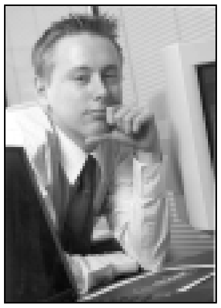
@stake℠

Research Report

OCTOBER 2003

War Nibbling: Bluetooth Insecurity

The Bluetooth protocol, which is deployed in millions of products ranging from cellular telephones to laptops, is quickly becoming the new standard for intra-device wireless communications. This paper examines methods of assessing the security of Bluetooth devices in relation to the protocol's design and implementation flaws. We will also discuss ways to proactively approach Bluetooth security and what security professionals can do to defend their organizations against unwanted compromise.



By Ollie Whitehouse
ollie@atstake.com

Ollie Whitehouse is a Director of Security Architecture for @stake, Inc. Mr. Whitehouse's experience in IT consulting includes security architectures, systems integration and project management. He has also worked for wireless operators in the US, Europe and the Middle East, where he has provided high level architecture assessments and policy development as well as mobile-based attack & penetration testing of 2.0, 2.5g and 3g networks.

Introduction

The topic of this paper is 'War-Nibbling', the process of mapping Bluetooth devices within an organization. War-Nibbling is similar to 'War-Driving' [1], but deals with smaller devices that rely on close proximity to communicate.

Before the concepts of War Nibbling are examined in detail, it is suggested that you have a basic understanding of Bluetooth [2] technologies. Therefore, you should keep the following information in mind.

The Bluetooth specification supports transmissions up to 100m with Bluetooth transceivers operating in the 2.4 GHz, ISM band; the same band WLAN devices use. Below are the documented classes of devices:

DEVICE CLASS	TYPE	STRENGTH	RANGE (METERS)
Class 1 Devices	High	100 mW (20 dBm)	Up to 100
Class 2 Devices	Medium	2.5 mW (4 dBm)	Up to 10
Class 3 Devices	Low	1 mW (0 dBm)	Well within 10

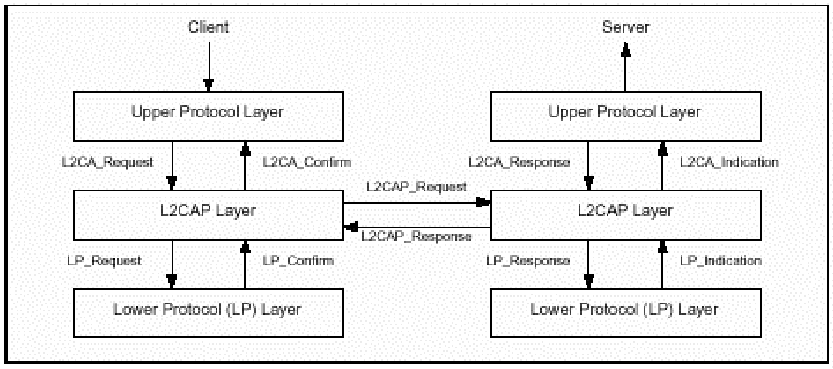
Currently the most common devices are Class 3 or 2. These include, but are not limited to:

- Cellular telephones
- Personal Digital Assistants
- Computer Peripherals (Keyboards and Mice)
- Audio Accessories
- Laptop Computers
- Access Points (PAN supports 8 devices with a PPP style connection)

Bluetooth is based upon the idea that a user should be able to create a PAN (Personal Area Network) around them. To facilitate client privacy Bluetooth contains a number of security features within the base protocol specification [3]. This includes Authentication, Authorization and Privacy.

For an overview of the Bluetooth protocol layers, refer to *Exhibit A*, which is a pictorial representation of the Bluetooth protocol stack.

Exhibit A: Overview of the Bluetooth protocol layers [18]



There are three primary security modes:

- Mode 1: No Security
- Mode 2: Application/Service based (L2CAP)
- Mode 3: Link-layer (PIN authentication/MAC address security/encryption)

Digging a little further into the specification we see the passage contained within *Exhibit B* below.

Exhibit B: Excerpt From the Bluetooth Official Specification v1.1 [4]

> *14.3 ENCRYPTION: User information can be protected by encryption of the packet payload; the access code and the packet header are never encrypted. The encryption is carried out with a stream cipher called E0 that is re-synchronized for every payload. The overall principle is shown in Figure 14.4 on page 159.*

Exhibit B indicates that Bluetooth headers are not encrypted, which can lead to a possible avalanche of potential attacks against the link layer. Discounting these attacks, we can look to the higher layers in the protocol stack and take advantage of other vectors. The techniques that will be covered in the paper are:

- Locating discoverable Bluetooth devices
- Locating non discoverable Bluetooth devices
- Enumerating service information

In addition we will look at the following defensive techniques:

- Hiding your devices
- Personal firewalls
- Bonding information checks

The purpose of this document is to make people aware of the close proximity attacks that can occur against any Bluetooth-enabled device, and to give insight into some of the future applications and extensions of such techniques in relation to assessments of Bluetooth deployments.

Hardware/Software Requirements

In order to perform the assessment, you need to have a computer which supports the Linux[5] operating system and a Bluez[6]- supported Bluetooth device. For example, the author successfully uses VMWare[7] on an IBM T30[8] with an additional TDK[9] USB based Bluetooth transceiver.

Looking for Discoverable Bluetooth Devices

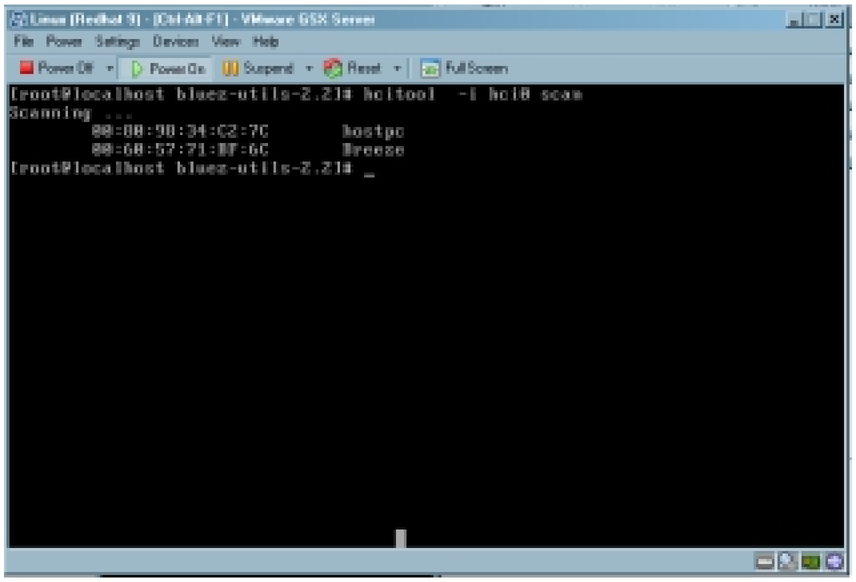
To begin, refer back to the introduction and understand the range that your assessment is going to have. Note that additional obstacles such as partitioning walls can limit your range in the same way that 802.11 is affected. The author has successfully performed an assessment in a 10 meter by 7 meter open plan office and has been able to locate over 25 devices in one scan.

Below is a quick high-level overview of how the Bluetooth protocol is split out:

```
--- L2CAP                            Logical Link Control and Adaptation
       |                             Protocol
       |
       +----- SDP                    Bluetooth Service Discovery Protocol
       |
       +----- RFCOMM                 Commports over RF
       |
       +----- TCS-BIN                Bluetooth Telephony Control   |
       |      |                             Specification
       |
       +----- TCS-BIN-CORDLESS       Bluetooth Telephony Control
       |                             Specification
       |
       +----- BNEP                   Bluetooth Network Encapsulation
       |                             Protocol
       |
       +----- HID_Control            Human Interface Device
       |
       +----- HID_Interrupt          Human Interface Device
       |
       +----- UPnP                   See [16]
       |
       +----- AVCTP                  Audio/Video Control Transport
Protocol
       |
       +----- AVDTP                  Audio/Video Distribution Transport

       |                             Protocol
       |
       +----- UDI_C-Plane            Unrestricted Digital Information
                                            Profile [UDI]
```

To conduct a scan of Bluetooth devices, which have been configured to be discoverable, perform the command located in *Exhibit C*.

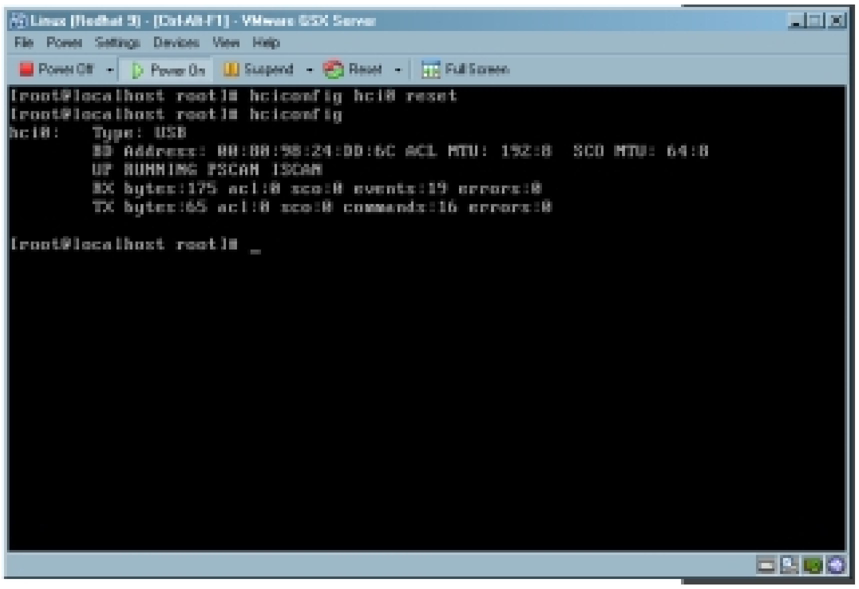
Exhibit C: Locate all Local Discoverable Bluetooth Devices


Linux (Redhat 9) - [Ctrl-Alt-F1] - VMware GSX Server
File Power Settings Devices View Help
Power Off
Power On
Suspend
Reset
Full Screen

```
[root@localhost bluez-utils-2.2]# hcitool  -i hci0 scan
Scanning ...
        00:80:98:34:C2:7C       hostpc
        00:60:57:71:BF:6C       Breeze
[root@localhost bluez-utils-2.2]# _
```

In the exhibit above, there are two (2) Bluetooth devices within range. It should be noted that similar commands for the Bluetooth interface appear when looking at normal network interfaces. For example, see *Exhibit D*.

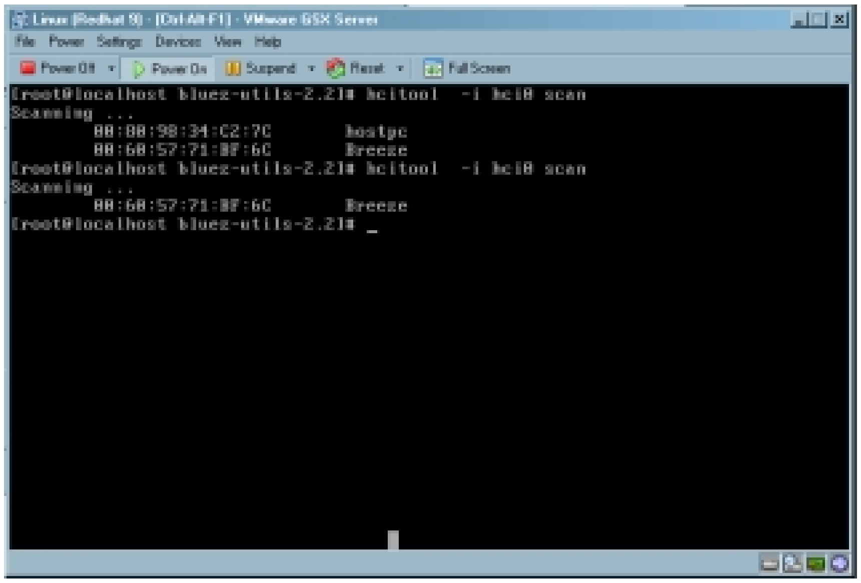
Exhibit D: Hciconfig Command Output


Linux (Redhat 9) - [Ctrl-Alt-F1] - VMware GSX Server
File Power Settings Devices View Help
Power Off
Power On
Suspend
Reset
Full Screen

```
[root@localhost root]# hciconfig hci0 reset
[root@localhost root]# hciconfig
hci0:   Type: USB
        BD Address: 00:80:98:24:DD:6C ACL MTU: 192:8  SCO MTU: 64:8
        UP RUNNING PSCAN ISCAN
        RX bytes:175 acl:0 sco:0 events:19 errors:0
        TX bytes:65 acl:0 sco:0 commands:16 errors:0

[root@localhost root]# _
```

If however during your assessment you find fewer devices than expected, such as in *Exhibit E*, this may be an indication that some of the Bluetooth devices are not configured as discoverable.

Exhibit E: Scan for Discoverable Bluetooth Devices Returns Only One Host


Linux (Redhat 9) - [Ctrl-Alt-F1] - VMware GSX Server
File Power Settings Devices View Help
Power Off
Power On
Suspend
Reset
Full Screen

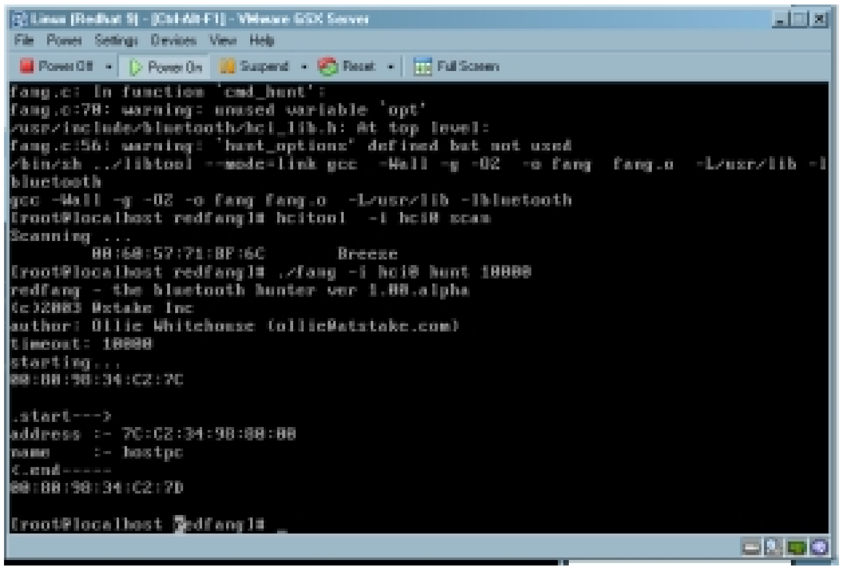
```
[root@localhost bluez-utils-2.2]# hcitool  -i hci0 scan
Scanning ...
        00:00:98:34:C2:7C       hostpc
        00:60:57:71:8F:6C       Breeze
[root@localhost bluez-utils-2.2]# hcitool  -i hci0 scan
Scanning ...
        00:60:57:71:8F:6C       Breeze
[root@localhost bluez-utils-2.2]# _
```

Locating Non-Discoverable Bluetooth Devices

So you have performed an assessment and most likely only found a few hosts, as is often the case. A large number of vendors disable all other security measures, but mark the device as non-discoverable with the hope that that this would, as the name implies, make the device non-discoverable. However, we applied general network principles (plus a little specification reading) and realized that devices that are marked non-discoverable should still in theory respond to direct name and services inquiry requests. Our initial logic at the time of making this assertion was that even non-discoverable authenticated/bonded devices require some mechanism to facilitate service and name updates.

Guess what the outcome of our testing yielded? Yes, a non-discoverable device will in fact still respond to these requests as *Exhibit F* shows.

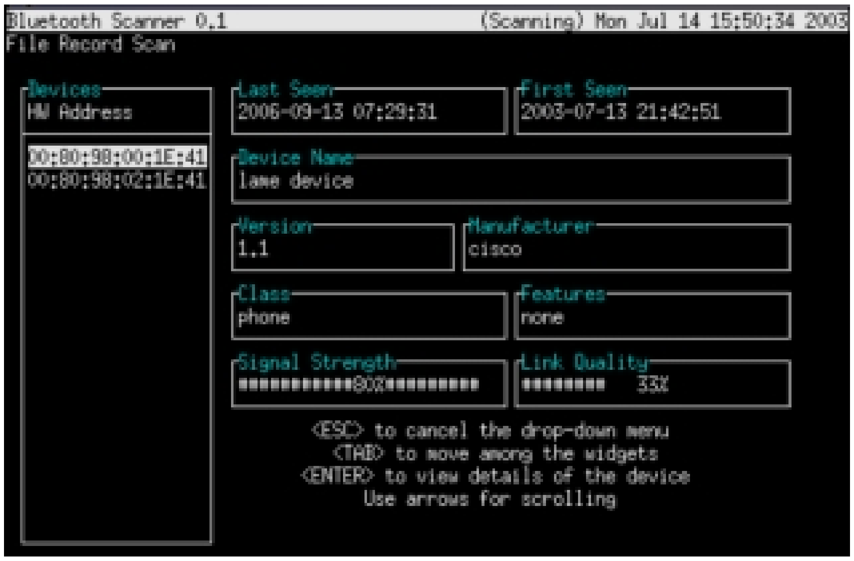
Exhibit F: Non-Discoverable Device Responding to Query


Linux [Redhat 9] - [Ctrl-Alt-F1] - VMware GSX Server
File Power Settings Devices View Help
Power Off
Power On
Suspend
Reset
Full Screen

```
fang.c: In function 'cmd_hunt':
fang.c:78: warning: unused variable 'opt'
/usr/include/bluetooth/hci_lib.h: At top level:
fang.c:56: warning: 'hunt_options' defined but not used
/bin/sh ../libtool --mode=link gcc  -Wall -g -O2  -o fang  fang.o  -L/usr/lib -l
bluetooth
gcc -Wall -g -O2 -o fang fang.o  -L/usr/lib -lbluetooth
[root@localhost redfang]# hcitool  -i hci0 scan
Scanning ...
        00:60:57:71:BF:6C       Breeze
[root@localhost redfang]# ./fang -i hci0 hunt 10000
redfang - the bluetooth hunter ver 1.00.alpha
(c)2003 @stake Inc
author: Ollie Whitehouse (ollie@atstake.com)
timeout: 10000
starting...
00:80:98:34:C2:7C

.start--->
address :- 7C:C2:34:98:80:00
name    :- hostpc
<.end-----
00:80:98:34:C2:7D

[root@localhost redfang]# _
```

This led the author to develop @stake RedFang v0.1 [10] which (currently) brute forces the Cambridge Silicon Radio [11] assigned BADDR address range (which is the same as a MAC).TDK uses this range for Bluetooth devices, but it has been extended through the work of the Shmoo Research Group [12], an example of which is contained in *Exhibit G*.

Exhibit G: Bluesniff



```
Bluetooth Scanner 0.1                  (Scanning) Mon Jul 14 15:50:34 2003
File Record Scan

Devices              Last Seen              First Seen
HW Address           2006-09-13 07:29:31    2003-07-13 21:42:51

00:80:98:00:1E:41    Device Name
00:80:98:02:1E:41    lame device

                     Version          Manufacturer
                     1.1              cisco

                     Class                  Features
                     phone                  none

                     Signal Strength        Link Quality
                     ###########80%#########  ########   33%

                          <ESC> to cancel the drop-down menu
                            <TAB> to move among the widgets
                         <ENTER> to view details of the device
                              Use arrows for scrolling
```

By this point you will have located all the Bluetooth devices you can contact (without the use of a Bluetooth sniffer). You will not have been able to locate devices marked as non-contactable, but since these can't be contacted, they shouldn't be of any interest to an assessor. At this point you will be able to ascertain the following:

- The Bluetooth address (i.e. MAC)
- Class of device
- Type of device
- Device name

It Can Take a Long Time

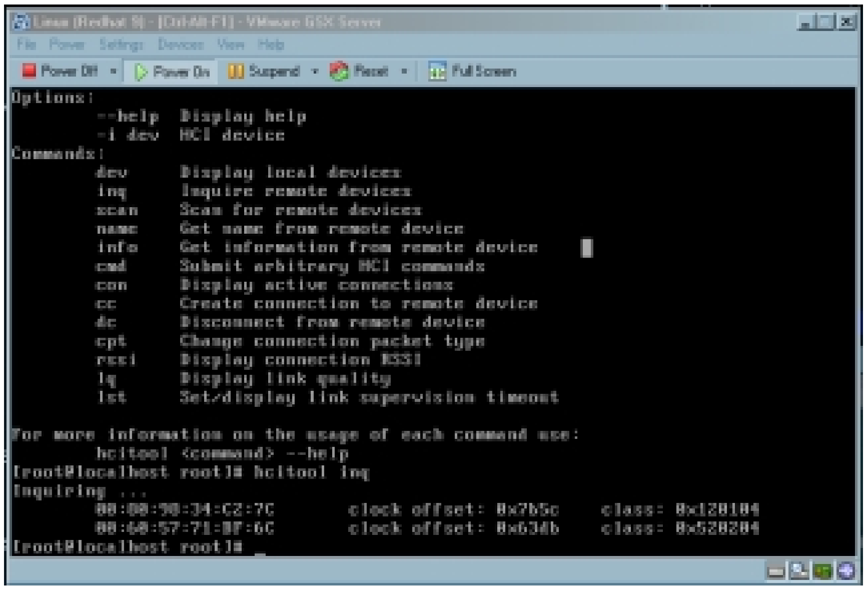
Some vendors [19] have criticized the amount of time such as attack could take. However, it is worth noting that there can be significant performance gains that can be demonstrated. For instance, a multi-threaded version of RedFang could simultaneously utilize up to 8 USB Bluetooth devices that would reduce the 11hrs claimed by TDK to approximately 90 minutes (based on one vendor's range). In addition the Shmoo Research Group has done some seeding within RedFang (i.e. only attacking assigned ranges). This again improves performance and significantly reduces scan time; hence the forthcoming Bluetooth 1.2 specification has apparently addressed this vulnerability.

Extracting Information from the Device You Have Located

In order to obtain the information mentioned in the pervious section, you need to perform a number of different steps. As we work through these steps we will also be working our way up the protocol stack, starting with the HCI (Link Level) through to SDP (Application) in order to obtain information on the host concerned.

The first step is to obtain the device class; this will typically indicate the range and capabilities of the target device. Once you have obtained this information you can use the reference located on the Bluetooth site [16] to understand the device's capabilities. In order to do this we use '`hcitool`' from the Bluez implementation within Linux as shown in *Exhibit H*.

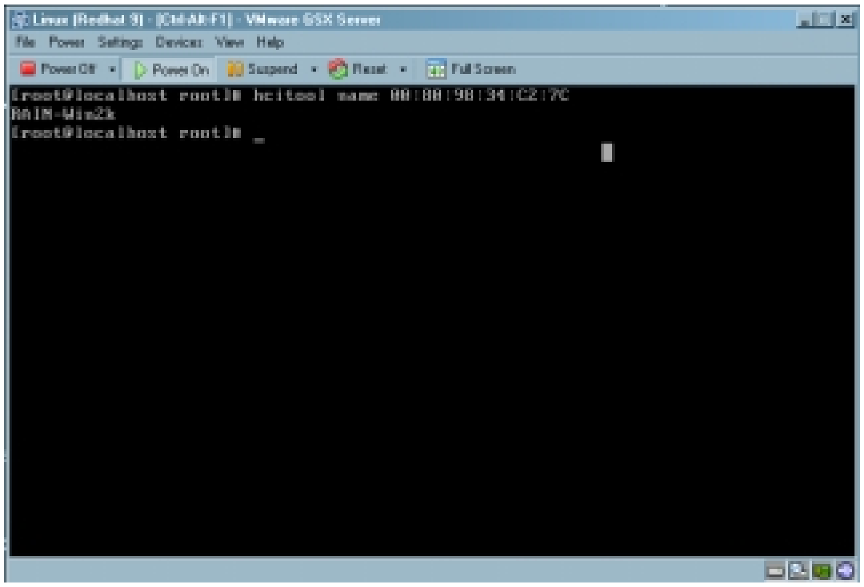
Exhibit H: Device class and clock offset extraction


Linux (Redhat 9) - [Ctrl-Alt-F1] - VMware GSX Server
File Power Settings Devices View Help
Power Off
Power On
Suspend
Reset
Full Screen

```
Options:
        --help  Display help
        -i dev  HCI device
Commands:
        dev     Display local devices
        inq     Inquire remote devices
        scan    Scan for remote devices
        name    Get name from remote device
        info    Get information from remote device
        cmd     Submit arbitrary HCI commands
        con     Display active connections
        cc      Create connection to remote device
        dc      Disconnect from remote device
        cpt     Change connection packet type
        rssi    Display connection RSSI
        lq      Display link quality
        lst     Set/display link supervision timeout

For more information on the usage of each command use:
        hcitool <command> --help
[root@localhost root]# hcitool inq
Inquiring ...
        00:00:90:34:C2:7C       clock offset: 0x7b5c    class: 0x120104
        00:60:57:71:BF:6C       clock offset: 0x63db    class: 0x520204
[root@localhost root]# _
```

The second command is one of verification. For example, if the host you are searching for does not appear in the output from your inquiry you may wish to confirm that the device exists. Also it is useful to verify that it is the correct device you're assessing. In *Exhibit I* we do a manual version of what RedFang does with '`hcitool`'.

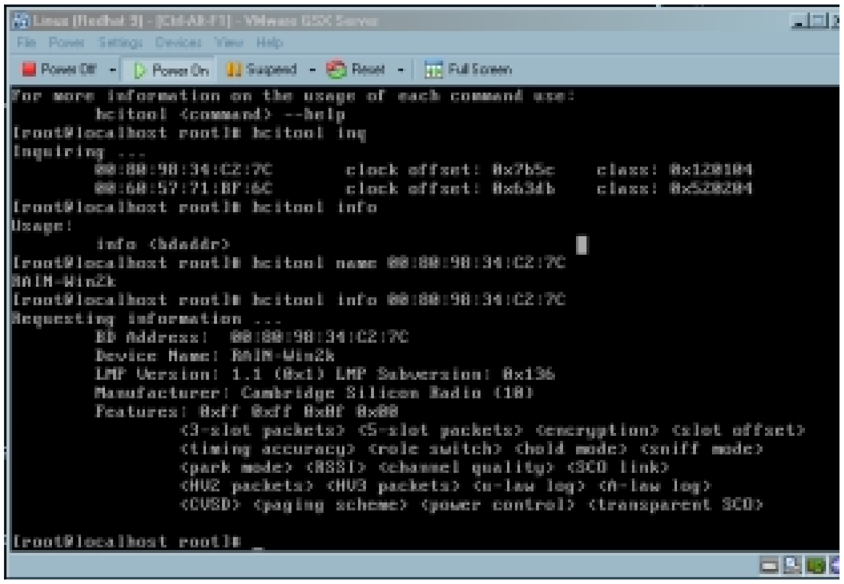
Exhibit I: Name extraction with hcitool


Linux (Redhat 9) - [Ctrl-Alt-F1] - VMware GSX Server
File Power Settings Devices View Help
Power Off
Power On
Suspend
Reset
Full Screen

```
[root@localhost root]# hcitool name 00:00:90:34:C2:7C
RAIN-Win2k
[root@localhost root]# _
```

The final stage for the HCI layer will be the ‘`info`’ parameter to ‘`hcitool`., This will reveal all the information that it is possible to obtain from this part of the Bluetooth protocol. *Exhibit J* shows the output of the command when run against a Windows 2000 based IBM T30 laptop.

Exhibit J: Device class and clock offset extraction


Linux [Redhat 9] - [Ctrl-Alt-F1] - VMware GSX Server
File Power Settings Devices View Help
Power Off
Power On
Suspend
Reset
Full Screen

```
For more information on the usage of each command use:
        hcitool <command> --help
[root@localhost root]# hcitool inq
Inquiring ...
        00:80:98:34:C2:7C       clock offset: 0x7b5c    class: 0x120104
        00:60:57:71:BF:6C       clock offset: 0xb3db    class: 0x520204
[root@localhost root]# hcitool info
Usage:
        info <bdaddr>
[root@localhost root]# hcitool name 00:80:98:34:C2:7C
RAIN-Win2k
[root@localhost root]# hcitool info 00:80:98:34:C2:7C
Requesting information ...
        BD Address:  00:80:98:34:C2:7C
        Device Name: RAIN-Win2k
        LMP Version: 1.1 (0x1) LMP Subversion: 0x136
        Manufacturer: Cambridge Silicon Radio (10)
        Features: 0xff 0xff 0x0f 0x00
                <3-slot packets> <5-slot packets> <encryption> <slot offset>
                <timing accuracy> <role switch> <hold mode> <sniff mode>
                <park mode> <RSSI> <channel quality> <SCO link>
                <HV2 packets> <HV3 packets> <u-law log> <A-law log>
                <CVSD> <paging scheme> <power control> <transparent SCO>

[root@localhost root]# _
```

Next we need to move to the SDP (Service Discovery Protocol) part of the assessment. This is where we will discover what applications/interfaces (services) will be offered over Bluetooth. Common interfaces the author has seen are:

- Serial Port
- FTP
- Audio Headset
- Network Interface
- Dial-Up Networking
- Inbox
- Business Cards
- Fax

There will be a wide range of outputs. The command you will use to perform this part of the assessment is ‘`sdptool`’ which ships as a different package of the Bluez distribution. This will allow you to perform a number of actions against both the local sdp daemon and the remote sdp daemon/implementation:

- Search for a service
- Browse all available services
- Add local service
- Delete local service
- Get local service
- Set/Add attribute to a SDP record
- Set/Add attribute sequence to a SDP record

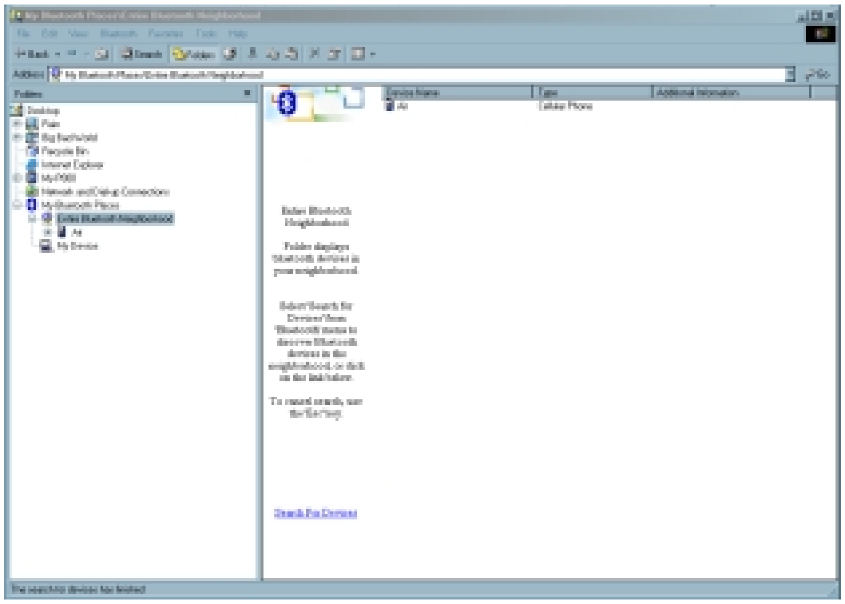
Contained within *Exhibit K* is the output observed when run against our favorite Windows host. As we can see, some interesting services appear to the casual observer (so long as additional security has not been implemented).

Exhibit K: Enumeration of Windows 2000 Host from Linux

```
[root@pc65 root]# sdptool browse 00:80:98:34:C2:7C
Browsing 00:80:98:34:C2:7C ...
Service Name: Bluetooth Serial Port
Service RecHandle: 0x10000
Service Class ID List:
  "Serial Port" (0x1101)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 1
Language Base Attr List:
  code_ISO639: 0x656e
  encoding:    0x6a
  base_offset: 0x100
Profile Descriptor List:
  "Serial Port" (0x1101)
    Version: 0x0100

Service Name: Dial-Up Networking
Service RecHandle: 0x10001
Service Class ID List:
  "Dialup Networking" (0x1103)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 3
Language Base Attr List:
  code_ISO639: 0x656e
  encoding:    0x6a
  base_offset: 0x100
Profile Descriptor List:
  "Dialup Networking" (0x1103)
    Version: 0x0100

Service Name: Information Exchange
Service RecHandle: 0x10002
Service Class ID List:
  "OBEX Object Push" (0x1105)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 4
  "OBEX" (0x0008)
Language Base Attr List:
  code_ISO639: 0x656e
  encoding:    0x6a
  base_offset: 0x100
Profile Descriptor List:
  "OBEX Object Push" (0x1105)
    Version: 0x0100

Service Name: File Transfer
Service RecHandle: 0x10003
Service Class ID List:
  "OBEX File Transfer" (0x1106)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 5
  "OBEX" (0x0008)
Language Base Attr List:
  code_ISO639: 0x656e
  encoding:    0x6a
  base_offset: 0x100
Profile Descriptor List:
  "OBEX File Transfer" (0x1106)
    Version: 0x0100

Service Name: Fax
Service RecHandle: 0x10004
Service Class ID List:
  "Fax" (0x1111)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 6
Language Base Attr List:
  code_ISO639: 0x656e
  encoding:    0x6a
  base_offset: 0x100
Profile Descriptor List:
  "Fax" (0x1111)
    Version: 0x0100
```

If we switch back to Windows 2000, *Exhibit L* contains the example you would see when looking for discoverable Bluetooth devices.

Exhibit L: Windows 2000 Discoverable Devices



Entire Bluetooth Neighborhood

Folder displays Bluetooth devices in your neighborhood.

Select 'Search for Devices' from 'Bluetooth' menu to discover Bluetooth devices in the neighborhood, or click on the link below.

To cancel search, use the 'Esc' key.

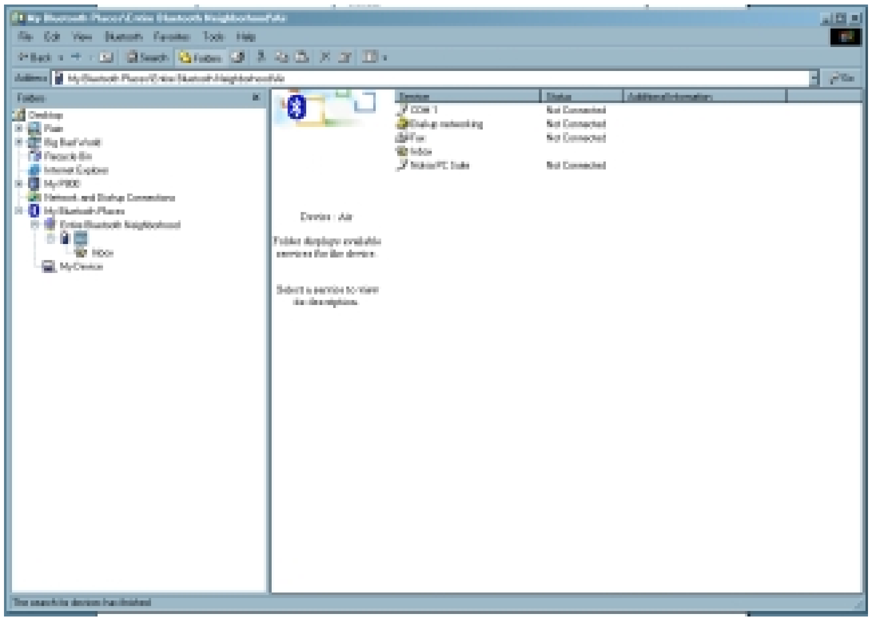
Search For Devices

Device Name	Type	Additional Information
Air	Cellular Phone	

The search for devices has finished

Exhibit M and N below shows Window's exploration of a Nokia 6310i cellular telephone without any authentication being performed.

Exhibit M: Enumeration of Nokia 6310i Anonymously (via Windows)



Device : Air

Folder displays available services for the device.

Select a service to view its description.

Device	Status	Additional Information
COM 1	Not Connected	
Dialup networking	Not Connected	
Fax	Not Connected	
Inbox		
Nokia PC Suite	Not Connected	

The search for devices has finished

Exhibit N: Enumeration of Nokia 6310i Anonymously (via Linux)

```
[root@pc65 root]# sdptool browse 00:60:57:71:BF:6C
Browsing 00:60:57:71:BF:6C ...
Service Name: Fax
Service RecHandle: 0x10000
Service Class ID List:
  "Fax" (0x1111)
  "Generic Telephony" (0x1204)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 2
Language Base Attr List:
  code_ISO639: 0x656e
  encoding:    0x6a
  base_offset: 0x100
Profile Descriptor List:
  "Fax" (0x1111)
    Version: 0x0100

Service Name: OBEX Object Push
Service RecHandle: 0x10001
Service Class ID List:
  "OBEX Object Push" (0x1105)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 9
  "OBEX" (0x0008)
Language Base Attr List:
  code_ISO639: 0x656e
  encoding:    0x6a
  base_offset: 0x100
Profile Descriptor List:
  "OBEX Object Push" (0x1105)
    Version: 0x0100

Service Name: Dial-up networking
Service RecHandle: 0x10002
Service Class ID List:
  "Dialup Networking" (0x1103)
  "Generic Networking" (0x1201)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 1
Language Base Attr List:
  code_ISO639: 0x656e
  encoding:    0x6a
  base_offset: 0x100
Profile Descriptor List:
  "Dialup Networking" (0x1103)
```

```
    Version: 0x0100

Service Name: Nokia PC Suite
Service RecHandle: 0x10003
Service Class ID List:
  "Serial Port" (0x1101)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 15
Language Base Attr List:
  code_ISO639: 0x656e
  encoding:    0x6a
  base_offset: 0x100

Service Name: COM 1
Service RecHandle: 0x10004
Service Class ID List:
  "Serial Port" (0x1101)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 3
Language Base Attr List:
  code_ISO639: 0x656e
  encoding:    0x6a
  base_offset: 0x100

Service Name: Voice Gateway
Service RecHandle: 0x10005
Service Class ID List:
  "" (0x111f)
  "Generic Audio" (0x1203)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 13
Language Base Attr List:
  code_ISO639: 0x656e
  encoding:    0x6a
  base_offset: 0x100
Profile Descriptor List:
  "" (0x111e)
    Version: 0x0100

Service Name: Audio Gateway
Service RecHandle: 0x10006
Service Class ID List:
  "Headset Audio Gateway" (0x1112)
  "Generic Audio" (0x1203)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 12
Language Base Attr List:
  code_ISO639: 0x656e
  encoding:    0x6a
  base_offset: 0x100
Profile Descriptor List:
  "Headset" (0x1108)
    Version: 0x0100
```

If the Nokia 6310i device had Link Layer (Mode 3) security in place then the attacker would have been stopped at this point. The subject of Link Layer security issues will be dealt with in the future, however for now the following should be kept in mind:

- Link layer security is usually Bluetooth address based
- Link layer security usually involves a PIN (normally between 4 and 16 digits)

The first can be easily bypassed by someone with the correct equipment and knowledge. The second may be susceptible to a number of attacks, however these will not be dealt with here. Now you are faced with two paths:

- Device does not have any security authentication (you're in)
- Device has service level authentication (the biggest challenge)

Without attacking service level authentication, you may be surprised what you can do anonymously or semi-anonymously. For example, you could push a business card or image onto the device without requiring the user to explicitly accept it (similar to e-mail). Alternatively you may be able to retrieve certain information such as a business card or out box from the device anonymously.

One of the more interesting things to observe is the service discovery process using '`hcidump`' from the Bluez implementation. This is a great tool for getting as low as we can within the protocol stack with today's mass produced hardware. This can be an invaluable diagnostic, IDS, and research tool for someone embarking on Bluetooth research.

This concludes the introduction to War Nibbling. How to connect to a device's specific services is left as an exercise for the reader, however there are some application-specific security features that may be encountered. While these have been discussed briefly in this paper, there also exists a wealth of documentation on the official Bluetooth site on the types of application-specific credentials you will need to be aware of.

Defensive Techniques - Hiding Your Devices

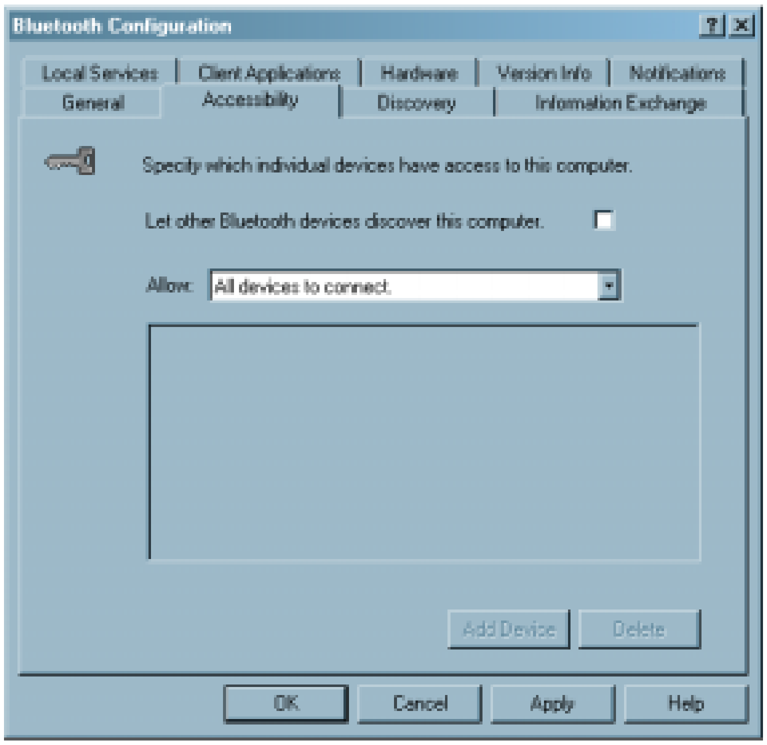
We have dealt with aggressive techniques used to perform assessments of Bluetooth deployments, but we have yet to visit the subject of defense. One of the first obstacles we had to overcome was the discovery of non-discoverable devices (it was surprising to see the number of devices that don't by default implement this security measure).

In *Exhibit O* we show how to make a Windows 2000 host with Bluetooth non-discoverable. This is possible in equipment from the following vendors:

- Red-M
- Nokia
- SonyEricsson
- Ericsson
- TDK / Cambridge Silicon Radio
- Bluez

However, the author would expect this to be present in all devices due to its inclusion within the specification. Under Linux with Bluez for example the '`hcid`' daemon can also be configured to prevent discovery. There is the added benefit in Linux of not having to start up this daemon to perform certain client operations, so that in other situations you can run silent. This has the added benefit for deployments such as Bluetooth monitoring solutions.

Exhibit O: How to Disable Discovery of a Windows 2000 Host



Specify which individual devices have access to this computer.

Let other Bluetooth devices discover this computer. ☐

Allow: All devices to connect.

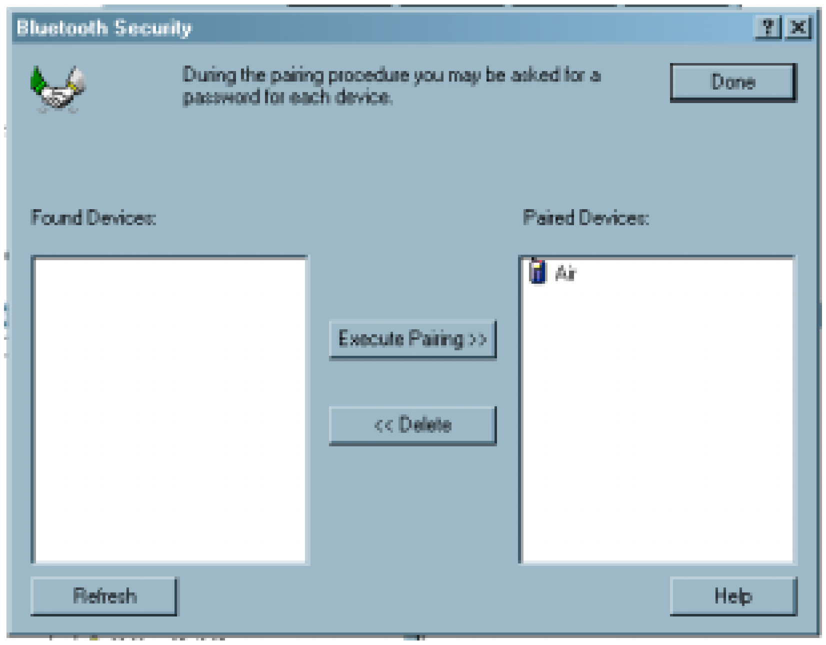
Defensive Techniques - Bonding Information Checks

One the most obvious parts of the Bluetooth configuration is the bonding and/or security configuration. Basically, this details devices that have had:

- User interaction to agree the bond should occur
- Authentication via mutual PINs

It is good practice to check this information regularly. Refer to the section below titled '*The Implementation Issue of Cellular Handset OEMs Overlooked*' as a good example of why you would want to do this. Contained within *Exhibit P* is an example of what we would observe on a Windows host. This shows that currently there is one (1) Bluetooth device in my locality and in addition that the host is paired/bonded with it. This means that the host can communicate with it and it can communicate with the host. This obviously does not include any application layer security (i.e. within a PPP connection there is a requirement for CHAP or PAP authentication). This can be easily performed on cellular equipment in a similar way.

Exhibit P: Bonding/Pairing Information on a Windows Host



During the pairing procedure you may be asked for a password for each device.

Defensive Techniques – Windows Registry, a Wealth of Information

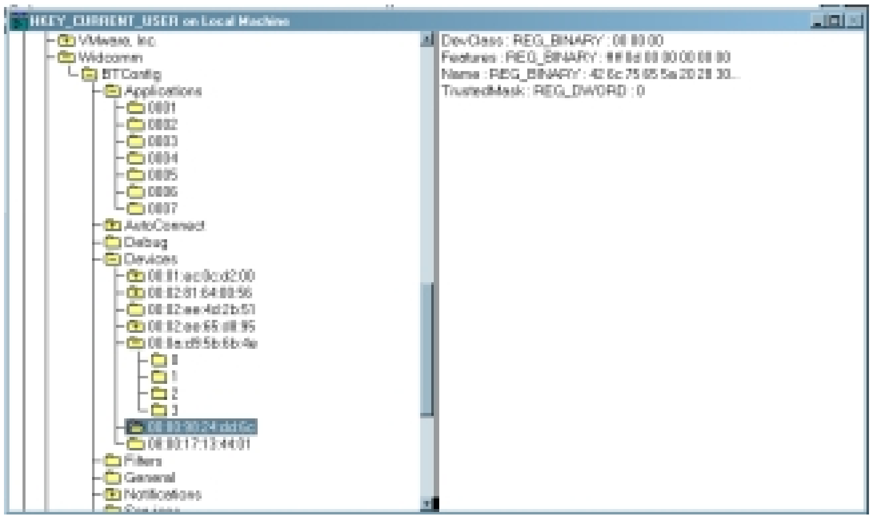
This is a rarely visited area that contains some of the most valuable information for those who perform incident investigations (or for those who take a proactive approach to security and look at such information sources on a regular basis). For example, recorded within the Windows registry are the following (see *Exhibit Q* below):

- List of all devices ever paired with the host
- Class of devices observed
- Services available on devices observed
- Auto connection settings

Security conscious users may wish to review what information is present and clear up on a regular basis.

If you believe that an intruder might have placed a temporary bond or pairing upon your host, you can quickly ascertain if the attack was sloppy. Again this is only beneficial if this area of the registry is monitored on a frequent basis with some baseline level to work from.

Exhibit Q: Bluetooth Information Contained Within the Registry



```
HKEY_CURRENT_USER on Local Machine

  VMware, Inc.
  Widcomm
   BTConfig
    Applications
     0001
     0002
     0003
     0004
     0005
     0006
     0007
    AutoConnect
    Debug
    Devices
     00:01:ec:0c:d2:00
     00:02:01:64:00:56
     00:02:ee:4d:2b:51
     00:02:ee:65:d8:95
     00:0a:d9:5b:6b:4e
      0
      1
      2
      3
     00:00:30:24:dd:6c
     00:00:17:13:44:01
    Filters
    General
    Notifications

DevClass : REG_BINARY : 00 00 00
Features : REG_BINARY : ff ff 0d 00 00 00 00 00
Name : REG_BINARY : 42 6c 75 65 5a 20 28 30...
TrustedMask : REG_DWORD : 0
```

Future Defensive Techniques - Personal Firewalls

This may be one of future security solutions for wireless embedded devices. For example, companies such as BlueFire Security [17] are developing products for this market. Currently the author is not aware of any which effectively protect the baseband protocol within Bluetooth, or any that provide any protection to at least HCI and SDP parts of the protocol stack. These solutions will also depend on the type of device that has Bluetooth deployed, as this will dictate the extra processing and memory that will be available for dealing with such overhead.

@stake Research - Antennas and Bluetooth

If there is a misconception that antennas cannot be added to Bluetooth devices then let us get rid of it now. Although possible, the author has yet to perform any in-depth analysis or research in this area, or on how much range can be gained from the application of, say, a directional antenna. There is no technical reason that gains should not be possible. One reason this may not be common practice in the US is Federal Communications Commission (FCC) regulations.

@stake Research - The Implementation Issue of Cellular Handset OEMs Overlooked

One of the security features within Bluetooth is the concept of bonding and/or pairing. This requires each device to authenticate. This is normally achieved by the user acceptance of the device pairing, and that both devices mutually supply the same numeric PIN.

Once a device is paired, usually there is no need for future authentication. The specifications do provide for this, but in the name of usability security has often been seen as an obstacle.

@stake found that a number of cellular handset manufactures do not erase this pairing information when SIM cards are swapped. This potentially exposes users of rental, recycled or temporary handsets to the risk that an attacker has previously placed a bond upon the device .

@stake performed the following procedures for each device tested. The purpose was to demonstrate that Bluetooth pairs persist across different subscriber SIMs.

PHONE

1. Erase all pairing information
2. Remove Battery
3. Remove User SIM(1) from phone
4. Place Attacker SIM(2) into phone
5. Start Phone
6. Enter Attacker's SIM(2) PIN
7. Configure bonding with PC
8. Save settings
9. Turn off phone
10. Remove Attacker SIM(2)
11. Place User SIM(1) in phone
12. Start Phone

13. Enter PIN SIM(1) PIN

WINDOWS 2000 LAPTOP

1. With User SIM(1) Attempt to establish a call via Bluetooth

As a result, @stake believes that the following avenues of attacks could exploit this vulnerability:

- Rental phones (airports)
- Loan phones (service centers)
- Recycled phones (third/developing world)

In addition the following are also potentially viable:

- Malicious distributor (new phones)
- Malicious insider (new/reissued phones)
- Lunchtime attack (physically unsecured phones)

@stake Research - Sweet-Tooth (The Bluetooth Honeypot)

One of the concepts the author has been working on is Honeypots [15] for Bluetooth. The initial implementation is based on the Bluez daemon code (`hcid` and `sdpd`) and is known as 'Sweet-tooth' This currently allows you to do the following:

- Make the device appear as anything (desktop pc, mobile phone, laptop)
- Make the device list any service you want (audio device, ftp)

The plan here to is to create a fully- working honeypot with full implementations to allow you to trap the attacker and obtain as much information about the attacking device as possible. One of the features currently being considered is to connect back to the attacking device and try and retrieve the vCard and thus obtain the attacker's contact information.

@stake Research – The Future

The author sees the following areas as being key to the future of Bluetooth security and would benefit from more in-depth research in relation to assessment and defense techniques:

- L2CAP
- SDP
- OBEX
- UPnP

- BNEP
- Over the air encryption enforcement and negotiation

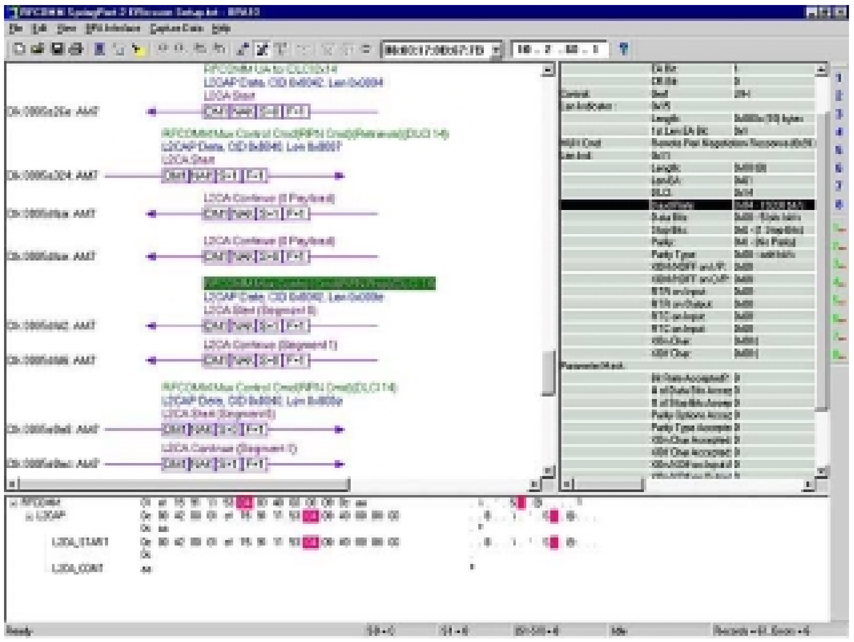
Future security research of the above should not only examine the standards perspective, but should also examine specific implementations that can yield additional vulnerabilities. @stake's RedFang 0.2 currently has the following features under development:

- Multithreading to speed up non-discoverable device scans
- ing the L2CAP layer
- Attacking the challenge/responses systems within Bluetooth
- Over the air encryption attacks

In addition, the Bluetooth Special Interest Group (BSIG) unveiled the 1.2 specification which contains additional enhancements to the security of Bluetooth. The author did attempt to locate a copy of the specification so it could be reviewed, however it is not published for general consumption as of yet [20]. The main addition appears to be 'Anonymity Mode' "to increase the security of Bluetooth links by masking the physical address of radios to prevent identity attacks and snooping."

If anyone is interested in learning more about the air interface (base band) specifications and vulnerabilities, I would encourage you to read up on the Bluetooth Protocol Analyzer "BPA-D10/D11/D12" from Computer Bus Tools (http://www.bustools.com/bluetooth/bpa.htm). See *Exhibit* R below.

Exhibit R: Bluetooth Protocol Analyzer



Summary

This paper is an introduction on how to pragmatically assess your environment for Bluetooth deployments using the latest vulnerabilities and attacks vectors. It also is an introduction to the Bluetooth protocol implementation and design in relation to security that need further research from the industry.

In summary, there are very real risks for Bluetooth enabled devices. These risks will proliferate as adoption becomes more widespread and the devices vary from their default configurations. We have also demonstrated the power of RedFang when trying to locate devices to assess [19].; Unlike 802.11 Blutooth devices are harder to locate due to their low power, and potentially contain very sensitive information and/or interfaces that could be exposed to rogue third parties without the obstacles of hunting through corporate networks.

With the mass arrival of Class 1 devices, we now have the same potential security crisis as with 802.11. This may even be worse due to the proliferation of the types of devices that are Bluetooth enabled, including everything from headphones to laptop computers. Vendors should understand these issues and risks and develop an effective mechanism of delivering the device as secure out of the box. This will not only enhance the default security posture of Bluetooth, but will also educate users as to the need to ensure security for these devices. While things are improving with the introduction of an 'Anonymity Mode' in the forthcoming Blootooth 1.2 specifications, , the real outcome will largely depend on both the design and the implementation of the final specifications and how quickly they get adopted. We already have millions of 1.0 and 1.1 based devices in the hands of users today.

Thanks

As with all white papers there is a team of people behind the scenes without whom it would never have been possible:

- Jesse Burns (@stake, TITLE)
- Colin Gillingham (@stake, TITLE)
- Graham Murphy (@stake, TITLE)
- Paul Ashton (@stake, TITLE)

I would also like to thank my peers for taking time to review this paper and for providing valuable feedback:

- Emmanuel Gadaix, Frank Rieger, Al Hood and Rob Mann

Related @stake Research

The following security advisories are some of the additional work resulting from @stake's research on the Bluetooth protocol and its specific implementations.

Advisories:

- Multiple Red-M 1050 Blue Tooth Access Point Vulnerabilities (http://www.atstake.com/research/advisories/2002/a060502-1.txt)
- Ericsson T39M Bluetooth Denial of Service (to be published)
- Multi Cellular Handset Vendor Bluetooth Vulnerability (to be published)

Acronyms and Terms

Protocol Layers

- Bluetooth Radio
- Bluetooth Base band
- Link Manager Protocol (LMP)
- Host Controller Interface (HCI) - Controller of Base band
- Logical Link Control and Adaptation Protocol (L2CAP)

Profiles

- GAP Profile: The Generic Access Profile
- DNP or DUN (Dial Up)
- SPP (Serial Port) (RFCOMM): LAN access profile
- Pan (BNEP): Personal area network
- SDAP (SDP): The Service Discovery Application Profile
- CTP: The Cordless Telephony Profile
- IP: Intercom profile
- HS: Headset profile
- FP: Fax profile
- GOEP: The Generic Object Exchange Profile (OBEX)
- OPP: Object push profile

- FTP: File transfer profile
- SP: Synchronisation profile (PIM-data)

Bibliography

[1] Bluetooth Office Homepage
https://www.bluetooth.org/

[2] War Driving Introduction
http://www.wardriving.com/

[3] Bluetooth Specification
https://www.bluetooth.org/docman2/ViewCategory.php?group_id=53& category_id=213

[4] Bluetooth Specifications
https://www.bluetooth.org/foundry/specification/docman/

[5] Linux
http://www.kernel.org/

[6] Official Linux Bluetooth Implementation
http://www.bluez.org/

[7] VMWare
http://www.vmware.com/

[8] IBM – T30 OEM
http://www.ibm.com/

[9] TDK USB Bluetooth Adapter
http://www.tdksystems.com/products/bluetoothchoice.asp?id=1

[10] RedFang
http://www.atstake.com/

[11] Cambridge Silicon Radio
http://www.csr.com/home.htm

[12] Bluesniff is Proof of Concept Code for a Bluetooth War Driving Utility
http://bluesniff.shmoo.com/

[13] Supplier of Single Board Computers
http://www.embedded586.com/

[15] Defacto Home of Honeypot Information
http://www.honeynet.org/

[16] Bluetooth Capabilities Breakdown
https://www.bluetooth.org/foundry/assignnumb/document/assignnumb_reference and
https://www.bluetooth.org/foundry/assignnumb/document/service_discovery

[17] Bluefire Security
http://www.bluefiresecurity.com/

[18] Reference on L2CAP, SDP, RFCOMM and Profiles
http://eie555.eie.polyu.edu.hk/tut/tutorial/l2cap.html
http://eie555.eie.polyu.edu.hk/tut/tutorial/rfcomm.html
http://eie555.eie.polyu.edu.hk/tut/tutorial/sdp.html
http://eie555.eie.polyu.edu.hk/tut/tutorial/profiles.html

[19] News - Red Fang "Bluetooth hack" not much use" - TDK
http://www.newswireless.net/articles/0300910-bluestake.html

[12] Communication Consortiums Craft New Conventions
http://www.reed - electronics.com/ednmag/index.asp?layout=article&articleid=CA309625&rid=0&rme=0&cfd=1

About @stake, Inc.

@stake, Inc., the premier digital security consulting firm, provides security services and award-winning products to assess and manage risk in complex enterprise environments. The company's SmartRisk services cover key aspects of security, including applications, critical infrastructure, wireless and wired networks, storage systems, education, and incident readiness. @stake consultants combine technical expertise with a business focus to create comprehensive security solutions to mitigate risks and maximize results. @stake clients include six of the world's tops ten financial institutions, four of the world's top ten independent software companies and seven of the world's top ten telecommunications carriers.

As the first company to develop an empirical model that measures Return On Security Investment (ROSI), @stake keeps security investments in line with business requirements. Headquartered in Cambridge, MA, @stake has offices in London, Chicago, New York, Raleigh, San Francisco, and Seattle. For more information, go to www.atstake.com.