Attaques inter-protocolaires par détournement du contrôleur Bluetooth d'un téléphone mobile

Romain Cayre*†, Florent Galtier*
*CNRS, LAAS, 7 avenue du colonel Roche, F-31400
†APSYS.Lab, APSYS
Email: *firstname.lastname@laas.fr †firstname.lastname@airbus.com

I. Introduction

Le déploiement rapide et massif des objets connectés, notamment dans un cadre industriel, pose aujourd'hui de nombreuses problématiques de sécurité. En particulier, leur mobilité, l'hétérogénéité des protocoles utilisés et les communications sans-fil qu'ils utilisent et qui sont souvent très peu sécurisées, ouvrent de nouvelles perspectives aux attaquants.

Ce dernier point a mené à l'apparition d'un nouveau type d'attaques, basées sur les similarités entre les couches physiques des différents protocoles utilisés, qui permettent de pivoter de l'un à l'autre pour rebondir sur des équipements vulnérables ou échapper à une surveillance des couches supérieures des piles protocolaires. Ce type de pivot n'est pas nécessairement malveillant, et est étudié dans l'état de l'art sous le nom de *Cross-Technology Communications*, ou CTC. Lors de nos travaux, nous avons pu nous baser sur une approche de ce type pour réaliser une attaque permettant d'émettre et recevoir des trames 802.15.4, notamment *Zigbee*, depuis une puce supportant uniquement le *Bluetooth Low Energy* (*BLE*)[1]. Toutefois cette attaque nécessitait d'avoir un contrôle suffisant sur la couche physique pour fonctionner de façon bidirectionnelle.

Dans cet article, nous étendons ce type d'attaque à des puces largement répandues, notamment dans les téléphones portables, en ajoutant directement de nouvelles primitives au firmware de la puce pour s'affranchir de cette limitation. L'attaquant peut ainsi mener l'attaque aisément et avec peu de matériel, ayant seulement besoin d'un téléphone comportant une puce dont les capacités auront été ainsi augmentées. Nous présentons d'abord la démarche de rétro-ingénierie qui a été nécessaire pour identifier et instrumenter les différentes fonctionnalités des firmwares originaux, sur une puce intégrée à des téléphones grand public (BCM4375). Cela nous a permis d'y implémenter des primitives d'émission et de réception d'autres protocoles que le *Bluetooth*. Ensuite, nous décrivons les expérimentations qui ont été menées, avec succès, pour communiquer depuis ces puces vers des protocoles tels que *Zigbee*, *Mosart* ou *Enhanced ShockBurst*.

II. Rétro-ingénierie des firmwares

A. *Le framework* Internalblue

Dans la suite de cet article, nous avons utilisé le framework *Internalblue*[2], servant à faciliter le patching et la rétro-ingénierie de firmwares des contrôleurs *Bluetooth Broadcom* et *Cypress*, très répandus parmi les cartes sans-fil grand public. Cet outil nous a permis de récupérer le firmware de ces cartes, de faciliter son analyse dynamique, et d'injecter nos patchs afin d'altérer son fonctionnement et de le doter de nouvelles capacités offensives.

B. *Methodologie*

Nous avons focalisé notre approche sur un contrôleur *Bluetooth* récent, le BCM4375, intégré sur des téléphones mobiles grands publics. Nous avons choisi cette puce en particulier car elle supporte la version 5 de la norme *Bluetooth*, et notamment le mode LE 2M de la couche physique, permettant d'atteindre un débit de 2Mbit/s indispensable pour communiquer en *Zigbee* et ShockBurst. Pour faciliter notre analyse, nous avons également étudié le firmware d'un kit de développement IoT, le CYW20735, pour lequel les symboles sont connus.

Nous avons procédé à la rétro-ingénierie partielle des firmwares, à la fois par une analyse statique de ceux-ci et par une analyse dynamique de certaines fonctions grâce à *Internalblue*. En comparant les binaires, il nous a également été possible de trouver des correspondances entre des fonctions connues du CYW20735 et le firmware du BCM4375, les deux firmwares partageant beaucoup de code commun.

C. *Détournement des tâches de* Scan *et d'*Advertising

Nous avons focalisé notre analyse sur les mécanismes liés aux *advertisements* du BLE, notamment les tâches de *scan* (pour la réception de paquets) et d'*advertising* (pour l'émission de paquets). En effet, ces mécanismes ne nécessitant pas l'établissement préalable d'une connexion, ils constituent de bons candidats au développement de primitives d'émission et de réception pour d'autres protocoles radio.

Nous avons notamment identifié les fonctions destinées à la configuration du matériel radio, le callback de réception pour la tâche de *scan* et celui d'émission pour la tâche d'*advertising*. Nous avons ensuite détourné le flot d'exécution de ces fonctions vers notre propre code, nous permettant ainsi de modifier la configuration des registres radio. Cela nous a également fourni un accès à la sortie du démodulateur, accessible depuis un registre mappé en mémoire, et à l'entrée du modulateur, contrôlable indirectement en insérant dans la payload d'un paquet d'*advertising* le paquet à transmettre, exploitant une approche similaire à l'attaque "Packet-in-Packet"[3].

D. Configuration du composant radio

Afin de pouvoir implémenter le support des différents protocoles, il est nécessaire d'être en mesure de contrôler les éléments suivants : le préambule, la fréquence, la sélection d'un débit de 2Mbit/s et les données en entrée du modulateur et en sortie du démodulateur. Nous avons notamment identifié la fonction de configuration initialement destinée au support de la couche physique LE 2M. L'*access address* utilisée en BLE étant notamment utilisée comme motif de détection d'un paquet, il est possible d'utiliser cette fonction pour configurer un préambule arbitraire en mode 2Mbit/s. La configuration du *whitening* est réalisée par l'intermédiaire d'une fonction spécifique, qui peut être utilisée pour désactiver cette fonctionnalité, facilitant ainsi la manipulation des données liées au composant radio. Deux registres matériels distincts sont utilisés pour configurer la fréquence utilisée, selon la tâche (*scan* ou *advertising*) active: ils permettent cependant tous deux de choisir une fréquence arbitraire dans la bande 2.4-2.5GHz, en indiquant un offset en MHz par rapport à la fréquence 2402 MHz. Ainsi, en détournant ces différents mécanismes, nous avons pu configurer le composant radio afin d'implémenter des primitives de réception et d'émission permettant d'émettre et recevoir des paquets arbitraires utilisant la modulation GFSK à 2Mbit/s.

E. Communication Hôte - Contrôleur

Les nouvelles fonctionnalités introduites dans le firmware doivent pouvoir être manipulées depuis le téléphone (ou Hôte). La spécification *Bluetooth* introduit une interface nommée *Host Controller Interface* (ou HCI), permettant à l'Hôte d'envoyer des ordres à un contrôleur *Bluetooth* grâce des messages nommés commandes et au contrôleur de transmettre des informations à l'Hôte par le biais d'évènements. Pour gérer la communication vers l'Hôte, et notamment remonter les paquets reçus, nous avons utilisé l'interface existante en identifiant les deux fonctions permettant respectivement de construire un évènement arbitraire et de le transmettre à l'Hôte. Les commandes, quant à elles, sont gérées par l'intermédiaire d'un tableau de pointeurs de fonction: l'identifiant de commande est utilisé comme indice afin de sélectionner la fonction associée à la commande. Il a ainsi été possible d'ajouter de nouvelles commandes associés à des identifiants non utilisés, notamment destinées à déclencher les différentes fonctionnalités introduites dans le firmware.

III. Application aux protocoles

A. Zigbee

Pour implémenter le support du *Zigbee*, nous avons utilisé l'attaque *WazaBee*: celle ci permet de détourner la modulation GFSK du BLE pour la rendre compatible avec l'O-QPSK utilisée par le protocole *Zigbee*, en tirant profit de similitudes dans le fonctionnement de ces deux modulations. Nous avons notamment intégré au sein du firmware une table de correspondances, permettant d'associer à chaque symbole *Zigbee* sa représentation GFSK, et développé des fonctions permettant de réaliser automatiquement cette conversion lors de la réception ou de l'envoi d'un paquet. Les paquets *Zigbee* étant préfixés d'octets nuls, nous avons sélectionné un préambule correspondant à la représentation GFSK du symbole 0. La sélection d'un canal donné peut être réalisée facilement en récupérant la fréquence centrale associée au canal choisi et en calculant l'offset correspondant (en MHz).

Nous avons testé les deux primitives construites en interagissant avec un réseau de capteurs XBee, d'abord en écoutant passivement les messages échangés puis en injectant une série de paquets malveillants afin d'altérer les données reçues.

B. Mosart

Le protocole *Mosart* est un protocole couramment utilisé par les claviers et souris sans fil: il est basé sur une modulation de type GFSK à 1Mbit/s. Un paquet *Mosart* est composé d'un préambule (0x5555), d'une adresse sur 4 octets, d'un payload et d'un CRC. Nous avons constaté que, bien que le BLE supporte le mode 1Mbit/s, il n'était pas possible de choisir un préambule arbitraire en mode 1Mbit/s, l'*access address* statique liée aux *advertisements* semblant directement intégrée dans le composant radio, ce qui complique l'implémentation de la réception. Pour contourner cet obstacle, nous avons utilisé le mode 2Mbit/s, en dédoublant chaque bit. Lors de la réception d'un message, nous avons développé une fonction permettant de sélectionner un bit sur deux lors de la lecture de la sortie du démodulateur et de reconstruire le message associé: divers traitements (notamment lié à l'*endianness* et au *scrambling*) sont ensuite appliqués pour décoder le paquet.

Nous avons validé le bon fonctionnement de ces primitives en construisant un *keylogger*, nous permettant de collecter tout texte tapé sur un clavier utilisant le protocole *Mosart*, puis en injectant une série de frappes claviers malveillantes.

C. Enhanced ShockBurst

Le protocole *Enhanced ShockBurst* est un protocole propriétaire basé sur une modulation GFSK à 2Mbit/s: il sert de base à de très nombreux protocoles utilisés par des drones ou des claviers et souris sans fil, tels que Logitech Unifying. Chaque paquet est préfixé d'un préambule 0xAA ou 0x55, suivi d'une adresse de 5 octets, d'un payload et d'un CRC. La modulation utilisée étant identique à celle du BLE, il est relativement simple d'utiliser les primitives précédemment décrites pour communiquer avec ce protocole: une fois l'adresse connue, les premiers octets de celle-ci peuvent être utilisés comme préambule, et seule l'*endianness* doit être corrigée pour correspondre aux caractéristiques du protocole. Pour identifier l'adresse, nous configurons un préambule arbitraire et remontons des blocs de données démodulées de grande taille, que nous parcourons afin d'identifier en leur sein des paquets valides et en extraire l'adresse associée. Nous avons ainsi pu détecter l'adresse et écouter le trafic émis par une souris sans fil Logitech, puis injecter des paquets malicieux destinés à provoquer des clics et des mouvements de souris arbitraires.

IV. Conclusion

Dans cet article, nous avons présenté une approche nous ayant permis de détourner le comportement d'une puce *Bluetooth* embarquée dans un smartphone pour la faire communiquer avec d'autres protocoles sans fil, démontrant la faisabilité de ce type d'attaques inter-protocolaires. Dans une perspective offensive, de telles fonctionnalités sont particulièrement intéressantes dans la mesure où elles permettent d'envisager des scénarios d'attaque complexes, tirant profit de la mobilité d'un tel équipement et de l'important déploiement du protocole *Bluetooth*. Par exemple, la compromission du téléphone d'un employé pourrait permettre à un attaquant de pivoter sur d'autres protocoles présents dans l'entreprise pour mener des attaques passives ou actives, telles que des injections de frappes claviers ou l'insertion d'un noeud malveillant au sein d'un réseau *Zigbee*. L'impact de scénarios de ce type nous semble potentiellement critique, et il nous semble pertinent de repenser les stratégies défensives existantes pour prendre en compte ces nouvelles menaces.

References

[1] R. Cayre, F. Galtier, G. Auriol, V. Nicomette, M. Kaâniche, and G. Marconato, "Wazabee: attacking zigbee networks by diverting bluetooth low energy chips," in *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2021.

[2] D. Mantz, J. Classen, M. Schulz, and M. Hollick, "Internalblue-bluetooth binary patching and experimentation framework," in *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, 2019, pp. 79–90.

[3] T. Goodspeed, S. Bratus, R. Melgares, R. Shapiro, and R. Speers, "Packets in packets: Orson welles' in-band signaling attacks for modern radios." in *WOOT*, 2011, pp. 54–61.