Sécuriser l'ESP32

Table des matières

Sécurité physique

Quelles attaques et quels risques ?

Tout d'abord, il faut savoir qu'aujourd'hui en IoT, tous les appareils sont vulnérables si un accès physique un possible. Il ne s'agit ensuite qu'une question de temps et de coût concernant la solution d'attaque.

Bien que l'ESP32 soit assez sécurisé physiquement, les attaques les plus courantes et les plus embêtantes concerne la vulnérabilité d'injection de fautes (Fault Injection Vulnerability). Le but de l'attaquant est de pouvoir lire et modifier des informations telles que le micro-logiciel et les données stockées dans la mémoire flash de l'appareil. De ce fait, il peut tout à fait démarrer un micro-logiciel non-approuvé et dangereux.

Si l'ESP32 n'est pas sécurisé correctement, n'importe qui peut démarrer un micro-logiciel sur la carte à l'aide d'un simple câble USB et d'un ordinateur. Cependant, si des mesures ont été prises, il faut faire preuve de beaucoup plus d'investissement en termes de coût et de temps. En effet, il existe diverses méthodes physiques pour une injection de défaut, telles que des fluctuations de tension ou d'horloge soigneusement chronométrées, une variation de température externe, une irradiation laser ou l'utilisation de champs magnétiques puissants. Pour réaliser l'investissement nécessaire si le système est sécurisé correctement, il est important de noter les points suivants concernant l'attaque :

- Elle nécessite un accès physique afin que des pièces puissent être retirées de la carte de circuit imprimé ou du module. Les traces de circuits imprimés peuvent également devoir être coupées ou modifiées avant d'être connectées à l'équipement de l'attaquant.
- L'équipement de tension parasite de l'attaquant doit être connecté à des broches d'alimentation particulières de l'ESP32 afin que des tentatives répétées puissent être faites pour introduire une fluctuation de tension correctement synchronisée.

Comment bien sécuriser l'ESP32 ?

Afin de rendre le plus difficile possible une attaque concernant la vulnérabilité d'injection de fautes, il faut mettre en place deux mesures de sécurité : secure boot et flash encryption.

Lorsque le secure boot est activé, le chargeur de démarrage logiciel (software bootloader) génère et programme une clé de démarrage sécurisé « unique par périphérique » dans la mémoire eFUSE (mémoire interne à l'ESP32) lors du premier démarrage. Après cela, les potentiels futurs attaquants ne pourront seulement récupérer que la clé public avec laquelle il est impossible de récupérer la clé privée. Or, chaque démarrage de micro-logiciel nécessite une vérification de la clé privé. Ainsi, sans cette dernière, il est impossible de modifier le code pour démarrer un micro-logiciel malveillant.

Concernant le flash encryption, le même procédé est utilisé et il faut une clé privé pour pouvoir accéder aux données de la mémoire flash.
À noter qu'il est possible d'utiliser le secure boot sans le flash encryption mais pour avoir un environnement sécurisé, il vaut mieux utiliser les deux.

Sécurité à distance

L'ESP32 permet nativement d'utiliser des fonctionnalités Wifi ou Bluetooth. Ces dernières présentent différentes vulnérabilités pouvant potentiellement permettre à une personne mal intentionnée d'attaquer l'appareil.
La conséquence la plus commune de ces attaques est le crash de l'ESP32, mais certaines vulnérabilités permettent à l'attaquant de rejouer, décrypter ou déguiser certaines frames via un access point rogue.
On peut retrouver le détail de plusieurs de ces vulnérabilités sur le site de la National Vulnerability Database, que ce soit par rapport aux crashs (**CVE-2019-12586)** ou à l'interception des frames (**CVE-2019-12587).** Cette dernière vulnérabilité est d'ailleurs classée en score de dangerosité HIGH.
Néanmoins, ces deux vulnérabilités, à l'image d'autres pouvant être trouvées sur Internet, profitent de faiblesses dans le protocole EAP d'authentification. Or, ce protocole est utilisé dans le cadre d'une connexion sécurisé Wifi et implique donc que le Wifi soit activé sur notre machine. Après consultation de différents sites, il apparaît que le Wifi est par défaut désactivé et qu'il est nécessaire d'intégrer des librairies particulières (notamment <Wifi.h>) afin de s'en servir. Notre ESP32 n'utilisant pas ces librairies, nous devrions être à l'abri d'attaques visant des vulnérabilités du réseau Wifi.
De même, nous devrions être à l'abri des vulnérabilités liées à l'utilisation du Bluetooth (voir source).

Que reste-il faire ?

Comme vous avez pu le voir, notre système n'est pas vulnérable aux attaques à distance. Il ne reste donc que les attaques physiques que nous n'avons malheureusement pas pu mettre en place cependant nous avons une piste. En effet, pour il se pourrait que le secure boot et la flash encryption soit assez simple à mettre en place.

- Il suffirait d'installer cette librairie sur vscode : https://marketplace.visualstudio.com/items?itemName=espressif.esp-idf-extension
- Puis de regarder le tutoriel ici : https://docs.espressif.com/projects/esp-idf/en/latest/esp32/security/secure-boot-v1.html

Le problème auquel nous avons dû faire face est la mise à jour de l'extension. Celle-ci ne s'installait pas comme expliqué sur le premier lien et de ce fait son utilisation était compromise. Les alternatives trouvées étaient bien trop compliquées ou trop risquées par rapport au temps et au nombre de composant disponible que nous avions car elles impliquaient de coder au cœur de l'ESP32 impliquant un potentiel dysfonctionnement si une erreur était commise.

Possibilité d'amélioration, alternatives

L'évolution du projet peut nous amener à considérer l'utilisation des fonctionnalités WIFI ou Bluetooth de l'ESP32 afin de réduire la taille du dispositif par exemple.
Le problème de la sécurisation du produit se pose. Avec un échange d'informations par Bluetooth ou WIFI, il est possible de nuire au fonctionnement de l'appareil, les menaces les plus fréquentes étant le déni de service, les écoutes et analyses passives, l'interception de données ou encore les attaques actives qui modifieraient le flux d'information. Il faudra donc faire un choix dans les différents mécanismes de protections possible en fonction de la finesse voulue et de ce qui est faisable avec notre ESP32 (voir les mécanisme de protections dans les sources).
En plus de ces sécurités, il faudrait également éviter les « mauvaises pratiques » et ainsi désactiver le Bluetooth lorsqu'il n'est plus nécessaire, ne pas répondre à une demande de connexion inattendue ou

éviter les protections par défaut avec des codes PINS trop simples. Si jamais vous en ressentez le besoin, un tuto Bluetooth avec ESP32_BLE_Arduino est disponible dans les sources.
Espressif a annoncé en Décembre 2020 un nouvel Esp : l'ESP32-C3, un microcontrôleur WIFI et BLE 5.0 dans lequel sont intégrés plusieurs mécanismes de sécurité. Ce microcontrôleur pourrait être une bonne alternative pour passer sur un signal sans fil, qu'il soit par WIFI ou Bluetooth. Il est équipé d'un démarrage rapide et sécurisé sur les applications de confiance (RSA-3072), d'un cryptage de mémoire flash (AES-128-XTS), d'un périphérique de signature numérique et d'un word controller (deux environnements d'exécution différents). On peut aussi penser à passer sur un système d'électrodes sans fil, qui seraient parfaitement prises en charge par ce microcontrôleur. Veuillez voir sur le site pour une demande de commande (site en sources).

Bibliographie

https://www.espressif.com/en/news/ESP32_FIA_Analysis
https://docs.espressif.com/projects/esp-idf/en/latest/esp32/security/secure-boot-v1.html
https://www.esp32.com/viewtopic.php?t=5100 : (ESP_igrr est un administrateur du site)

> Re: WiFi & BT not required, how?
> by ESP_igrr » Thu Mar 22, 2018 9:48 am
>
> With Arduino-esp32, If you don't include WiFi.h, then WiFi will stay disabled. Same for BT.

https://www.securiteinfo.com/attaques/phreaking/securite_reseaux_sans_fil_Bluetooth.shtml

http://tvaira.free.fr/bts-sn/activites/activite-ble/activite-ble-esp32.html

https://www.espressif.com/en/products/socs/esp32-c3