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

Luca Asmus
Marius Würstle
Rolf Wiersch

November 27, 2020

1 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.

Contents

2 Abbildungsverzeichnis

List of Figures

3 Allgemeines

3.1 Fachbegriffe

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

4 Hardware

4.1 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, in diesem Fall wird er zu einem Access-Point konfiguriert.

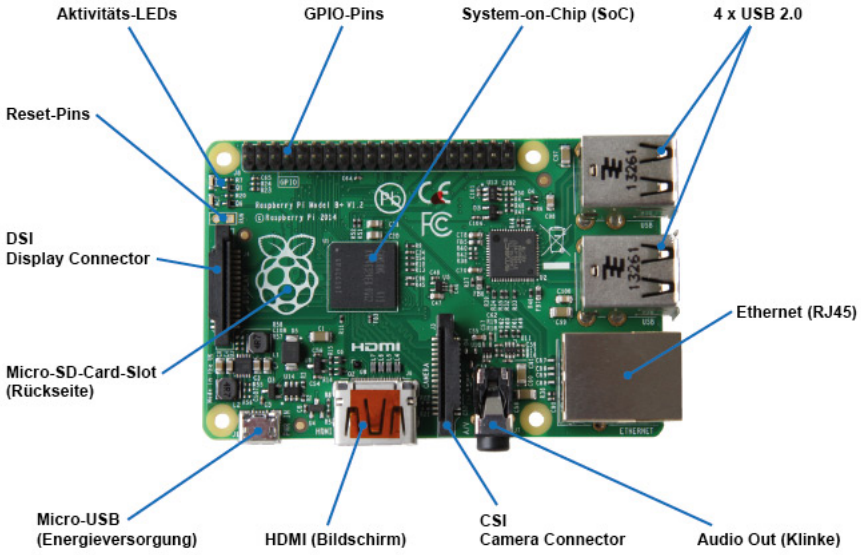

Abbildung 1: Raspberry Pi 3b - Quelle: [?]

Technische Spezifikationen unseres 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 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: [?]

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

4.3 SD-Karte

Die SD-Karte ist eine SanDisk extreme mit einer Speicherkapazität von 32GB. Sie dient als Speichermedium des Raspberry Pi's. Zu Beginn wird das Betriebssystem auf die Karte geflasht von dieser wird der Einplatinencomputer gebooted.

5 Software

5.1 balenaEtcher

Flash Sd Card

5.2 hostapd

Mit hostapd 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. [**?**]

5.3 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.

5.3.1 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.

5.4 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.

5.5 Cron

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

5.6 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.

5.6.1 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.

5.6.2 gpiozero

Im Beta-Stand noch nicht umgesetzt

6 Vorbereitung des Raspberry Pi

6.1 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.

6.2 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.

6.3 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 Konfiguration des RaspberryPi als funktionalen Access-Point

7.1 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 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
```

7.3 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 nsmasq.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.

7.4 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:

```
sudo systemctl unmask hostapd
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 auf null gesetzt. Mit dieser Einstellung sucht der Access Point automatisch einen passenden Kanal. Bedeutet, den mit den wenigsten Störungen bzw. einen Kanal der sich möglichst wenig mit anderen Netzen überschneidet.
Um Wireless-Networking auf dem Raspberry Pi zu ermöglichen muss ein "Country Code" gesetzt werden. In diesem Fall auf "DE", welches Deutschland entspricht. Die ist notwendig, denn je nach Land die Frequenzbänder unterschiedlich vergeben bzw. reguliert sind. Im folgenden der Inhalt der Konfigurationsdatei:

```
country_code=DE
interface=wlan0
ssid=HimberrWLAN
hw_mode=g
ieee80211n=1
channel=0
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=GeneratePW
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

Nach den Einstellungen erfolgte ein Reboot und der Access Point war nun einsatzbereit.

8 Passwortgenerierung

Die Passwortgenerierung wird mithilfe eines Python Skripts gelöst. Dieses ist in unserem GitHub repository hinterlegt und für jeden zugänglich (ref zum Link). 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.

```
import secrets
import string

def get_random_password():
temp = string.ascii_letters + string.digits
+ string.punctuation

alphabet = temp.replace('\'', '')
.replace('\\', '')
.replace('\"','').replace('\`', '')
.replace(';', '')

password = ''.join(secrets
.choice(alphabet) for i in range(10))
return password

if __name__ == "__main__":
print(get_random_password())
```

Als Passwortkonzept wurde sich auf einen 10 Zeichen langen Key geeinigt. Dieser benützt 90 Zeichen in Form von Groß- und Kleinbuchstaben, Zahlen und Sonderzeichen. Begründet wurde diese Entscheidung mit folgender Annahme:

Angenommen ein leistungsstarker Rechner schafft durch Brute-Force 2 Billionen Keys pro Sekunde, so würde er 346 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:

$$(90^{10})\ keys\ \div\ 2000000000000\ \frac{keys}{s} = 17433922,005\ s \tag{1}$$

$$29936846.961918945312\ s\ \div\ 60\ \div\ 60\ \div\ 24\ \approx 202\ Tage \tag{2}$$

$$202\ Tage\ \div\ 2\ =\ 101\ Tage \tag{3}$$

Die Passwortgenerierung wird mithilfe eines Python Skripts gelöst. Dieses ist in unserem GitHub repository hinterlegt und für jeden zugänglich (ref zum Link). 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 +
6          string.digits
7          + string.punctuation
8
9          alphabet = temp.replace('\'', '')
10         .replace('\\', '')
11         .replace('\"','').replace('\`', '')
12         .replace(';', '')
13
14         password = ''.join(secrets
15         .choice(alphabet) for i in range(10))
16         return password
17
18 if __name__ == "__main__":
19         print(get_random_password())
```

9 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 QRCodes 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 QRCode
        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
```

9.1 Automischer Tausch

Cron erlaubt es, das Passwort regelmäßig zu einer definierten Zeit, hier beispielsweise Montags um 03:00 Uhr nachts, zu tauschen. Damit das Passwort und der QRCode dennoch auf dem angeschlossenen Bildschirm angezeigt werden ist es wichtig beim Aufrufen des Skripts den Standardoutput 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
```

10 Ausgabe des Passworts

Derzeit wird aufgrund dem fehlenden Display (bisher noch nicht erhalten) das Passwort im Klartext auf die Konsole ausgegeben. Zusätzlich wird im Folgenden die Generierung des QR-Codes erläutert.

10.1 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};;
               ↪ '
7          .format(
8                  ssid=ssid,
9                  security=security,
10                 password=password
11         ))
12         print(qr.terminal())
13
14 if __name__ == "__main__":
15         ssid = sys.argv[1]
16         security = sys.argv[2]
17         password = sys.argv[3]
18         create_qr_code(ssid, security, password)
```

11 Fazit mit Ausblick

12 Quellenverzeichnis