PROGETTO IoT SECURITY

Sviluppato da: Dott. Alessandro Orsatti

Il progetto realizzato è un sistema IoT che genera delle password sicure tramite i rumori dei segnali radio. Il sistema cambierà ogni secondo la frequenza per connettersi ad una stazione radio diversa, in questo modo, oltre alla casualità del segnale, è una buona tecnica di difesa contro gli attacchi.
La password verrà poi criptata e salvata in un database in Google firebase, in aggiunta, il dispositivo IoT registra sulla blockchain un evento di log immutabile firmato digitalmente, per garantire un *audit trail*, **senza mai esporre in chiaro la password stessa**.

Requisiti hardware

- Scheda ESP32 con modulo wi-fi e bluetooth
- Modulo radio FM I2C
- Pulsante (se premuto stamperà la password generata sul monitor seriale)
- Microfono

Requisiti software

- Arduino IDE
- Google Firebase
- Python (per script di test e verifica duplicati)

Sviluppo hardware

Un modulo radio FM I2C viene utilizzato per sintonizzarsi sulle frequenze radio tramite antenna realizzata con un semplice cavo jumper non schermato, in questo modo converto il segnale digitalmente in una stringa con caratteri dell'ASCII, garantendo la casualità dei caratteri e di conseguenza la non replicabilità. Il segnale prima di essere elaborato dal processore dell'ESP32 viene prima amplificato da un microfono per aumentare il rumore e renderlo ancora più variabile. Al sistema è stato integrato un bottone che ad ogni iterazione genera una password in chiaro visibile sul Serial monitor, successivamente l'algoritmo la cifra e la invia alla blockchain.

Sviluppo software

La parte software è stata sviluppata in C++ attraverso Arduino IDE che permette una comunicazione semplice con l'ESP32 e una visualizzazione dei dati elaborati da esso. Il software sviluppato e caricato sull'ESP32 permette di convertire il segnale radio in un segnale digitare il cui input genera una stringa di caratteri dell'ASCII (da 32 a 126 della tabella) di lunghezza 16. Per irrobustire l'algoritmo e prevenire rari casi di assenza di segnale (silenzio radio), è stato implementato un algoritmo ibrido: al rumore radio viene applicata una leggera perturbazione software aggiuntiva (salting). Per questioni di sicurezza e per garantire la casualità della password è quindi necessaria questa implementazione e attraverso uno sniffer costruito in python sono state generate e salvate oltre 10 mila password e successivamente è stata eseguita una ricerca dei duplicati per verificare l'affidabilità del progetto ed è risultato che non erano presenti doppioni. Grazie a questo test e questo metodo di generazione delle password si è potuto stimare che le probabilità che possano esserci delle password duplicate sono molto basse. Dopo che la password è stata generata, viene cifrata utilizzando l'algoritmo **AES-128** (Advanced Encryption Standard) con una chiave segreta a 128 bit. La stringa cifrata viene poi inviata al database Firebase attraverso la rete, poiché il modulo ESP verrà connesso alla rete WI-FI. Qui, il sistema simula una struttura **Blockchain**: ogni nuovo salvataggio include l'Hash (SHA-256) del record precedente, creando una catena immutabile che garantisce l'integrità dello storico (Audit Trail).

Modellazione dei requisiti

Caso di Applicazione:

- Generazione di password ad alta entropia per sistemi critici tramite sorgente TRNG radio.

Errori (Non intenzionali):

- *Guasto Hardware:* Il modulo radio si disconnette o l'antenna si stacca -> Il sistema deve rilevarlo e fermarsi o segnalare errore (non generare zeri).
- *Assenza di Rete:* Impossibile raggiungere Firebase -> Il sistema deve memorizzare localmente o riprovare.

Vulnerabilità (Debolezze intrinseche):

- Chiavi crittografiche salvate nella memoria Flash in chiaro.
- Dipendenza dalla disponibilità di spettro radio "sporco" (rumore).

Attacchi (Sfruttamento delle vulnerabilità):

- *Jamming:* Saturazione della banda FM per ridurre l'entropia.
- *Dump della Memoria:* Estrazione fisica del firmware.

Soluzioni:

- Algoritmo ibrido (Salting) per mitigare il jamming.
- Blockchain chaining per rilevare manomissioni storiche.

Use case

Questo sistema permette di generare una password totalmente casuale che può essere usata in vari modi:

NOME USE CASE	ATTORE PRINCIPALE	TRIGGER	PRE-CONDIZIONI	FLUSSO DI EVENTI	POST CONDIZIONE
Generazione manuale di una password sicura	Utente/Operatore di sicurezza	L'utente preme il pulsante fisico sull'ESP32	L'ESP32 è acceso, connesso al Wi-Fi e il modulo radio è inizializzato	1. Il sistema rileva la pressione del pulsante. 2. Il modulo radio scansiona le frequenze e cattura il rumore di fondo. 3. L'algoritmo converte il segnale analogico in una stringa alfanumerica e applica la perturbazione software. 4. Il sistema stampa la password **in chiaro** sul Monitor Seriale per l'utente.	La password è visualizzata a schermo e il suo backup cifrato è immutabile su Firebase.

				5. Il sistema cifra la password con **AES-128.** 6. Il sistema scarica l'ultimo Hash da Firebase, calcola il nuovo blocco e lo invia al cloud.	
Audit Trail e Verifica Integrità	Revisore/ Amministratore database	Controllo periodico di sicurezza o sospetto incidente.	Accesso alla console di Google Firebase.	1. L'Auditor apre il Realtime Database su Firebase. 2. Seleziona la cartella "chain". 3. Esamina due blocchi consecutivi. 4. Esamina due blocchi consecutivi	**Successo:** Gli hash coincidono -> La catena è integra, i dati sono validi. **Fallimento:** Gli hash non coincidono -> Qualcuno ha modificato/cancellato un dato nel database (Tampering rilevato).
"Self-Healing" e Sincronizzazione all'Avvio	ESP32	Accensione del dispositivo	-	1. Il sistema avvia la connesione WI FI. 2. Il sistema si autentica su Firebase tramite API Key. 3. Il sistema esegue una richiesta **READ** per ottenere l'impronta digitale	Il dispositivo è pronto a generare un *nuovo* blocco che si collegherà matematicamente a quello precedente, senza spezzare la catena.

				dell'ultimo blocco salvato. 4. Il sistema memorizza l'hash nella RAM	

Misuse case

Primo caso

Attore

Jammer

Scenario

Un attaccante potrebbe eseguire un "Signal injection" ovvero andrebbe a disturbare il segnale radio creando così un segnale muto, di conseguenza l'entropia crolla, ad un certo punto verrebbero generate sempre le stesse password.

Obiettivo

L'obiettivo di questo attacco è di far generare al sistema la stessa password per rubarla e accedere al sistema. (da vedere meglio)

Contromisure

Un modo per difendersi sarebbe quello di cambiare attraverso un algoritmo pseudorandomico implementato nel codice, dei caratteri scelti casualmente con altri caratteri alfanumerici scelti sempre casualmente. In questo modo anche la password generata dal segnale pulito risulta difficile da indovinare.

Secondo caso

Attore

Crittanalista

Scenario

L'attaccante sa che è presente una modifica software dove vengono scelti 3 caratteri casuali utilizzando una funzione randomica per compensare il segnale piatto. Sfruttando il jamming, se l'attaccante riesce a indovinare il seed del generatore random dell'ESP32, può prevedere le password future, bypassando tutta la complessità della radio.

Obiettivo

L'obiettivo dell'attaccante è disturbare intenzionalmente il segnale radio (Jamming) per costringere il tuo software a usare pesantemente la funzione random () invece del rumore radio.

Contromisure

Per difendersi la soluzione sarebbe quella di aumentare il numero di caratteri modificati dalla funzione in modo da rendere l'attacco quasi inefficiente.

Terzo caso

Attore

Attaccante fisico

Scenario

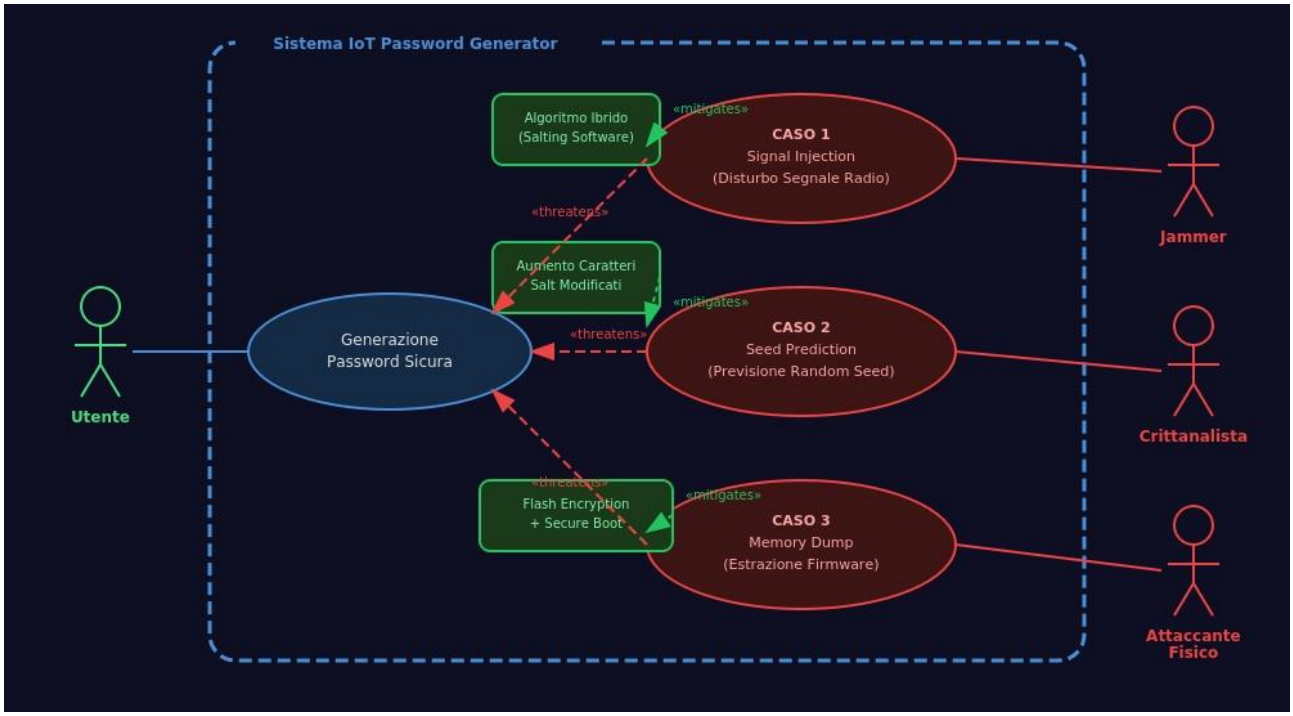
L'attaccante ruba l'ESP32 e scarica il contenuto della memoria Flash. Con la chiave AES recuperata, l'attaccante può decifrare tutto lo storico delle password salvate su Firebase.

Obiettivo

Recupera il codice così ottiene in chiaro la chiave API di firebase, la chiave di crittografia e le credenziali WI-FI.

Contromisure

Per difendersi, la contromisura tecnica sarebbe l'abilitazione della Flash Encryption e del Secure Boot, rendendo inutile il dump della memoria anche se il dispositivo viene rubato.



Abuse case

In questa sezione verranno mostrati 2 casi in cui gli attori non sono degli attaccanti, ma degli utenti con autorizzazione legittima sull'uso del dispositivo, i quali lo usano in maniera errata.

Primo caso

Attore

Utente interno

Scenario

L'utente ha legittimo accesso fisico al dispositivo. Invece di generare una password quando serve, preme il **pulsante** ripetutamente e ad altissima frequenza.

Obiettivo

L'obiettivo è quello di saturare il database Firebase e/o consumare tutta la banda del Wi-Fi rendendo il servizio inutilizzabile.

Contromisure

Implementare un "Rate Limiting", cioè la password può essere generata alla pressione del pulsante solo ogni 10 secondi.

Secondo caso

Attore

Operatore di sicurezza

Scenario

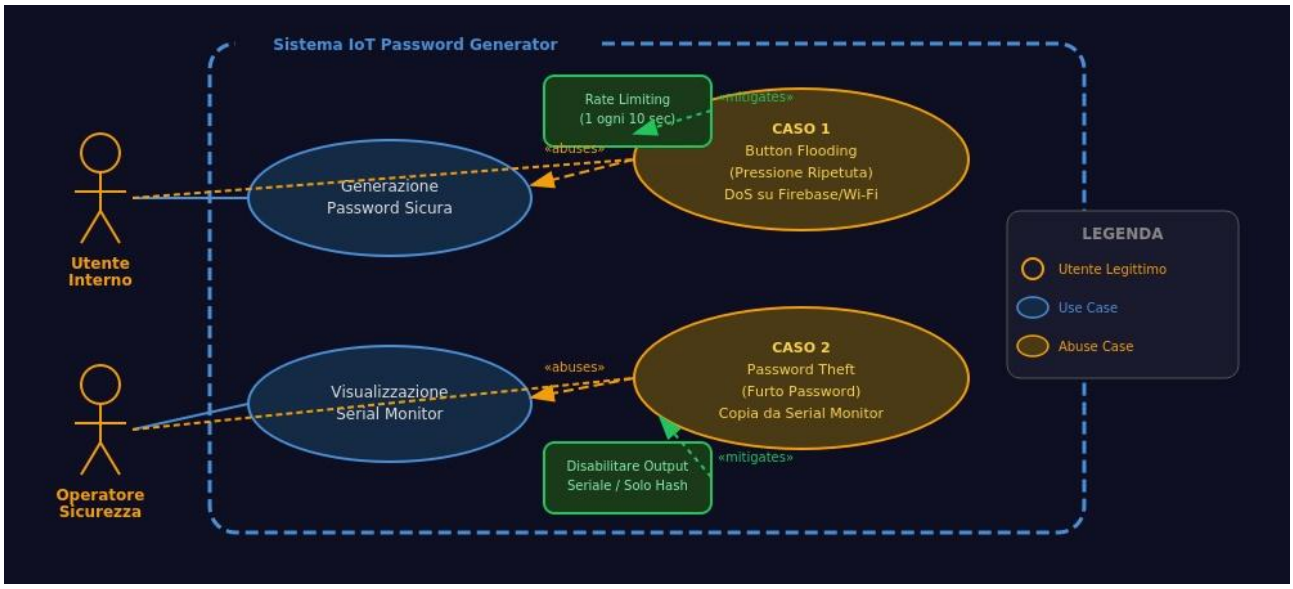
L'operatore deve generare una password per un dipendente dell'azienda. Usa il dispositivo, vede la password sul Monitor Seriale, se la appunta segretamente, e poi consegna la password al dipendente.

Obiettivo

L'obiettivo dell'operatore è rubare la password in chiaro del collega prima che venga criptata al fine di usarla per i suoi scopi.

Contromisure

Disabilitare l'output seriale della password in chiaro nella versione di produzione, oppure mostrare solo l'hash.



Fonte

https://www.mdpi.com/1424-8220/19/19/4130