Faire évader secrètement des données d'un réseau BLE sécurisé

Timothy Claeys[1], Franck Rousseau[1], Boris Simunovic[2] et Bernard Tourancheau[1]

[1] *Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, Grenoble France*
[2] *Univ. Grenoble Alpes, Grenoble INP-Esisar, Valence, France*

Ce papier présente une technique pour faire évader des données via un réseau à synchronisation temporelle sans nécessiter de privilèges particuliers sur l'appareil faisant échapper le secret. Pour cela nous proposons l'utilisation de la dérive d'horloge causée par la température de l'oscillateur. Nous proposons une implémentation en utilisant un réseau BLE, *Bluetooth Low Energy*, qui fonctionne avec divers équipements, dont des téléphones mobiles.

Mots-clefs : réseau sans fil, MAC synchronisé, BLE, sécurité, canal caché

1 Introduction

Les connexions radio sont aujourd'hui disponibles dans de nombreux appareils usuels, notamment les téléphones mobiles et leurs nombreux périphériques sans fil Bluetooth. Beaucoup reposent sur la variante BLE, *low energy*, très efficace en énergie, qui autorise une grande autonomie de ces périphériques. L'émergence de l'Internet des Objets renforce encore la présence de ces équipements. Les bonnes performances énergétiques de BLE reposent sur un cycle d'activité très faible grâce à une couche radio à synchronisation très fine, et rendue robuste dans la bande libre des 2.4 GHz en utilisant le saut de fréquences sur 40 canaux. Leurs capacités de communication font de ces équipements une cible de choix pour les attaquants, surtout qu'ils collectent et/ou contiennent parfois des données sensibles sur leurs utilisateurs.

Dans ce papier, nous présentons la réalisation d'un canal caché thermique reposant sur BLE. Les canaux cachés permettent à des équipements complices d'échanger secrètement des données. Cela permet par exemple de faire fuiter des clés de chiffrement sans déclencher de système d'alerte ou éveiller les soupçons. Cela fait, il est possible de continuer l'attaque ou l'espionnage de manière bien plus simple.

2 Principe et scénario d'attaque

Pour offrir de bonnes performances énergétiques, le protocole BLE synchronise très fortement les transmissions entre équipements. Cela permet d'éteindre la radio et éventuellement de se placer en mode de sommeil pour économiser l'énergie entre les transmissions. Pour se réveiller au même instant, les équipements synchronisent précisément leurs horloges internes et échangent périodiquement des paquets qu'il y ait ou non des données applicatives. Nous exploitons cette synchronisation précise pour établir un canal caché entre une *victime* compromise et un *récepteur* malveillant. Notre technique repose sur un principe similaire à celui proposé précédemment pour établir des attaques par canal auxiliaire [Mur06]. L'idée essentielle consiste à utiliser la chaleur émise par le processeur pour perturber l'oscillateur utilisé par la radio et agir ainsi indirectement sur le temps inter-paquet de transmissions BLE consécutives. Contrairement aux travaux cités plus haut, l'attaquant n'interagit pas directement avec la victime. Nous utilisons le trafic d'une connexion BLE légitime existante vers un appareil tiers, que nous dénommons *helper*, pour y cacher notre canal de transmission. L'attaquant mesure la variation des temps inter-paquet en écoutant passivement le trafic entre la victime et son périphérique, comme illustré FIGURE 1. Dans ce scénario, la victime et le récepteur sont contrôlés par l'attaquant et le *helper* est quelconque. Nous supposons que la victime a installé une application tierce qui aurait été corrompue, dans notre cas une application de contrôle pour ampoule

BLE. Cette application a accès à des données sensibles mais ne peut pas utiliser le réseau pour les transmettre directement au récepteur, ce qui est le cas avec BLE par exemple, qui demande un appairage sécurisé. Sans autorisations spécifiques sur l'équipement, juste en contrôlant précisément des tâches gourmandes en puissance de calcul, il est possible de transmettre de l'information sur le canal caché, en l'encodant dans les variations de dérive d'horloge.

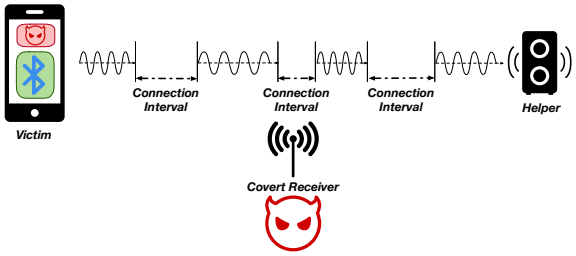

FIGURE 1: Scénario d'attaque. Une application malveillante lance des tâches intensives à des instants bien choisis pour faire chauffer l'appareil. Le récepteur mesure la dérive d'horloge induite entre les paquets transmis. (Icônes de The Noun Project)

3 Protocole à canal caché

Encodage et décodage. Le canal de transmission caché est unidirectionnel, pour cette raison nous utilisons de la correction d'erreur pour pallier l'absence de retransmissions. Le protocole de transmission repose sur une modulation tout-ou-rien, OOK. Pour encoder le bit '1', une forte variation de la dérive d'horloge est provoquée en faisant chauffer l'oscillateur à l'aide de lourds calculs RSA sur le processeur. L'application malveillante doit maintenir l'activité un temps suffisant pour que la chaleur se propage jusqu'à l'oscillateur. Lorsque la charge du processeur est relâchée, la température retombe et la dérive d'horloge retourne à sa valeur intiale. Pendant ce temps, le récepteur caché mesure et archive les temps inter-paquets de la transmission en cours.

Une fois la transmission terminée, l'attaquant utilise un filtre passe-bas pour extraire la dérive d'horloge $S(t)$. En calculant la dérivée $\frac{dS}{dt}$ sur le signal filtré il est possible de détecter les variations soudaines. Lorsque la valeur absolue de la dérive est supérieure à un seuil prédéfini V_{th}, un bit '1' est décodé, sinon il s'agit d'un '0', cf. Eq. 1. La valeur du seuil V_{th} est calculée à partir de l'écart type (σ) de $\frac{dS}{dt}$ et d'un facteur α déterminé expérimentalement, Eq. 2.

$$\text{Bit string}(t) = \begin{cases} 1 & |\frac{dS}{dt}| \geq V_{th} \\ 0 & |\frac{dS}{dt}| < V_{th} \end{cases} \quad (1) \qquad V_{th} = \sigma(\frac{dS}{dt}) * \alpha \quad (2)$$

Le début d'une transmission est détecté à l'aide d'un préambule constitué d'un seul bit a '1' dans nos expériences, le bit '1' étant le seul à pouvoir être clairement détecté par le décodeur.

Étalonnage du canal. Avant de pouvoir transmettre, l'application malveillante doit étalonner le canal. La température de base T_b est mesurée, elle correspond à la température lorsque le processeur est au repos. Ensuite, la température cible T_t est définie ainsi que la période Δt_h pendant laquelle la température T_t doit être maintenue pour encoder un '1'. Enfin, l'application doit connaître le temps de refroidissement Δt_c nécessaire pour que le système retrouve la température de base T_b et que la dérive d'horloge se stabilise à la valeur initiale. La durée d'un bit est dérivée de ces valeurs et doit être supérieure à $\Delta t_h + \Delta t_c$.

Correction d'erreur et embrouillage des données. Pour avoir une transmission robuste, l'application malveillante utilise de la correction d'erreur et embrouille les données. Par exemple, la correction d'erreur utilisée avec le Raspberry Pi est un code BCH(63, 31) qui ajoute un maximum de 31 bits de contrôle aux 32 bits de données. L'embrouillage permet d'éviter les longues suites de '0' et de '1', ce qui est indispensable pour éviter qu'une longue suite de '1' maintienne la température trop élevée trop longtemps, ce qui aurait pour conséquence de saturer la sensibilité de l'oscillateur aux variations de température. Nous empruntons la méthode utilisée par Bluetooth [Blu16] en utilisant le même LFSR : $y(x) = x^7 + x^4 + 1$.

4 Évaluation et performances

Méthodologie. Chaque expérience reproduit le scénario de la FIGURE 1. Les expérimentations ont été menées dans un bureau à température ambiante. L'appareil tiers utilisé est une simple ampoule LED connectée BLE [UU]. Le récepteur malveillant est un ordinateur équipé d'un Ubertooth One [Gad18]. L'Ubertooth est capable de suivre une connexion BLE spécifique s'il parvient à écouter le *handshake* initial pour en récupérer les paramètres temporels et de saut de fréquence. Nous avons implémenté l'attaque sur trois plateformes matérielles victimes contenant l'application malveillante, le Raspberry Pi 3B et deux téléphones, un Moto X 2^{e} gén. ainsi qu'un iPhone 5s. Nous présentons ci-dessous uniquement les résultats sur le Raspberry Pi qui sont les plus faciles à présenter, les résultats complets sont disponibles dans le papier original [CRST19].

Avant de mettre en place l'attaque, il faut une phase d'enquête sur l'appareil victime pour déterminer ses caractéristiques thermiques. L'objectif est d'estimer la température au repos, l'augmentation de température qu'il est possible de générer, la vitesse de propagation thermique dans l'appareil jusqu'à l'oscillateur de la radio, ainsi que la vitesse de dissipation. Nous avons aussi vérifié l'échelle de la dérive due aux variations de température, les grandes variations sont caractéristiques d'un cristal non-compensé, et les petites d'un cristal compensé, TCXO.

TABLE 1: Étalonnage du canal caché pour le RPI-3B.

Constante	Valeur	Remarque
T_b	39 °C	Température processeur au repos
T_t	47 °C	$T_t + 1$ pour chaque bit à '1' consécutif
Δt_h	10 s	
Δt_c	65 s	Temps de refroidissement : $T_t \rightarrow T_b$

Raspberry Pi 3B. Cet ordinateur sur carte repose sur un processeur quad-core Broadcom BCM2835 64 bits à 1.2 GHz et un circuit radio Wi-Fi et BLE BCM43438 [Cyp18]. La distance entre le processeur et l'oscillateur est approximativement de 1.5 cm. Nous supposons que le BCM43438 utilise un oscillateur non-compensé bien qu'il ne soit pas possible de s'en assurer. Dans un premier temps nous avons analysé les caractéristiques thermiques de la carte en suivant en temps réel la température de l'oscillateur à l'aide d'un thermocouple tout en faisant varier la charge processeur. De ces informations nous dérivons les valeurs d'étalonnage du canal en TABLE 1.

Performances. La FIGURE 2 présente une transmission cachée d'environ 1 h. La dérive brute est mesurée à l'aide du Ubertooth et stockée, puis le signal est filtré. Le second graphe présente la dérivée du signal filtré, et cette même courbe en mettant à 0 les valeurs négatives. On peut constater que les pics correspondent aux brusques variations de la température qui est représentée sur le graphe suivant. Finalement le graphe du bas montre les bits décodés grâce aux équations 1 et 2. Le débit du canal est approximativement de 40 bit/h.

Discussion. Le débit du canal est dépendant de plusieurs facteurs. Cela comprend le temps de montée et de descente en température de l'oscillateur ainsi que le taux d'erreur de transmission. Ces paramètres dépendent directement de l'architecture matérielle de la victime et de la distance entre le processeur et l'oscillateur. Un autre facteur important est la sensibilité de l'oscillateur à une brusque augmentation de la température. Sur du matériel haut de gamme comme les téléphones, ces variations sont fortement estompées par l'utilisation d'oscillateurs compensés, ce qui cause une forte diminution du débit. Sur le Moto X il tombe à 6–8 bit/h.

Comparé aux travaux de Matsi [MRR^{+}15], notre canal a un débit plus faible. Les auteurs utilisent directement la température du processeur, ce qui est plus réactif, cependant la portée est très faible puisque cela se limite aux applications s'exécutant sur une même machine physique. Les travaux sur Bitwhisper [GMME15] permettent d'atteindre une portée du canal thermique de 40 cm. Notre approche offre un débit similaire mais avec une portée bien supérieure, puisque limitée par l'utilisation de BLE, au moins plusieurs dizaines de mètres. De plus il n'est pas besoin d'être en ligne de mire ni d'interagir de quelque manière avec la victime, ce qui permet une discrétion maximale de l'attaque, il suffit d'écouter passivement les messages dans l'air.

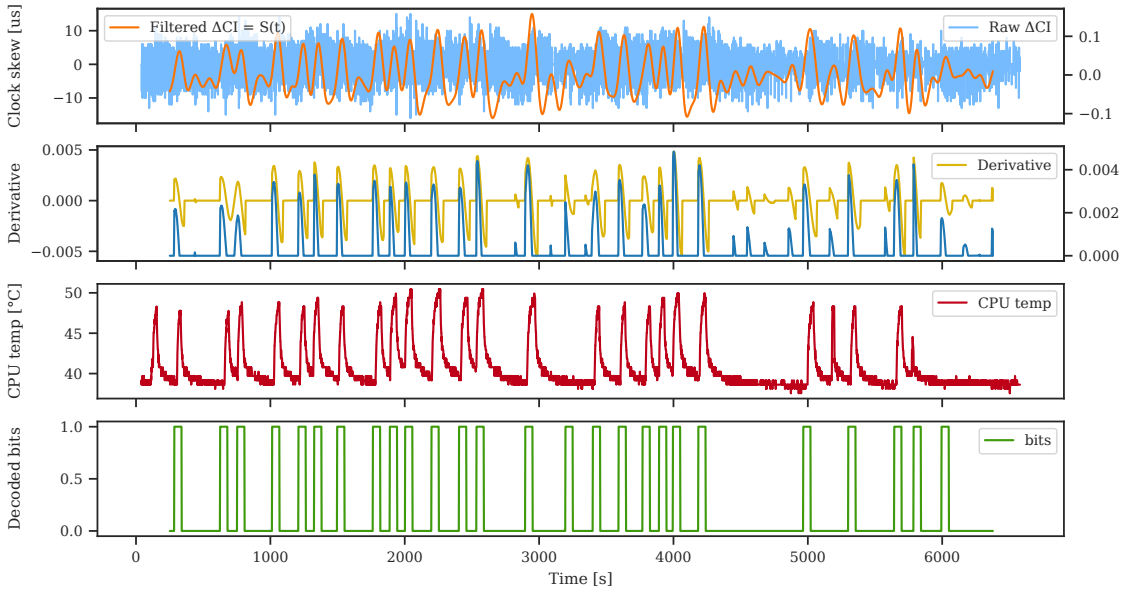

FIGURE 2: Implementation du canal caché sur un Raspberry PI 3B. De haut en bas, dérive brute et filtrée, dérivée de la dérive filtrée, température du processeur et bits décodés.

5 Conclusion

Nous avons présenté la mise en œuvre d'un canal caché thermique permettant de faire fuiter des données sensibles de manière discrète sans disposer d'autorisations spécifiques ni d'accès au réseau sur l'appareil victime. La variation de charge du processeur nous permet de moduler les informations qui sont reçues par un appareil complice. Le débit obtenu est comparable aux travaux précédents, mais nous augmentons considérablement la portée jusqu'à la portée de BLE, contrairement aux travaux précédents limités par la portée de diffusion de la chaleur, soit quelques dizaines de centimètres tout au plus.

Références

[Blu16] Bluetooth SIG. *Bluetooth Core specification 5.0*, 12 2016.

[CRST19] Timothy Claeys, Franck Rousseau, Boris Simunovic, and Bernard Tourancheau. Thermal Covert Channel in Bluetooth Low Energy Networks. In *ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, Miami, FL, USA, May 2019.

[Cyp18] Cypress Semiconductor Corporation. *CYW43438 : Single-Chip IEEE 802.11 b/g/n MAC/Baseband/Radio with Integrated Bluetooth 4.2*, 07 2018.

[Gad18] Great Scott Gadgets. Project Ubertooth, 2018.

[GMME15] Mordechai Guri, Matan Monitz, Yisroel Mirski, and Yuval Elovici. BitWhisper : Covert Signaling Channel between Air-Gapped Computers using Thermal Manipulations. In *28th IEEE Computer Security Foundations Symposium (CSF)*, pages 276–289, 2015.

[MRR+15] Ramya Jayaram Masti, Devendra Rai, Aanjhan Ranganathan, Christian Müller, Lothar Thiele, and Srdjan Capkun. Thermal Covert Channels on Multi-core Platforms. In *USENIX Security Symposium*, pages 865–880, 2015.

[Mur06] Steven J. Murdoch. Hot or Not : Revealing Hidden Services by their Clock Skew. In *13th ACM conference on Computer and communications security*, pages 27–36. ACM, 2006.

[UU] Magic Blue UU. Bluetooth Bulb.