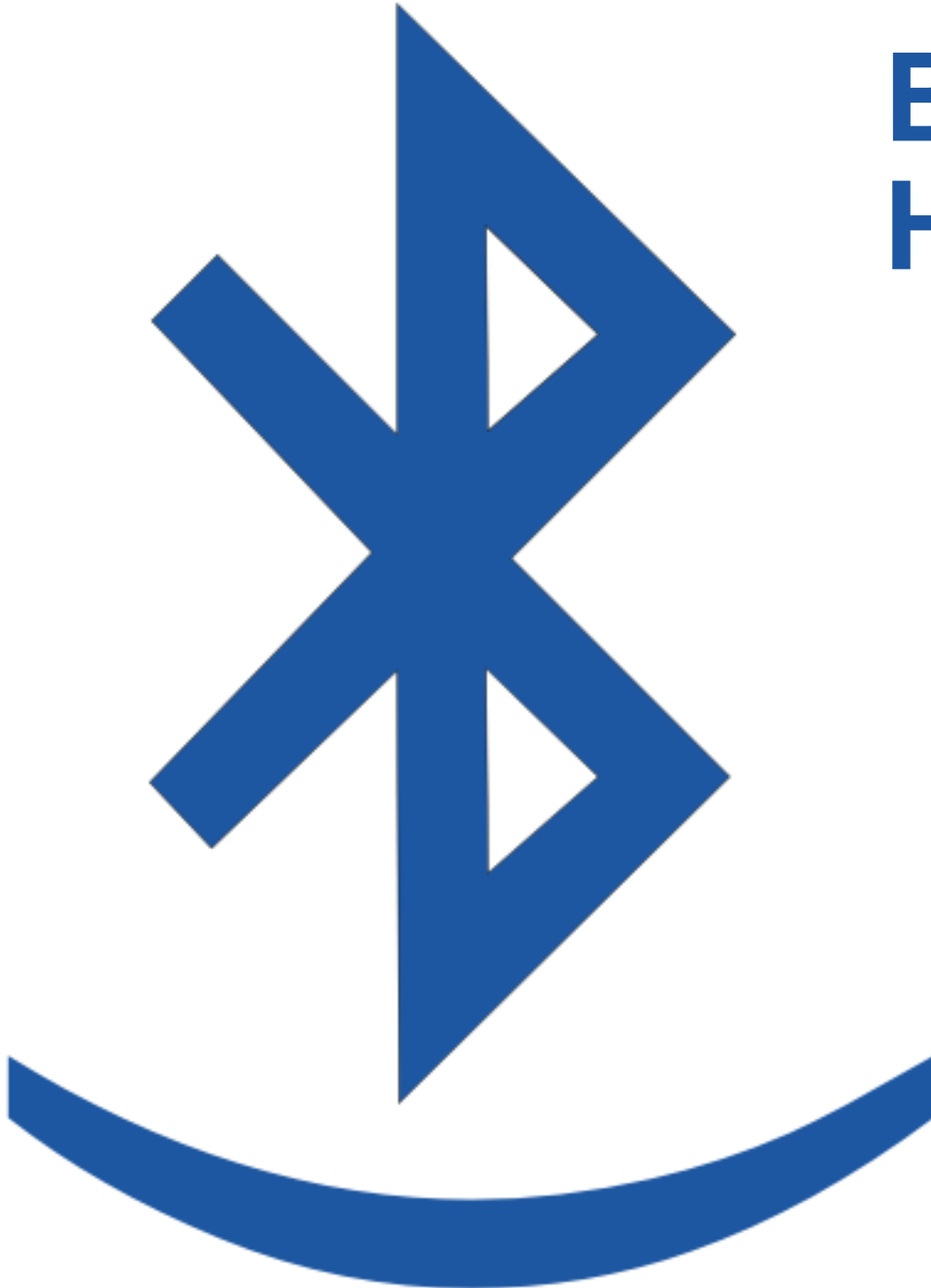


# Bluetooth Hacking



# Bluetooth Hacking

---

- Historia
- Introducción
- La stack de GNU/Linux: BlueZ
- Aplicaciones
- Seguridad en bluetooth
- Ataques (...)
- Ejemplos

^D

# Historia

---

- El nombre viene del rei Vikingo Danés y Noruego "Harald Blâtand", que consiguió convertir al cristianismo las tribus noruegas, suecas y danesas (aka estandarizar).
- Blâtand en danés significa: "blâ": piel oscura, y "tan": buena persona.
- Los ingleses tradujeron el nombre a "bluetooth". \*logico\* :P
- En 1994, Ericsson inició un estudio para investigar la viabilidad de una interficie via radio de bajo coste para el consumo masivo que permitiese comunicar dispositivos mobiles entres si.
- El "SIG" (Special Interest Group), son un conjunto de compañías interesadas en el desarrollo de esta tecnologia, entre ellas encontramos a Ericsson, Intel, IBM, Toshiba, Nokia.
- En 1999 el SIG aceptó a Microsoft, Lucent, 3COM y Motorola. (actualmente tienen más de 2000 miembros)

# Introducción

---

- Bluetooth es un estandar de comunicaciones inalambricas con muchas similitudes con el wireless, pero orientado a tareas especificas para dispositivos embebidos. Parecido al USB.
- Trabaja a 2.4GHz (igual que el wireless, genera ruido).
- Utiliza "frequence-hopping" basado en el clock-offset, una llave aleatoria y otra predefinida por el usuario (PIN1)
- Diseñado para trabajar a distancias cortas y intercomunicar dispositivos (redes PAN).
- Permite conexiones serie a servicios publicados por SDP enlazados a un canal. Podriamos comparar los canales a los puertos TCP, podemos publicar o no publicar los servicios, aunque sean accesibles
- Algunos de los servicios más habituales: OBEX, OPUSH, HID, SERIAL, HEADSET, HANDSFREE, DIALUP, FAX, IMAGING, UPNP. (no tienen porque estar en el mismo canal, ya que esto lo especifica SDP)

# Un poco de numeros...

---

- Clases: 1 (100m), 2 (40m), 3 (10m)
- BT 1.0 = 751 Kbit/s
- BT 1.2 = 2.1Mb/s
- 79 canals
- 1600 hops/s
- Piconets de hasta 8 dispositivos ( 1 master, 7 slaves ). > Scatternet.

# BlueZ

---

- En GNU/Linux tenemos la más potente implementación libre de bluetooth: BlueZ.
- Generalmente la programación de bluetooth suele ser a bajo nivel, ya que ha crecido demasiado rapido y es un estandar demasiado extensible, cosa que ha imposibilitado una api común y estandar
- Esto nos permite un control muy bueno sobre los dispositivos:

```
$ hcid # Levantamos el demonio HCI
# levantamos la interficie + inq.visib. + sdp.visib.
$ hciconfig hci0 up iscan pscan
$ sdpd # Levantamos el dimonio de SDP.
$ hcidump & # ponemos el sniffer en bg.
$ hcitool scan # Inq. de dispositivos.
```

# BlueZ::Demonios

---

## - hcid

- Siglas: Host Controller Interface Daemon.
- Sirve para inicializar las interfaces bluetooth (hci#, ubt#, ..)
- Configuración en el `${SYS.CONFDIR}/bluetooth/hcid.conf`
- Lo que hace hcid podemos implementarlo con un shellscript llamando a hciconfig.

## - sdpd

- Siglas: Service Discovery Protocol Daemon.
- Sirve para publicar los servicios disponibles
- Se configura con el comando 'sdptool'

```
$ sdptool add --channel=9 OPUSH
```

- El comando nos permite escanear servicios de otros dispositivos, buscar dispositivos que publiquen cierto servicio.

```
$ sdptool browse
```

```
$ sdptool --bdaddr ${BDADDR} OPUSH
```

# BlueZ::Tools

---

## - rfccomm

- Nos permite enlazar los devices `/dev/rfcomm#` a un BDADDR y canal para acceder a ellos como si fuese un puerto serie. Tambien nos permite escuchar en un canal concreto y esperar a que se realice una conexión
- Facilmente manejable en shellscripts para crear automatizaciones o servicios

## - hcidump

- Es el equivalente a tcpdump sobre bluetooth.
- Sólo se puede capturar trafico local, debido al frequency-hopping. Hay hardware especializado que permite descifrar la llave a partir de capturas de señal en todos los canales posibles. Este hardware es caro y difícil de conseguir. A parte bluetooth también soporta encriptación por hardware. Así que es complicado..

# BlueZ::Tools (2)

---

## - l2ping

- Permite enviar paquetes a nivel 2 (L2CAP), es el equivalente al comando 'ping' en ICMP.
- Podemos deducir la carga del sistema (en dispositivos móviles) o la /distancia/ relativa a un dispositivo por los tiempos de respuesta.
- Puede ser útil para monitorizar la presencia de un dispositivo dentro de un rango de acción (bluemon)

## - bluepin

- Callback del hcid que se ejecuta cuando recibimos una conexión y nos pide autenticación con PIN1.
- Puede ser un shellscript que escriba por stdout la string "PIN:0000" o "" en el caso de cancelar la operación.

# Aplicaciones

---

- Hay bastantes aplicaciones libres que nos permiten manejar servicios de bluetooth y hasta algunos que extienden los protocolos o /hacks/

## **Bluemon - (\*nix)**

Bloqueo de las X en funcion de la distancia de un dispositivo bluetooth (hace l2ping hasta que llega a un limite predefinido).  
puede ser usado para bloquear servicios en funcion de la distancia de una persona, PDAs para empresarios, cuerpos de seguridad, etc.  
(RFID like)

## **p3nfs - (sys)**

Permite montar por NFS sobre bluetooth dispositivos Psyloc o Symbian.

## **jemote - (j2me)**

Control remoto de mplayer, tambien compatible con otras aplicaciones como imposter (para ver diapos de OO).

# Aplicaciones (2)

---

## **BtBrowser - (j2me)**

Nos permite buscar dispositivos bluetooth, ver los servicios que publican, a que canales y ver si el dispositivo tiene encriptación, autenticación y las URLs btsp.

## **BlueMote - (\*nix)**

Programa que cambia el profile del dispositivo para hacer pasar por un manos libres y recibir comandos desde un movil (control de volumen etc) y lanzar shellscripts para cada acción

# Aplicaciones (3)

Ethernet sobre Bluetooth: Podemos utilizar BNET, o utilizar el bluetooth com si fuese una tarjeta de red o simplemente crear un rfcomm y montar un pppd encima. El rendimiento es de unos 50KB/s. Podemos montar nodos en plan AP.

```
$ cat bt-server
while : ; do hciconfig hci0 up ;
rfcomm release all ; echo "Waiting..."
rfcomm listen /dev/rfcomm0 ${channel}
while : ; do if [ -n "`rfcomm`" ]; then
pppd /dev/rfcomm0 10.0.0.1:10.0.0.2 noauth
nodetach novj debug 115200 local
break ; fi ; sleep 1; done ; done
```

```
$ cat bt-client
rfcomm bind 0 ${bdaddr} ${chan}
pppd /dev/rfcomm0 10.0.0.2:10.0.0.1 defaultroute
noauth novj debug 115200
```

# Ataques

---

- Partiendo de la base que todo sistema tiene sus puntos debiles, bluetooth tampoco se puede liberar. Han surgido decenas de ataques diferentes a diferentes capas. Algunas explotadas gracias a defectos de la stack, otras en la gestión de la autenticación, conversiones de encoding o la implementación de un servicio en concreto.
- Bluetooth esta bien diseñado, asi que la mayoría de vulnerabilidades no son por culpa del bluetooth en si (como pasa con wireless o voip), sino en la implementación que ofrecen ciertos fabricantes. Estas vulnerabilidades pueden estar a dos niveles:
  - Enlace: Gestión de conexiones, autenticación (PSM), spoofing.
  - Aplicaciones: Implementación a nivell de usuario de OBEX, OPUISH.
- Muchos de los /exploits/ reciben nombres distintos aunque suelen tener la misma finalidad: acceso al puerto serie del dispositivo, extraer información de la agenda, llamar, pinchar conversaciones de manos libres, colgar el SO.

# Ataques::BlueSmack

---

- Aka El besito de la muerte.
- Seguramente recordareis el /ping-de-la-muerte/ de winblows... Este es el equivalente en bluetooth :)
- Esta vulnerabilidad la sufren algunos dispositivos embebidos, con una stack mal implementada que gestiona mal los paquetes L2CAP de tamaño no estandar

```
$ l2ping -f -s 667 ${bdaddr}
```

# Ataques::HelloMoto

- Reciclando la coña del anuncio de Motorola...
- Es una vulnerabilidad **\_MUY\_** grave en la stack de algunos dispositivos motorola (<2005). Cuando el dispositivo recibe una conexión y esta en estado 'stablished' convierte el dispositivo en modo 'promisc' y acepta cualquier conexión por cualquier otro canal siempre que esa conexión esté establecida.
- El único servicio que nos permite crear una conexión con un dispositivo sin autorización del usuario es "OPUSH". En este canal, el dispositivo recibe una conexión y se queda esperando a que el cliente le envíe los datos del fichero que va a enviar.
- Si hacemos la conexión al OPUSH y no enviamos ningún dato, podremos realizar cualquier conexión paralela sin autentificación O:)

```
$ rfcomm connect rfcomm0 ${bdaddr} ${opush-channel} &  
$ rfcomm connect rfcomm0 ${bdaddr} ${serial-channel}
```

# Ataques::BlueSnarf

---

- El objetivo de BlueSnarf es usar OPUSH para usar PULL para leechear la agenda de direcciones y demás ficheros sin la autorización del propietario.
- Esto podemos hacerlo de distintas formas:
  - OBEX Push Profile (OPP)
  - Serial port. (ATZ, AT+CPBS="ME", AT+CPBR="1,100", AT+CMGL, AT+CMGR=1,100, ATI1, ATI2..)
  - ObexFTP: telecom/pb.vcf, telecom/cal.vcs

```
$ obexftp -b ${bdaddr} -B 10 -g telecom/pb.vcf >  
/tmp/${bdaddr}.phonebook
```

# Ataques::BlueSnarf++

---

- Esta es una variación del BlueSnarf que usa OBEXFTP para extraer la agenda o modificar ficheros.
- Algunos ericsson son vulnerables de serie a través de obexftp, nos permiten sobrescribir la agenda de contactos sin autorización del usuario.

# Ataques::BluePrinting

---

- Consiste en identificar un dispositivo a través de información que podamos extraer de él sin realizar ninguna conexión. Es parecido al OS fingerprinting.
- Estos datos se pueden extraer de distintos sources:
  - BDADDR (Como les MACs en las NICs)
  - SDP (Servicios exportados)
    - Cuantos servicios, que servicios, numero de canal asignado a cada servicio.
- La gente de trinite tienen publicada una aplicación escrita en perl que funciona fatal ("bp" -> blueprinting). BP no funciona, pero es relativamente sencillo escribirlo bien.
- Una vez sepamos cual es el dispositivo, podremos usar esa información para guiarnos y afinar más en ciertos tipos de ataques si sabemos que sufre cierta vulnerabilidad

# Ataques::PIN1-Cracking

---

- Consiste en deducir un código PIN de un dispositivo. Podemos hacerlo de dos formas:
  - pasivo por spoofing: Se sustituye a un dispositivo (bdaddr y/o name) y esperamos a que la victima cliente se conecte a nosotros y introduzca el PIN. Nuestro dispositivo debe ser más visible que la victima para que todo vaya bien (bdaddr)
  - pasivo por análisis de radiofrecuencia: bastante complicado. Ref: <http://www.eng.tau.ac.il/~yash/shaked-wool-mobisys05/index.html>
  - activo: consiste en hacer fuerza bruta y probar PINs desde 0000 hasta 9999, a unos 10 segundos entre PIN y PIN, tardaremos como máximo unas 277 horas (11 dias) en descubrir el PIN.
- Ciertos dispositivos pasivos (GPS, HeadSets) tienen ciertos PINs por defecto: 0000, 1234, 9999

# Ataques::BlueJack

---

- Consiste en diferentes tecnicas para enviar mensajes a dispositivos sin la autorización del usuario
  - Cambiar el nombre del dispositivo (< 248 caracteres)
  - Enviar VCards manipuladas (algunos dispositivos (pda, móviles) visualizan la información cuando la reciben y preguntan al usuario para añadirla a la agenda).
  - Podem suplantar la identidad de un nodo /oficial/ (btgracia, nikebt) ofreciendo el 'mismo' servicio que ellos.
  - La finalidad de este ataque es principalmente con fines terroristas como causar panico a la masa social, engañar, desinformar, infectar...

# Ataques::BlueBug

---

- Es una variante específica de snarfing consistente en utilizar comandos AT.
- El acceso al puerto serie de un móvil es muy parecido al de un modem, solo que tenemos unos cuantos comandos AT de más.
- Los comandos AT-GSM nos permiten obtener información del usuario y del dispositivo (nivel de batería, cobertura, model, numero de telefono...)
- Tambien podemos utilizarlo para llamar a través de ese dispositivo, definir redirects, extraer registros de la agenda, enviar sms, ...

Bloover

[http://trifinite.org/trifinite\\_stuff\\_bloover.html](http://trifinite.org/trifinite_stuff_bloover.html)

GSM-AT Guide:

<http://news.nopcode.org/pdf/GSM-AT-Guide.pdf>

# Ataques::BlueBump

---

- La palabra "bump" viene del mundillo del /lockpicking/. Las llaves 'bump' son llaves maestras con todos los dientes del mismo tamaño y separación. Con un golpe o meneo podremos situar las levas en la posición correcta abriendo el cerrojo.
- En bluetooth esta técnica consiste en aprovechar una vulnerabilidad en el protocolo de autenticación en el cual un usuario acepta una conexión entrante. El cliente (atacante) puede enviar comandos de control para borrar la llave de encriptación y generar una nueva (hcitool key). Después de esto, el dispositivo tendrá vía abierta para conectarse tantas veces como quiera sin que la víctima tenga que aceptar la conexión.

# Ataque::NastyVCard

---

- Algunos dispositivos tiene un parser XML mal programado. Esto nos puede permitir provocar segmentaciones o ejecución de código en el parseo de VCARDs.
- Tambien podemos usar las VCARDs para enviar mensajes (BlueJack), ya que algunos dispositivos aceptan el vcard sin preguntar al usuario.

# Ataques::OSs

---

(WindowsMobile-dos) La stack de las PDAs con windows y bluetooth es vulnerable a un buffer overflow a través del servicio OPUSH. El servicio no controla un buffer overflow al leer el nombre del archivo que vamos a enviar, esto nos permite ejecutar código o provocar una denegación de servicio.

(Symbian-reboot) Symbian tiene un problema en la conversión de caracteres. El caracter <tabulador> no se transforma correctamente y provoca una segmentación+reboot del dispositivo

```
$ hciconfig hci0 name "`printf "\t."`" (\x09\x0a)
```

Podemos configurar un dispositivo con este nombre invalido (phyle para symbian) y llevar el bluetooth encendido en el bolsillo.

[http://cvs.nopcode.org/cgi-bin/cvsnop/s60/src/py/phyle\\_editor.py](http://cvs.nopcode.org/cgi-bin/cvsnop/s60/src/py/phyle_editor.py)

# Ataques::OSs

---

(OSX-path-transversal) En febrero salieron algunos gusanos para OSX. Uno de ellos utiliza una vulnerabilidad del daemon de OBEX que nos permite subir archivos con nombre ".././../tmp/xx". El gusano sube el troyano en un directorio que el sistema usa en el arranque para asegurarse la permanencia en el sistema. (fixed)

```
$ ussp-push ${bdaddr}@${chan} ./vx.out  
" / .%2E/ .%2E/%2E%2E/%2E%2E/tmp/vx "
```

# Ataques::BlueWhisper

---

- Consiste en pinchar una conexión de manos libres o headsets sin la autorización del usuario e inyectar o recibir audio.
- Aquí podemos ver un longshot con antena desde un puente. A 120Km/h tenemos tiempo suficiente para enlazar el dispositivo e inyectar el audio con sox o escuchar.
- Algunos dispositivos al ser tan limitados solo activan la visibilidad del dispositivo durante 10 segundos (redfang). Y usan PINs preestablecidos e inmutables (0000, 1234, 9999).

# Ataques::BlueWhisper (2)

- Aqui podemos ver un grafico de la sitacuión del puente y la antena :P



Ref: [http://trifinite.org/blog/archives/2005/07/introducing\\_the.html](http://trifinite.org/blog/archives/2005/07/introducing_the.html)

# Ataques::BlueWhisper (3)

---

- Si conseguimos hacer una conexión al puerto serie de un móvil, tendremos control absoluto sobre el móvil. Enviar sms, consultar agenda, llamar. Hay servicios que nos permiten trackear un móvil a partir de un sms. O:)
- Situados en un puente dispondremos de unos 15 segundos de cobertura sobre los vehículos a una velocidad mediana. Tiempo suficiente para realizar la conexión y enviar los comandos para pinchar el manos libres y trackear el vehículo.
- También podemos hacer bluedriving desde un coche, para situarnos al lado de otro vehículo con bluetooth y tener más tiempo para jugar.
- También podemos usarlo para comunicar dos vehículos (moto, coche) sin cables. Cruzando las conexiones de los manos-libres.

# Ataques::BTSpam

---

- La difusión masiva de mensajes a través de bluetooth es una tecnica utilizada por muchas empresas en sitios publicos (nike, aquarius, gracia), o en recintos cerrados como meetings, congresos, etc..
- Se utilitza para distribuir propaganda de forma automatica. Futurlink (empresa de bcn) ofrece un servicio que funciona sobre windows móvil y es vulnerable (TFTP-over-bluetooth + wmobile-opush-overflow (ejecución de código). Ahora son GEODEs con GNU/Linux y 6 dongles
- Se ha utilizado también en concentraciones /manifestaciones con tal de distribuir mensajes. Esto se puede hacer en un simple shellscript o un programa en j2me para llevarlo en el bolsillo :)
- Con una antena podemos llegar a realizar conexiones de kilometros de distancia. Podemos usar antenas de wireless ya que funcionan a 2.4GHz.

# Ataques::BSS

---

BSS: Bluetooth Stack Smasher

- Esta es una herramienta que nos permite estresar la stack de un dispositivo para comprobar si es vulnerable, maneja distintos tipos de ataques tipo SYN flood, etc.
- Gracias a esta herramienta se han descubierto que algunos dispositivos no manejan bien ciertos estados de la stack pudiendo provocar un DoS

<http://www.secuobs.com/>

# Ataques::Redfang

---

Redfang: The bruteforce bt scanner

- Esta herramienta nos permite descubrir dispositivos que no sean visibles. Utiliza fuerza bruta sobre las BDADDR pidiendo el nombre del dispositivo. Ya que es la única información que sabemos que nos va a servir aunque esté oculto :) (aunque no debería).
- Para acelerar las búsquedas se pueden definir rangos de BDADDR de fabricantes conocidos. Esto reduce bastante la búsqueda, pero no deja de ser extremadamente lento un escaneo de estas características. El programa tarda unos 10 segundos por bdaddr, aunque se le puede definir un timeout.

<http://www.blackops.cn/tools/redfang.2.5.tar.gz>

# Ataques::Spoofing

---

- Enviando unos comandos al dispositivo bluetooth podemos cambiar la dirección física y por lo tanto falsificar dispositivos o romper enlaces entre dos dispositivos
- Si sustituimos al dispositivo que recibe la conexión. La stack de BlueZ registra en rfcomm como "closed" la conexión que acabamos de romper. Así podemos llegar a saber en que canal estaba conectado.
- Esta /feature/ la podemos encontrar en los sources de la herramienta "bluediving". No está documentada ni se instala con "make install".
- Teóricamente se puede llegar a robar una conexión establecida mediante algunos datos previos de los dispositivos y suplantando la identidad del cliente, pero todo depende de la stack del dispositivo servidor. Para ello necesitaremos el clock offset y el PIN1. No es sencillo ni está documentado

# Ataques::Spoofing (2)

- Algunas implementaciones dejan la stack en un estado incorrecto, pudiendo dejar inútil algún servicio o provocar un DoS/reboot. Algunas versiones viejas de Symbian son vulnerables.
- Aquí tenemos los comandos para cambiar la dirección física (bluez) para romper una enlace entre dos dispositivos.

```
$ ./bdaddr -i hci0 ${new-bdaddr}
$ hciconfig hci0 down
# extraer el dongle y volverlo a insertar.
$ hciconfig hci0 up
$ l2ping ${target-bdaddr}
```

# Referencias y ^D

---

Mi blog:

<http://news.nopcode.org/pancake>

GSM-AT Guide:

<http://news.nopcode.org/pdf/GSM-AT-Guide.pdf>

Trifinite:

<http://www.trifinite.org/>

BlueZ:

<http://bluez.sf.net/>

BlueDiving:

<http://bluediving.sf.net/>

Spanish Bluetooth Security Group:

<http://bluehack.endorasoft.es/>

Bluetooth Browser - J2ME

<http://benhui.net/modules.php?name=Bluetooth&page=DiscoveryMain>

# EOF

Contact: [pancake<pancake@phreaker.net>](mailto:pancake@phreaker.net)