Verificare il Green Pass europeo

Da una settimana, il Green Pass (che dimostra l'immunità o comunque il non contagio dal Covid-19) è obbligatorio per entrare in buona parte degli edifici in tutta Italia. Molti gestori di attività che si svolgono al chiuso sono preoccupati di non riuscire a controllare i Green Pass del pubblico. Ma è davvero così difficile? In realtà, no. L'EU Digital COVID Certificate è infatti un certificato digitale, le cui specifiche sono pubblicamente disponibili proprio per consentire a chiunque di verificare la validità di uno di questi QR code.

Esistono delle app per smartphone che svolgono questa operazione, ma richiederebbero comunque un operatore che scansioni i vari QR code degli avventori di un locale, e non sempre questo è praticabile. Esiste, però, la possibilità di automatizzare tutto il procedimento, gestendo l'accesso all'edificio tramite un meccanismo come un tornello, o una porta azionabile elettronicamente, e un semplice programma in Python che possiamo scrivere in breve tempo. Utilizzando un computer dotato di pin GPIO come il RaspberryPi è possibile realizzare un sistema completamente automatico: si può usare una webcam per riconoscere il QRcode, un lettore di smartcard per confrontare i dati con quelli della Tessera Sanitaria, e un relay per attivare il tornello (o la porta) soltanto nel caso in cui il Green Pass risulti valido. Il ricorso alla Tessera Sanitaria è fondamentale perché altrimenti un utente potrebbe presentarsi alla porta d'ingresso con un QRcode appartenente a un'altra persona, magari fotografato e condiviso tra tanti utenti. La Tessera Sanitaria è invece una sola per ogni cittadino, quindi permette di identificare automaticamente le persone e assicurarsi che ogni accesso sia legittimo. Naturalmente si potrebbero utilizzare altri

meccanismi, come la CIE o la Firma Digitale, ma la Tessera Sanitaria è l'unico ID digitale a disposizione di tutti i cittadini italiani. Il progetto che proponiamo si basa su un RaspberryPi2 o superiore, con RasperryOS Buster, una webcam USB, un lettore di smartcard USB, un altoparlante passivo con jack audio, e un eventuale modulo relay per far scattare l'apertura della porta.

Table of Contents

Installare i requisiti

I requisiti di questo software sono parecchi, ma possiamo installarli con una serie di comandi. Da notare che ci serve lo script https://github.com/panzi/verify-ehc, e quindi dovremo anche installare tutti i suoi requisiti. Possiamo farlo con questa serie di comandi:

In poche parole, prima di tutto installiamo le varie librerie necessarie per leggere le smartcard, poi quelle necessarie per prelevare immagini dalla webcam e manipolare le immagini (utilizzeremo OpenCV e PIL). Poi serviranno anche le librerie per gestire i pin GPIO del Raspberry, in modo da attivare un relay, e quelle per decodificare i QRCode. Infine, la libreria per emettere suoni (beepy) e i vari requisiti di Verify EHC.

Riconoscere il QRCode

Per prima cosa, scriveremo il codice che ci serve per riconoscere il QRcode, cioè per ottenere il testo (che poi tradurremo in JSON). Qui è necessario un piccolo hack: al momento, la versione di Debian disponibile come RaspberryOs è Buster. Purtroppo questa versione è ormai molto vecchia, quindi non è possibile avere le ultime versioni dei pacchetti. E la libreria qrtool è disponibile solo per Python2 (si trova su apt come **python-qrtools**). Esistono ovviamente divese altre librerie, ma questa è quella che abbiamo notato essere più efficiente e semplice da utilizzare. Quindi per ora metteremo le sue poche righe di codice in uno script a parte, che verrà interpretato da Python2 invece che da Python3: in futuro, usando la nuova versione di Debian, sarà possibile usare questa libreria nella versione per Python3. Il codice è questo:

Il codice è estremamente semplice: lo script si aspetta in argomento il percorso di un file contenente l'immagine di un QRcode da interpretare. Con questo file si può costruire un oggetto QR, decodificabile con la funzione **decode()**. A questo punto la variabile **.data** contiene il testo che è stato riconosciuto nel QRCode, che possiamo restituire sullo standard output. E che andremo a leggere dal programma principale.

Il Green Pass è un testo

firmato con le informazioni del paziente memorizzate in un JSON

Le funzioni di servizio

Ora iniziamo a scrivere lo script principale, quello che si occuperà di svolgere tutto il processo di verifica sia del Green Pass che della Tessera Sanitaria. Cominciamo dall'importazione delle librerie:

Librerie extra sono sostanzialmente quelle a cui accennavamo per lo script di installazione delle dipendenze, poi servono alcune librerie standard di Python per la gestione del JSON, dei sottoprocessi e dell'orario.

Continuiamo definendo alcuni oggetti che saranno utili per tutto lo script, e che quindi avranno valore globale. Per esempio, il percorso in cui trovare il file di configurazione, oppure quello in cui scrivere i log. Poi proviamo a importare le librerie per gestire i pin GPIO del RaspberryPi: saranno necessarie per attivare il relay, e quindi aprire automaticamente la porta o il tornello nel caso il Green Pass risulti valido. In realtà possiamo anche utilizzare lo script su un computer diverso dal Raspberry, e in quel caso non riusciremmo a importare le librerie dei GPIO. In questo caso impostiamo la variabile **rpi** a False, così sapremo che non ci troviamo su un Raspberry. Creiamo un dizionario vuoto per memorizzare la configurazione, e poi l'oggetto **reader**: questo rappresenterà il nostro punto di accesso al lettore di smartcard. Siccome dovrebbe essere possibile procedere anche senza il lettore, perché l'utente potrebbe decidere di usare solo la webcam per il riconoscimento del QR code e poi lasciare a un operatore l'identificazione della persona, se non riusciamo a trovare il lettore di Smart Card catturiamo

l’eccezione e andiamo avanti comunque.

Definiamo due funzioni “di servizio”, non fondamentali ma utili per definire due procedure e non preoccuparsene più. La prima si occupa di leggere il file di configurazione, che sarà nel formato JSON, e memorizzare il contenuto in un dizionario. La seconda è quella che apre la porta facendo scattare il relay: ci serve il numero del pin GPIO da attivare, ma vogliamo anche assicurarci di essere davvero su un Raspberry, perché altrimenti non ci sono i GPIO e non dobbiamo fare nulla.

Decodificare il Green Pass

Iniziamo con le cose “serie”: lo script **verify-ehc.py** si occupa di decodificare la stringa del Green Pass (che è un testo codificato in Base45).

Lo chiamiamo direttamente con os.system, scrivendo l’output in un file. Poi leggiamo il file, memorizzandolo come testo in una variabile. Utilizziamo os.system perché il modulo subprocess ha difficoltà a leggere tutte le righe, dal momento che lo script scrive l’output a intervalli non regolari.

L’output è diviso su più righe, e in realtà a noi interessano solo alcune. Nello specifico, ci interessa la riga **Is Expired** che, se presente, indica che il certificato era valido, ma ora è scaduto. E poi la riga **Signature Valid**, che è presente solo se la firma del certificato risulta corretta: questa indica che il certificato è stato generato da uno dei ministeri della salute dell’Unione Europea, e quindi possiamo considerarlo non contraffatto. Infine, cerchiamo la riga **Payload**, perché dopo di essa viene riportato l’intero contenuto del Green Pass vero e proprio, con i dati personali della persona. Questo payload è in formato JSON, quindi possiamo tranquillamente caricarlo

in un dizionario, assicurandoci di prendere il testo e rimuovere gli invii a capo per evitare che il modulo json di Python possa avere difficoltà a interpretarlo.

Leggere la Tessera Sanitaria

Purtroppo non esiste una documentazione pratica per l'utilizzo delle informazioni presenti nella Tessera Sanitaria italiana, solo delle specifiche tecniche. Noi ci siamo basati sul lavoro di decodifica fatto alcuni anni fa da MMXForge.

I dati su una tessera sanitaria sono memorizzati in un particolare filesystem, ed è possibile selezionare i file inviando una serie di comandi binari (che codifichiamo in esadecimale per leggiblità). La funzione per la lettura dei dati personali dalla tessera sanitaria deve quindi iniziare stabilendo una connessione con la smartcard e poi utilizzando quella connessione per inviare una serie di comandi.

La Tessera Sanitaria italiana contiene un microchip leggibile con un comune lettore di smartcard

Otteniamo come risposta una tupla di tre oggetti: il primo rappresenta i dati restituiti dalla smartcard, gli altri due eventuali codici per identificare errori. Se tutto va bene, sw1 e sw2 dovrebbero sempre contenere i valori 0x90 e 0x00 rispettivamente. Nel nostro caso non c'è bisogno di

verificarli, perché siamo solo interessati a estrarre i dati dal file corretto, qualsiasi cosa vada storta indica che la tessera inserita non era corretta e possiamo considerare nulla l'identificazione.

A questo punto, la variabile data contiene tutti i dati dell'utente, ma come byte. Dobbiamo convertirla in stringa e poi estrarre i singoli dati. I dati sono codificati in modo abbastanza semplice: i primi due caratteri contengono il numero di byte che costituiscono il successivo dato, così sappiamo sempre esattamente quando leggere. Quindi dobbiamo leggere i primi due caratteri, trasformarli in un numero intero, e leggere quel numero di byte per estrapolare il codice dell'emettitore della tessera. Poi leggiamo i due caratteri successivi per conoscere il numero di byte da leggere per avere il cognome. Segue il nome dell'intestatario della tessera.

Con la stessa logica possiamo continuare a leggere i dati personali dell'utente. Sono, in sequenza, sesso, statura, codice fiscale, cittadinanza, comune di nascita e stato di nascita (nel caso la persona non sia nata in Italia). Memorizziamo tutti questi dati in un dizionario, così sarà più facile accedere a quello che ci interessa.

Verificare se il certificato è valido

Iniziamo ora la funzione che ci permetterà di stabilire se il Green Pass sia valido.

I due oggetti che dobbiamo ricevere in argomento sono il dizionario con i dati del green pass e quello con i dati della tessera sanitaria. Possiamo considerare il green pass immediatamente non valido se dai suoi stessi dati risulta che

sia scaduto (expired) o se la sua firma non risulti correttamente apposta da uno dei ministeri della salute europei (in questo caso **signature_valid** sarebbe **False**).

Se è stata fornita una Tessera Sanitaria valida, possiamo confrontare i suoi dati con quelli del Green Pass. Dobbiamo solo fare una piccola conversione sulla data di nascita, perché nella TS è memorizzata nel formato GG/MM/YYYY, mentre nel GP è memorizzata come AAAA-MM-GG. Poi possiamo confrontare data di nascita, nome, e cognome: li confrontiamo in minuscolo, per evitare problemi con eventuali lettere mauscole non corrispondenti.

Se non è stata fornita una tessera sanitaria, per esempio perché la persona non è un cittadino italiano, e la configurazione consente comunque all'operatore di verificare l'identità della persona, facciamo apparire un semplice prompt per chiedere proprio all'operatore se il Green Pass appartenga davvero alla persona che si è presentata. Se l'operatore preme i tasti **y** oppure **s**, il Green Pass è considerato legittimo, ma segnaliamo comunque che non era stata fornita una tessera sanitaria. Così nell'eventuale log viene indicato che l'identificazione è stata manuale.

Per leggere un QR code basta una comune webcam, anche non FullHD

Catturare il QRcode dalla webcam

Per catturare il QRcode creiamo una funzione che utilizza OpenCV, così è facile scattare foto in tempo reale dalla webcam.

L'immagine verrà inserita nella variabile **img**.

Ora utilizziamo OpenCV per scrivere l'immagine su un file temporaneo (sempre lo stesso, tanto possiamo gestire un solo ingresso alla volta). Poi cerchiamo di tradurre questa immagine nel testo del GreenPass usando lo script qrcodereader. Non utilizziamo direttamente a funzione di lettura del QR code di OpenCV perché non funziona bene con webcam a bassa definizione. Se abbiamo ottenuto qualcosa, lo scriviamo nella variabile **data**.

Se è disponibile una sessione di Xorg, il server grafico di GNU-Linux, mostriamo una finestra con l'anteprima della foto scattata dalla webcam, così l'utente può capire se ha allineato correttamente il QR code. Chiaramente non possiamo farlo quando non c'è uno schermo. La funzione fa un ciclo continuo finché non viene riconosciuto un QRcode valido.

Mettere tutto assieme

Nella routine principale del programma possiamo riunire le varie funzioni che abbiamo scritto seguendo il filo logico della verifica del Green Pass: lettura del QRcode, lettura della tessera, confronto dei dati, responso all'utente sotto forma di segnale audio, apertura della porta, e eventuale messaggio sullo schermo.

Nella routine principale del programma prima di tutto leggiamo la configurazione dall'apposito file. Poi attiviamo un ciclo continuo, che svolgerà le varie operazioni in sequenza: prima di tutto si riproduce un suono, per segnalare che siamo pronti a leggere un nuovo QRcode. Poi procediamo a avviare la funzione per la lettura delle immagini dalla webcam: quando questa avrà identificato e decodificato un QR code, potremo procedere a verificarne il contenuto. Fatto questo, andiamo a leggere l'eventuale Tessera Sanitaria presente nel lettore (se la tessera non è stata inserita, il dizionario risultante sarà vuoto).

Ora abbiamo tutto quello che potrebbe servirci, quindi possiamo procedere alla verifica delle credenziali. Come abbiamo visto, la funzione **isCertValid** ci restituisce una tupla di due oggetti. Il primo è un semplice booleano, chiamato **val**, che sarà True se il Green Pass è valido e corrispondente alla Tessera Sanitaria e False negli altri casi. Mentre **err** è una stringa che contiene l'eventuale codice di errore ottenuto. Se il Green Pass è valido non soltanto lo segnaliamo con un suono, così è subito palese se l'accesso sia consentito oppure no, ma inneschiamo anche l'apertura della porta o del tornello con la funzione **open_door**.

Per finire, gestiamo il caso in cui la configurazione preveda di loggare i dati, per esempio per identificare. Ovviamente questa è una eventualità che richiede una certa cautela, perché si tratta di memorizzare dati privati sensibili delle persone, quindi non è detto che qualcuno voglia attivarla. Se il log è attivo, quindi, generiamo una riga di log costituita dal timestamp, lo stato della validità del green pass (**OK** oppure **ERROR**), l'eventuale codice fiscale (che però è una stringa vuota se non è stata fornita una Tessera Sanitaria), e l'eventuale messaggio di errore.
Prima di ripetere il ciclo attendiamo un secondo, per dare all'utente il tempo di togliere la propria tessera sanitaria e il QRcode dai lettori. Poi siamo pronti per un altro ciclo,

verificando le credenziali di un’altro avventore..

Il codice sorgente

Potete trovare il codice sorgente di questo strumento su GitHub:
https://github.com/zorbaproject/greenpass-turnstile
Il repository integra già la dipendenza https://github.com/panzi/verify-ehc, lo script che esegue la decodifica del Green Pass.