full size HDMI
- CSI camera port for connecting a Raspberry Pi camera
- DSI display port for connecting a Raspberry Pi touchscreen display
- Micro SD port for loading your operating system and storing data
- Upgraded switched Micro USB power source up to 2.5A [4]

2.1.2 Raspberry Pi Shield - Display LCD-Touch, 3,2in

Der Touchscreen wertet den Raspberry Pi zu einem vollwertigen Touch-PC auf. Für zusätzliche Funktionen besitzt der Display 3 Buttons an der Seite, welche einfach über die GPIO Pins eingelesen werden können.

Abbildung 2: Touchscreen Display für den Raspberry Pi - Quelle: [11]

Technische Spezifikationen unseres Raspberry Pi Shield - Display LCD-Touch, 3.2in:

- Display 8,13cm (3,2")
- Auflösung 320 x 240 Pixel
- LED-Hintergrundbeleuchtung
- 3 frei belegbare Taster (angebunden an GPIO12, 16, 18)
- SPI-Schnittstelle
- Touchscreen Technologie resistiv

2.2 Software

2.2.1 balenaEtcher

Programm mit dem ein OS auf eine SD Karte geflasht werden kann.

2.2.2 hostapd

Mit ist es möglich Geräte, die ein WLAN-Modul besitzen, als Access Point zu betreiben. Jedoch können keine Einstellungen im Bereich IP und Routing vorgenommen werden. Die Software ist nur für das Erstellen eines "wireless Ethernet switches" zuständig. [5]

2.2.3 dnsmasq

Geräte in einem Netzwerk benötigen zur Kommunikation eine IP Adresse und einen DNS Server für die Namensauflösung. Deshalb muss in diesem Projekt ein DHCP und DNS erstellt werden. Von diesen bekommen die Endgeräte ihre IP Konfiguration im WLAN. Die Software dnsmasq wird in diesem Projekt verwendet, um dies zu ermöglichen.

2.2.4 netfilter-persistent und iptables-persistent

Für die Durchführung des Projektes ist es nötig iptables-Regeln anzulegen. Diese sollten nach einem Neustart nicht neu angelegt werden müssen. Deshalb wurden die Pakete netfilter-persistent und iptables-persistent installiert. Damit können die Regeln in eine Datei abgespeichert und beim Neustart automatisch geladen werden.

2.2.5 Bash

Im Projekt wird Bash benutzt um die einzelnen Python Skripte aufzurufen, Infos aus Konfigurationsdateien auszulesen und schnelle Änderungen an Diesen vorzunehmen. Bash ist als Standardshell bei Raspberry Pi OS Lite vorinstalliert.

2.2.6 Cron

Cron ermöglicht das zeitbasierte Ausführen des Bash Skripts. So kann beispielsweise jeden Montag um 03:00 Uhr nachts das Passwort automatisch geändert werden.

2.2.7 fim

fim ermöglicht es Bilder oder andere Media Dateien im Terminal anzuzeigen. In diesem Fall wird der QR-Code als .png erzeugt und mithilfe von fim angezeigt. [10]

2.2.8 Python 3.7

Für die Skripte zur Passwortgenerierung, QR-Code Generierung und zum Einlesen der Buttons wird die Sprache Python verwendet. Python 3.7 ist bei Raspberry Pi OS Lite vorinstalliert und erleichtert durch verschiedene Bibliotheken die Umsetzung des Projektes.

2.2.9 pyqrcode

Das Modul pyqrcode wird dafür benutzt, möglichst einfach und frei QR-Codes zu erzeugen. Zum Erzeugen des Codes benötigt sie nur die Parameter die enthalten sein sollen. [9]

2.2.10 GPIO zero

Das Modul GPIO zero wird in diesem Projekt für das Einlesen der einzelnen Buttons genutzt, die bereits in dem Display integriert sind. Das Mapping der Buttons wird auch in der Anleitung des Displays erklärt. (Button1 = PIN12/GPIO18) [1]

3 Problemstellung

Vielen Endnutzern ist die eigene Bequemlichkeit sehr wichtig. Daraus resultiert, dass der pre-shared Key eines WLANs in den meisten Netzwerken einmal oder nie geändert wird. Dadurchk önnen Gäste dauerhaften Zugang zum Netzwerk behalten, obwohl das nicht erwünscht ist. Dies hat zur Folge, dass sich sobald einmal das Passwort bewusst/unbewusst weitergegeben wurde, jeder mit dem Access-Point verbinden kann. Negative Auswirkungen könnten sich äußern in Form von Missbrauch des Internetzuganges oder durch Angriffe auf das interne Netzwerk.Ein weiteres Problem ist die Umständlichkeit einen sicheren pre-shared Key zu verwenden. Es führt zu unangenehmen Mehraufwand eine kryptische und lange Zeichenkette auf Endgeräten einzugeben bzw. zu merken, den viele Endnutzer nicht eingehen wollen. Deshalb werden oft einfache pre-sharedKeys in Form von z.B Wortkombinationen verwendet. Dies hat zur Folge,dass der Zugang einfacher geknackt werden kann.

4 Anforderungsanalyse –> Ausformulieren

Im Folgenden werden die verschiedenen Funktionen definiert und kategorisiert.

4.1 Funktionale Anforderungen

4.1.1 Funktionaler Access-Point

4.1.2 Internet Zugriff

DHCP Server DNS Server Firewall

4.1.3 Sicherer Key

4.1.4 Automatisches Wechseln des Key

4.1.5 Ausgabe des Key

4.2 Nichtfunktionale Anforderungen

4.2.1 Manueller Wechsels des Key

4.2.2 Energiespar-Konfiguration

4.2.3 Automatisierungs-Skript

4.3 Priorisierung

Das wichtigste zu Beginn ist, dass die Hardware aufeinander abgestimmt wird. Bedeutet, alle Teile passen zusammen und können angeschlossen werden. Danach muss die Konfiguration des Raspberry Pi zum Access-Point erfolgen. Sobald dies funktioniert kann das Generieren des pre-shared Keys und dessen Austausch stattfinden. Folglich wird die Ausgabe des Keys am Bildschirm realisiert. Danach kann dies, um das Generieren des QR-Codes und dessen Ausgabe erweitert werden.
Wenn diese Punkte voll funktional umgesetzt werden konnten, kann sich um einen automatischen oder manuellen Job (bei Tastendruck) zum Generieren und Austauschen des Keys gekümmert werden.

5 Lösungsidee

Ein Lösungsansatz stellt ein sich automatisch oder auf Tastendruck änderbarer pre-shared Key dar. Der Key wird jeden Montagmorgen um 03:00 Uhr automatisch durch einen "Cron-Job" gewechselt. Ein Taster kann zusätzlich betätigt werden, falls das Passwort sofort geändert werden soll.
Der Key wird so gewählt, dass er aus mathematischer Sicht nicht in dem Zeitraum geknackt werden kann, bis automatisch ein neuer erzeugt wird. Durch diese Maßnahme wird die Sicherheit des WLANs verbessert und sichergestellt. Mit diesem neu generierten Key können sich Endgeräte anmelden und kommunizieren. Hiermit wird das Problem der ungewollten Nutzern gelöst. Diese können sich sobald der Key gewechselt hat, nicht mehr im Netz anmelden. Der Key wird an einem Display in zwei verschiedenen Varianten angezeigt:

- Klartext für Endgeräte ohne Kamera z.B. Laptops
- QR-Code zum Scannen für z.B Smartphones

Es wurde sich für einen QR-Code anstatt eines RFID/NFC Transponder entschieden, da hier die Möglichkeit des Abgreifens des Keys nicht besteht. In Abbildung 3 ist der Aufbau des Lösungsansatzes graphisch dargestellt.

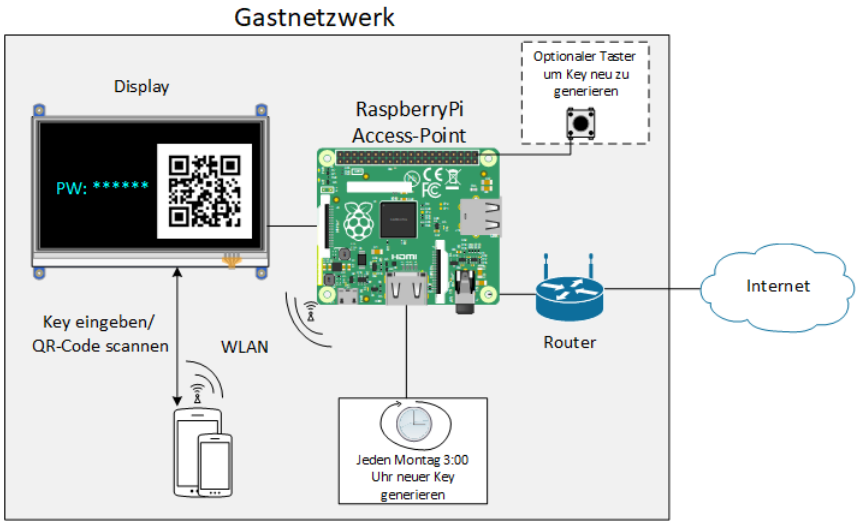

Abbildung 3: Aufbau des Gastnetzes

Es ist zu erwähnen, dass sich diese Lösung auf die Umsetzung eines Gastnetzes bezieht. Dieses ist getrennt vom restlichen Heimnetz. Es melden sich dort hauptsächlich mobile Geräte an. Dies führt zu einer hohen Fluktuation an Endgeräten im Netzwerk und ist nicht für stationäre Geräte wie z.B Drucker geeignet.

5.1 Hardware

Für dieses Projekt wird ein Raspberry Pi 3 B benötigt. Grund hierfür ist das integrierte WLAN-Modul, ausreichend Leistung und alle nötigen Anschlüsse. Zusätzlich wird eine Mikro-SD Karte zum Laden des Betriebsystems und zur Speicherung der Daten benötigt.

Das Display, welches verwendet wird, muss eine ausreichende Auflösung für die Darstellung des QR-Codes besitzen. Deshalb können keine kleineren und billigeren LCD Anzeigen verwendet werden. Außerdem muss es die Möglichkeit haben entweder externe Taster anzubinden oder schon eigene Taster besitzen.

5.2 Software

Um den Raspberry Pi als Access-Point verwenden zu können, müssen zusätzliche Pakete installiert (hostapd) und eine Netzwerkkonfiguration vorgenommen werden. Darunter fällt das Einstellen von DHCP und DNS für die Clients durch "dnsmasq" und Anpassen der IP-Konfiguration.

Als Skriptsprache empfiehlt sich Python, da es dort sehr viele Libraries gibt, die einiges an Arbeit abnehmen. Zudem kann mit Python auf einfache Kommandos des Betriebssystems zugegriffen werden. Für die QR-Code Generierung eignen sich die Libraries "qrcode" und "PyQRCode".

5.3 Passwortkonzept

Als Passwortkonzept wurde sich auf einen 12 Zeichen langen Key geeinigt. Dieser benützt 62 Zeichen in Form von Groß-, Kleinbuchstaben und Zahlen. Begründet wurde diese Entscheidung mit folgender Annahme:

Angenommen ein leistungsstarker Rechner schafft durch Brute-Force 2 Billionen Keys pro Sekunde, so würde er 18670 Tage benötigen, um alle Keys zu testen. Wenn schon zur Hälfte der Zeit der richtigen Key gefunden wurde, wäre dies immernoch mehr als ausreichend für eine Woche. Zu sehen ist dies in der folgenden Rechnung:

$$(62^{12})\ keys\ \div\ 2000000000000\ \frac{keys}{s} = 1613133381.198\ s \tag{1}$$

$$1613133381.198949911\ s\ \div\ 60\ \div\ 60\ \div\ 24\ \approx 18670\ Tage \tag{2}$$

$$18670\ Tage\ \div\ 2\ =\ 9335\ Tage \tag{3}$$

So wäre die Sicherheit des Key gewährleistet.

5.4 Tests

Zu Testzwecken werden unterschiedliche Smartphones (Android/iOS) und Notebooks (Windows/Linux/macOS) benutzt. So soll sichergestellt werden, dass mögliche Probleme aufgrund von Diskrepanzen zwischen den Betriebssystemen bzw. Hardware erkannt werden.

6 Bewertung der Lösung anhand der Anforderungen

7 Implementierung

7.1 Vorbereitung des Raspberry Pi

7.1.1 Auswahl und Installation des Betriebssystem

Um mit dem Projekt beginnen zu können musste zuerst ein Betriebssystem bestimmt werden. Es wurde sich für das Raspberry Pi OS Lite entschieden. Begründet wurde diese Entscheidung durch die weniger vorinstallierten Pakete

und einer fehlender grafischen Bedienoberfläche. Hierdurch konnte Speicherplatz und Sicherheitsrisiken eingespart werden. Je weniger unbenutzte Software, desto weniger Angriffsfläche.

Nach der Auswahl des Betriebssystems konnte dieses auf eine SD-Karte geschrieben werden. Hierzu wurde die Software balenaEtcher verwendet.

7.1.2 SSH Zugriff einrichten

Da das Projektteam aus drei Personen besteht, wurde ein SSH Zugriff in den Einstellungen des Raspberry Pi eingerichtet. Die Einstellungen können mit folgendem Befehl geöffnet werden:

```
1 sudo raspi-config
```

In diesem Zuge wurde der SSH Zugriff aktiviert und das Standardpasswort geändert. Durch den Zugriff konnte das parallele Arbeiten am Projekt ermöglicht werden.

7.1.3 Aktualisierung und Paketinstallation

Nach der Neuinstallation eines Betriebssystems fehlen diesem oft die aktuellsten Versionen von Softwarepaketen und Updates. Deshalb wurden diese zuerst aktualisiert und installiert. So werden Konflikte aufgrund veralteter Software vermieden und die Sicherheit verbessert. Darauf folgte das Nachinstallieren der für das Projekt noch benötigten Pakete. Diese wurden im Abschnitt Software genauer beschrieben.

7.2 Konfiguration des RaspberryPi als funktionalen Access-Point

7.2.1 WLAN Interface

Der Raspberry Pi benötigt eine statische IP Konfiguration für sein WLAN Interface. Diese wird in der Datei /etc/dhcpcd.conf vorgenommen. Die Datei wird um folgendes ergänzt:

```
1 interface wlan0
2     static ip_address=192.168.4.1/24
3     nohook wpa_supplicant
```

Mit der 192.168.4.1 wird eine statische IP Adresse vergeben unter die der Raspberry Pi im WLAN erreichbar ist. Weiterhin wird der wpa_supplicant deaktiviert um keine Konflikte mit hostapd zu verursachen.

7.2.2 Routing

Der Access Point muss den Datenverkehr der Endgeräte im WLAN zum Router weiterleiten können. Hierzu wird in /etc/sysctl.d/routed-ap.conf ein Eintrag hinzugefügt bzw. das Kommentarzeichen entfernt:

```
1 # Enable IPv4 routing
2 net.ipv4.ip_forward=1
```

Endgeräte können nun den Hauptrouter erreichen. Um jedoch eine Kommunikation zu ermöglichen muss NAT eingestellt werden. Die wird durch einen Eintrag in die iptables Firewall erreicht:

```
1 sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Bei Datenverkehr zum Hauptrouter wird nun die Absender IP Adresse der Endgeräte mit der IP der LAN-Schnittstelle ersetzt. Bei Rückantworten an den Raspberry Pi werden diese an den jeweiligen Absender richtig weitergeleitet.

Um die Firewall Regel bei einem Neustart zu behalten, wurde diese abgespeichert:

```
1 sudo netfilter-persistent save
```

7.2.3 DNS und DHCP

Durch dnsmasq können nun die DHCP und DNS Einstellungen erfolgen. Diese werden in der /etc/dnsmasq.conf Datei vorgenommen. Diese dient als Vorlage und gibt Erklärungen zu den Einstellungen. Zur Übersichtlichkeit wurde diese in dnsmasq.conf.orig umbenannt und eine neue Datei mit dem ürsprünglichen Namen erzeugt. In der neuen Datei werden nur die getätigten Konfigurationen eingetragen:

```
1 interface=wlan0 # Listening interface
2 dhcp-range=192.168.4.2,192.168.4.20,255.255.255.0,24h
3                 # Pool of IP addresses served via DHCP
4 domain=wlan     # Local wireless DNS domain
5 address=/gw.wlan/192.168.4.1
6                 # Alias for this router
```

Zuerst wird das Interface angeben, bei den die DHCP/DNS Konfiguration gelten soll. Die ist das schon vorherig erstellte Interface "wlan0". Es wurde sich auf einen DHCP Bereich von 192.168.4.2/24 - 192.168.4.20/24 entschieden. Dieser umfasst 18 IP Adressen, welcher als ausreichend für eine Woche angesehen wird. Die Lease - Zeit wurde auf 24 Stunden eingestellt, da Gäste meist nicht länger als einen Tag anwesend sind. Zuletzt wurde eine lokale

DNS Domäne und ein Alias für den Access Point eingestellt. Unter diesem Alias ist dieser nun erreichbar.

7.2.4 Access Point Einstellungen

Um den Raspberry Pi als Access Point nutzen zu können musste nun hostapd konfiguriert werden. Hierzu wurde zuerst der Dienst aktiviert und so eingestellt das er beim booten gestartet wird:

```
1  sudo systemctl unmask hostapd
2  sudo systemctl enable hostapd
```

Nun musste die Konfigurationsdatei unter /etc/hostapd/hostapd.conf erstellt und gefüllt werden. In dieser werden verschiedene Parameter eingestellt. Darunter fallen unter anderem die SSID, das Passwort und die Art der Verschlüsselung. Es wurde eingestellt das nur WPA2 verwendet wird, da WEP als unsicher gilt. Weiterhin wurde der Funkstandard auf n und 2.4GHz eingestellt. Grund hiefür war das der Raspberry Pi keinen höheren Standard in Form von z.B AC unterstützen würde. Weiterhin ist der eingestellte Standard ausreichend für das surfen im Gast-Internet.
Der Kanal wurde fest auf Sechs gesetzt. Eine passende Kanalsuche mittels ACS kann nicht erfolgen. Die Implementierung von ACS in Hostapd wird nur von bestimmten Atheros Treibern unterstützt [2]. Der Raspberry Pi besitzt onboard jedoch nur einen Broadcom-Chip [4] und ist somit nicht kompatibel.
Um Wireless-Networking auf dem Raspberry Pi zu ermöglichen muss ein "Country Code" gesetzt werden. In diesem Fall auf "DE", welches Deutschland entspricht [6]. Dieser ist notwendig, denn je nach Land sind die Frequenzbänder unterschiedlich vergeben bzw. reguliert. Im folgenden der Inhalt der Konfigurationsdatei:

```
1  country_code=DE
2  interface=wlan0
3  ssid=HimbeerWLAN
4  hw_mode=g
5  ieee80211n=1
6  channel=6
7  macaddr_acl=0
8  auth_algs=1
9  ignore_broadcast_ssid=0
10 wpa=2
11 wpa_passphrase=GeneratePW
12 wpa_key_mgmt=WPA-PSK
13 wpa_pairwise=TKIP
14 rsn_pairwise=CCMP
```

Nach den Einstellungen erfolgt ein Reboot und der Access Point ist nun einsatzbereit.

7.3 Passwortgenerierung

Die Passwortgenerierung wird mithilfe eines Python Skripts gelöst. Dieses ist in unserem GitHub Repository hinterlegt und für jeden zugänglich. Das Skript verwendet die zwei Imports string und secrets. Mithilfe der Bibliothek string können die für Bash problematischen Zeichen aus dem Alphabet entfernt werden. Das secrets Modul wird für das Generieren von stark kryptographischen Passwörter verwendet. Die verwendete Funktion secrets.choice wählt aus der mitgelieferten Sequenz ein zufälliges Zeichen aus. Welches anschließend an den schon vorhandenen String angehängt wird. Dies wird 10 mal wiederholt.

```
1  import secrets
2  import string
3
4  def get_random_password():
5          temp = string.ascii_letters + string.
               ↪ digits + string.punctuation
6          alphabet = temp.replace('\'', '').replace(
               ↪ '\\', '').replace('\"','').replace('
               ↪ \`', '').replace(';', '')
7          password = ''.join(secrets.choice(alphabet
               ↪ ) for i in range(10))
8          return password
9
10 if __name__ == "__main__":
11         print(get_random_password())
```

7.4 Passworttausch

Das Tauschen des Passworts wird durch ein Bash Skript, mit dem Namen changePassword.sh, vorgenommen. Zunächst wird das neue Passwort mit Hilfe von generatePassword.py generiert und der Typ der Verschlüsselung sowie die SSID des Access Point aus der hostapd.conf Datei gelesen. Die drei Parameter werden einmal als Klartext ausgegeben und dann werden sie dem Python Skript zum Generieren des QR-Codes übergeben. Im Anschluss wird mit dem Unix Tool sed die Zeile der hostapd.conf angepasst, welche das Passwort enthält. Mit der Option -i nimmt sed die Änderung direkt an der gegebenen Datei vor statt nur zu stdout zu schreiben. Damit der Tausch in Kraft tritt muss der hostapd Service neu gestartet werden.
Die zugehörigen Python Dateien müssen sich im selben Ordner wie das Bash Skript befinden. Das Skript bezieht sich auf den Pfad an dem es liegt und nicht den aktiven Pfad, damit es von überall ausführbar ist und keine Probleme mit Cron auftreten.

```
#!/usr/bin/env bash

readonly SCRIPT="$(test -L "${BASH_SOURCE[0]}" && readlink "${BASH_SOURCE[0]}" || echo "${BASH_SOURCE[0]}")"
readonly SCRIPT_DIR="$(cd "$(dirname "${SCRIPT}")"; pwd)"

execute_script() {
        # Get Access Point Parameter
        local pass=$(python3 "${SCRIPT_DIR}/generateKey.py")
        local wpa=$(grep /etc/hostapd/hostapd.conf -e wpa | cut -f 2 -d '=' | head -n 1)
        local ssid=$(grep /etc/hostapd/hostapd.conf -e ssid | cut -f 2 -d '=' | head -n 1)
        echo "WPA-Type:${wpa} Ssid:${ssid} Passphrase:${pass}"

        # Generate QR-Code
        python3 "${SCRIPT_DIR}/qrCodeGenerator.py" "${ssid}" "WPA${wpa}" "${pass}"

        # Change the Password and restart Access Point
        sed -i "s/wpa_passphrase=.*/wpa_passphrase=${pass}/g" \
                "/etc/hostapd/hostapd.conf"
        systemctl restart hostapd.service
}

# main
if [[ "${BASH_SOURCE[0]}" != "$0" ]]; then
        echo "Script is being sourced"
else
        set -x
        set -euo pipefail
        execute_script "$@"
fi
```

7.4.1 Automischer Tausch

Cron erlaubt es, das Passwort regelmäßig zu einer definierten Zeit, hier beispielsweise Montags um 03:00 Uhr nachts, zu tauschen. Nachts bietet sich an, da zu dieser Zeit für gewöhnlich das Netz kaum bis nicht genutzt wird. Damit das Passwort und der QR-Code dennoch auf dem angeschlossenen Bildschirm angezeigt werden ist es wichtig beim Aufrufen des Skripts die Standardausgabe das korrekte Gerät umzulenken. In diesem Fall wird stdout auf /dev/tty1 umgelenkt. Da manche Änderungen root-Rechte benötigen, wird der Aufruf in der crontab Datei des root Nutzers definiert.

```
* 3 * * 1 /home/pi/scripts/changePassword.sh > /dev/tty1
```

8 Einrichten des Displays

Die Einrichtung des Displays wurde nach der mitgelieferten Anleitung durchgeführt. [7] Zusätzlich wurde auf einem anderen Raspberry Pi das Beispiel Image installiert und mit dem vorliegenden verglichen. Zunächst musste

die /boot/config.txt so verändert werden, dass die Ausgabe mit dem richtigen Treiber auf dem Display erscheint. Folgende Konfigurationen sind zu machen:

```
dtparam=audio=on
dtparam=spi=on
dtoverlay=joy-IT-Display-Driver-32b-overlay:rotate=270,swapxy=1

hdmi_ignore_edid=0xa5000080
hdmi_force_hotplug=1
```

Anschließend wird der Treiber auf die passende Konsole gemapped. Hierzu fügt man an das Ende der ersten Zeile in /boot/cmdline.txt die Knofiguration:

```
fbcon=map:10
```

Natürlich muss man den Treiber auch noch herunterladen, auspacken und nach /boot/overlays verschieben. Über diesen Link kann das Paket heruntergeladen werden:

"http://joy-it.net/files/files/Produkte/RB-TFT3.2-V2/joy-IT-Display-Driver-32b-overlay.zip"

Jetzt muss man die Datei 99-calibration.conf in /usr/X11/xor.conf-d/ mit folgenden Einstellungen erstellen:

```
Section "InputClass"
        Identifier "calibration"
        MatchProduct "ADS7846␣Touchscreen"
        Option "Calibration" "189␣3767␣3842␣249"
        Option "SwapAxes" "0"
EndSection
```

In diesem Projekt wird das Raspberry OS Lite verwendet, weil in diesem Betriebssystem sehr wenige Pakete vorinstalliert sind, muss für den Display noch zusätzlich xserver-xorg-video-fbturbo installiert werden.
Anschließend muss für die Touch-Funktion die Datei /usr/share/X11/xorg.conf.d/10-evdev.conf in das selbe Verzeichnis kopiert und in 45-evdev.conf umbennant werden. Zum Schluß muss auch für die Touch-Funktion das Packet server-xorg-input-evdev installiert und der Raspberry Pi neugestartet werden.

9 QR-Code Generierung

Das Skript qrCodeGenerator.py wird über changePassword aufgerufen und bekommt 3 Argumente die SSID des Netzwerks, WPA Einstellung und das Passwort. Diese werden ausgelesen und in die pyqrcode.create als String mitgegeben. Das Format des Strings ist sehr wichtig, denn so wird definiert wie das Handy den QR-Code zu interpretieren hat. Am Ende wird der Code

mit einem print Statement auch auf die Konsole ausgegeben.

```
1         import sys
2         import pyqrcode as pqr
3
4         def create_qr_code(ssid, security,
              ↪ password):
5         qr = pqr.create(
6         'WIFI:S:{ssid};T:{security};P:{password};;
              ↪ '.format(ssid=ssid, security=security
              ↪ , password=password))
7         print(qr.terminal())
8
9         if __name__ == "__main__":
10        ssid = sys.argv[1]
11        security = sys.argv[2]
12        password = sys.argv[3]
13        create_qr_code(ssid, security, password)
```

10 Ausgabe des Passworts

Bei der Erstellung des Passworts wird es bereits im Format: "SSID: <ssid> WPA<version> PW:<passwort>" als Plaintext auf die Konsole geschrieben.

10.1 Tastenbelegung

Damit der Knopfdruck erkannt wird, gibt es in dem GitHub Repository [8] das buttonInput.py Skript. Mithilfe von GPIO zero kann man einfach die Buttons auf die gewünschten GPIOs hören und reagieren.
Button1 zeigt den QR-Code an, wie das gemacht wird, erklärt das nächste Kapitel.
Button2 wird dafür verwendet ein neues Passwort zu generieren. Dort wird einfach unser changePassword.py aufgerufen.
Button3 führt ein "clear" aus somit kann der Display mal wieder aufgeräumt werden, falls zuviele Passwörter dortstehen.

10.2 Anzeigen des QR-Codes

Damit der QR-Code angezeigt wird muss der erste Knopf an dem Display gedrückt werden. Dies hat den Grund, dass der QR-Code eine .png Datei ist und er damit nicht einfach auf dem Terminal ausgegeben werden kann. Ein weiterer Grund ist, dass der QR-Code den Plaintext verdrängen würde und dann nicht mehr sichtbar wäre.

Der erste Button wird für das Anzeigen des QR-Codes verwendet. Dort wird ein Subprocess erstellt und damit sobald der Childprozess getötet wird, nicht der Parent Process mitstirbt das Kommando mit dem Parameter preexec_fn=os.setsid vorher ausgeführt. 15 Sekunden später wird die ganze Prozess Gruppe getötet. Mit dem gesendeten Signal SIGTERM wird die fim Ansicht beendet und man kehrt zur Shell zurück.

```
1  #!/usr/bin/env python3
2
3  from gpiozero import Button
4  from time import sleep
5  import subprocess
6  import os
7  import signal
8
9  key1 = Button(18)
10 key2 = Button(23)
11 key3 = Button(24)
12
13 while True:
14         if key1.is_pressed:
15                 p = subprocess.Popen('exec␣fim␣/
                       ↪ home/pi/qrcode.png', shell=
                       ↪ True, preexec_fn=os.setsid)
16                 sleep(15)
17                 os.killpg(os.getpgid(p.pid),
                       ↪ signal.SIGTERM)
18         if key2.is_pressed:
19                 subprocess.run(['changePassword.py
                       ↪ '])
20         if key3.is_pressed:
21                 os.system('clear')
```

11 Fazit und Ausblick

11.1 Fazit

Die hier vorgestellte Implementierung war im großen und ganzen sehr erfolgreich. Der Raspberry Pi kann als Access Point verwendet werden und wechselt seinen Schlüssel automatisch. Dieser wird dann auf dem Display angezeigt, also sind die angegebenen Pflichtfunktionen enthalten.
Zusätzlich kann das Passwort auch mit einem Knopfdruck zu einer beliebigen

Zeit gewechselt werden. Mit dem erstellten Setup Skript wurde das Nachmachen dieses Projekt sehr vereinfacht. Die interessierten müssen nur die im Repository [8] gezeigten Schritte durchführen und das Skript ausführen und schon haben sie einen funktionierenden Access Point.
Allerdings gibt es auch noch einzelne Probleme zum Beispiel können die Gäste vom Gastnetz auf das Heimnetz zugreifen. Zudem liegtdie hostapd.conf Datei ungeschützt auf dem Raspberry Pi und beinhaltet das Passwort.

11.2 Ausblick

11.3 eventuell übertragbarkeit

12 Abbildungsverzeichnis

List of Figures

13 Quellenverzeichnis

References

[1] Ben Nuttall, 2020. https://gpiozero.readthedocs.io/en/master/index.html [aufgerufen am 11.12.2020].

[2] Chaitanya T K, 2020. https://wireless.wiki.kernel.org/en/users/documentation/acs [aufgerufen am 10.12.2020].

[3] Elektronik-kompendium, 2020. http://www.elektronik-kompendium.de/sites/raspberry-pi/bilder/19052512.jpg [aufgerufen am 25.11.2020].

[4] Elektronik-kompendium, 2020. http://www.elektronik-kompendium.de/sites/raspberry-pi/2102291.htm [aufgerufen am 25.11.2020].

[5] Gentoo Foundation, Inc., 2020. https://wiki.gentoo.org/wiki/Hostapd#Capabilities_of_Hostapd [aufgerufen am 26.11.2020].

[6] International Organization for Standardization, 2020. https://www.iso.org/obp/ui/#iso:code:3166:DE [aufgerufen am 11.12.2020].

[7] joy-it, 2020. https://joy-it.net/files/files/Produkte/RB-TFT3.2-V2/RB-TFT-Manual_04082020.pdf [aufgerufen am 11.12.2020].

[8] Luca Asmus, Rolf Wiersch, Marius Würstle, 2020. https://github.com/1xca/QRRaspAP [aufgerufen am 11.12.2020].

[9] Michael Nooner, 2020. https://pythonhosted.org/PyQRCode/ [aufgerufen am 12.12.2020].

[10] Michele Martone, 2020. https://www.unix.com/man-page/debian/1/fim/ [aufgerufen am 12.12.2020].

[11] Reichelt, 2020. https://cdn-reichelt.de/bilder/web/artikel_ws/A300/TFTV2.jpg [aufgerufen am 26.11.2020].