RUXCON 2004

Ollie Whitehouse

July, 2004

@stake

Where Security & Business Intersect℠

Agenda

- Overview of Bluetooth security
- Brief summary of attacks to date
- Introduction to new attacks
- Hunting Bluetooth
- Tool: RedFang 3.0
- Tool: RedSnarf 1.0
- Tool: BlueFang 1.0
- Miscellaneous product research
- Continued research
- Questions

Overview of Bluetooth security

- Bluetooth Security 101
- How it works (Mode 3)
- Some interesting quotes from the specifications

Overview: Bluetooth security 101

- **Mode 1: No Security**
  - Implemented in some cases on Bluetooth access points running PAN profiles
- **Mode 2: Application/Service based**
  - Example: PPP authentication used on certain Bluetooth access points
- **Mode 3: Link-layer (PIN authentication / MAC address security / encryption)**
  - This is the bonding process as well as Bluetooth encryption as specified by the Bluetooth SIG

Overview: How it works (Mode 3)

- **Link keys are the keys to Kingdom, many different types**
  - Kinit – Initialization key
    - Used to protect the transfer of initialization parameters
  - Ka – Unit key
    - Generated in a single device: “The unit key shall be generated once at installation of device”
  - Kab – Combination key
    - Combination of two Ka’s and some algorithm magic
  - Kmaster – Temporary key
    - A temporary replacement for the current link key. Can be used to communicate with more than one device with the same encryption key
- **Other Keys**
  - Kc – Encryption key

Overview: How it works (Mode 3)

- **PINS are the pre-shared secret used as a basis for authentication and seeding certain parts of Kinit**
  - According to the specification static PIN's can be used on devices where no user-interface exists (examples here are Bluetooth headsets)
  - If no PIN then '0' is used which is actual fact 0x00 (i.e. NULL)
  - Can be between 1 and 16 bytes (0-9) in length, typically only numeric this results in 10,000,000,000,000,000 (10^16) maximum possible PIN's.
  - If we take the average PIN length of 6 bytes (0-9) we end up with 1,000,000 possible PIN's (10^6)
- **A number of equations are used are different stages (spec defined)**
  - E22 – SAFER+ which produces Kinit / E21 similar but produces Kmaster
  - E0 – SAFER+ which produces K'c
  - E1 – SAFER+ which handles authentication
  - E2 – SAFER+ used for auth key generation
  - E3 – SAFER+ which produces Kc

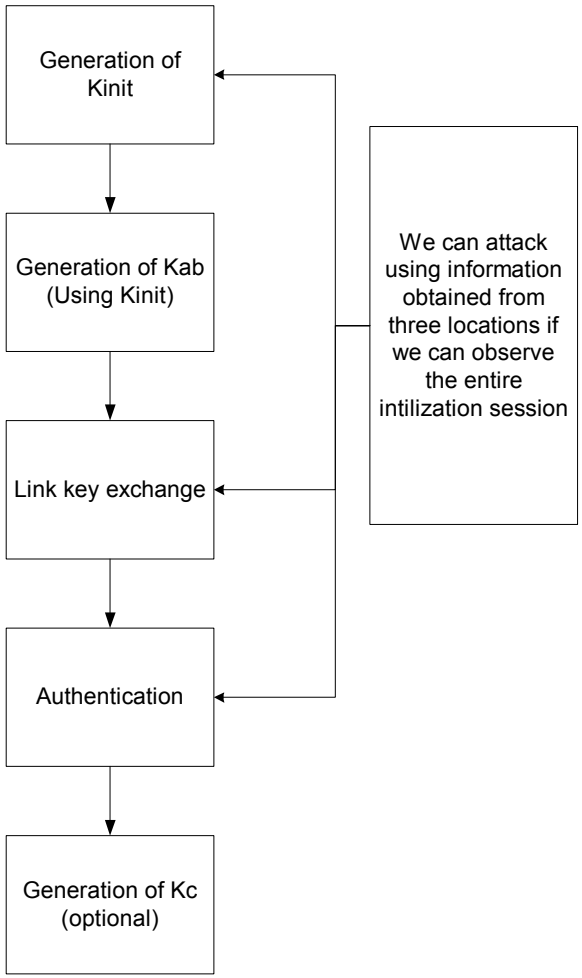
Overview: How it works (Mode 3)

- **Kinit**
  - Generated from:
    - Bluetooth address (48 bits)
    - RAND (128 bits)
    - PIN could be between 8 and 128 bits & PIN Length (8 bits)
- **Ka (Unit Key)**
  - Generated from:
    - Bluetooth address (48 bits)
    - RAND (128 bits)
- **Kab (Combination Key)**
  - Generated from:
    - XOR'ing two Unit Key's

Overview: How it works (Mode 3)

- **Kc**
  - Typically generated from:
    - COF (which is the result of two Bluetooth address (48 bits) XOR'd & ACO which is an additional result of the 'Authentication' to SRES) – 96bits total
    - RANDOM (128bit)
    - Kab (128bit Link-Key)
- **K'c & Kcipher**
  - Generated from:
    - Kc
    - Bluetooth Address of master device (BD_ADDRa)
    - Master clock

Overview: How it works (Mode 3)



Overview: Interesting observations

- **Encryption key size is factory preset**
  - Currently very little, if any, product documentation includes details on encryption key sizes used (could be between 8 to 128 bits)
- **Repeated PIN attacks should incur ever increasing delays between allowed tries**
  - No documentation on when this should be reset

Brief summary of attacks to date

- **Finding non-discoverable devices**
  - RedFang (single then multi threaded)
- **PSM scanning**
  - Port scanning for Bluetooth
- **OBEX attacks - PUSH**
  - Blue Jacking
- **OBEX attacks – PULL**
  - Blue Stumbling

Existing attacks: Non-discoverable devices

- **Idea conceived by *@stake***
  - http://www.atstake.com/research/tools/info_gathering/
- **Principle:**
  - Bluetooth supports the concept of discoverable and non-discoverable
  - Bluetooth (BD_ADDR) addresses are 6 bytes and very similar in nature and purpose to MAC addresses
  - 3 bytes are for vendor assigned ranges, and 3 being specific to the device in that range
  - If we brute force the ranges and do a name inquiry we can discover non-discoverable devices
- **Impact:**
  - Non-discoverable Bluetooth devices can be found

Existing attacks: PSM Scanning

- **Idea conceived by *Collin Mulliner***
  - http://www.betaversion.net/btdsd/
- **Principle:**
  - Works on the idea that not all PSM (Protocol/Service Multiplexer) ports are registered with the local SDP (Service Discovery Protocol)
  - So if we bypass the SDP database and try and connect to PSM's sequentially we may locate hidden functionality
- **Impact:**
  - No PSM's found to-date that offer other than advertised services
  - Idea could be used to create a 'knock' style backdoor for Bluetooth devices

Existing attacks: OBEX - PUSH

- **Idea conceived by *bluejackQ team***
  - http://www.bluejackq.com/
- **Principle:**
  - OBEX allows you to PUSH items anonymously in some cases between devices
  - Can be in a number of formats (i.e. vCards or pictures)
- **Impact:**
  - Annoying, no real security impact
  - Possible extensions to this idea is around sending vCard's with common names such as 'Home' or 'Work' in an attempt to overwrite an existing phone book entry in the recipients cell/smart phone

Existing attacks: OBEX - PULL

- **Idea conceived by *Bruce Potter, Adam Laurie & Ben Laurie***
  - http://www.bluestumbler.org/
  - http://www.shmoo.com/~gdead/dc-11-brucepotter.ppt
- **Principle:**
  - OBEX allows you to PULL items anonymously in some cases between devices
- **Impact:**
  - A number of Nokia, Ericsson & Sony Ericsson handsets are susceptible
  - Very much dependant on vendor's implementation of OBEX/Bluetooth stack
  - Information obtainable can include calendar, real time clock, business card, properties, change log, IMEI
  - CBIT paper showed what you could get up to

Introduction to new attacks

- **Based on analysis of newest 1.2 specification**
  - Older versions may have other vulnerabilities not covered (i.e. 1.0 and 1.1)
- **Purpose**
- **Off-line PIN (via Kinit) recovery**
- **On-line PIN cracking**
- **Off-line encryption key (via Kc) recovery**
- **Impact & Counter measures**

New attacks: Purpose

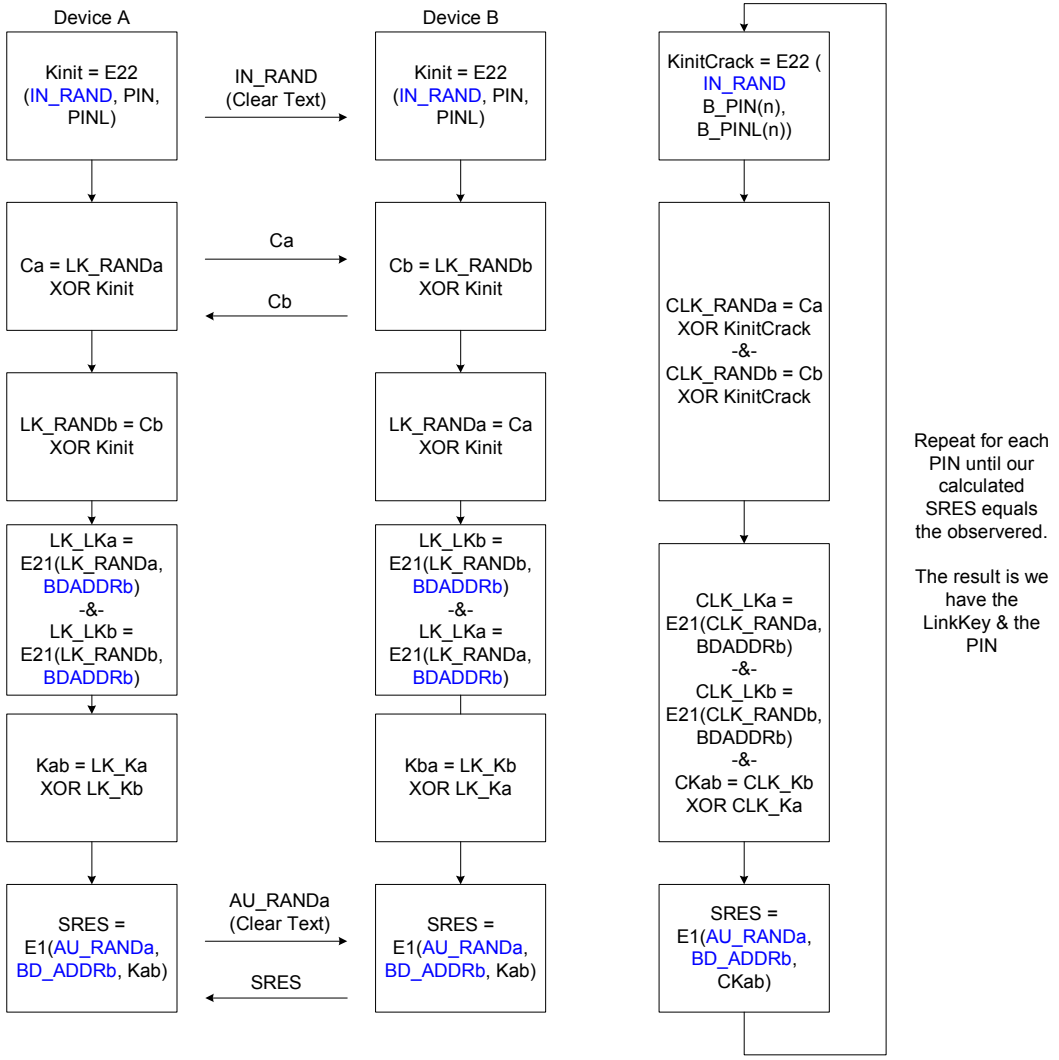
- Recover the PIN used to authenticate the device
- Recover the temporary key (Kinit) used to protect the combination link key (Kab) inputs
- Recover the link key which is used to produce encryption key (Kc)

New attacks: Off-line PIN (via Kinit) recovery

- **We sniff* the initial ‘RAND’ transfer between two devices which occurs in clear-text (effectively the first stage of the bond)**
- **We sniff the XOR’d ‘RAND’(s) used for LinkKey generation**
- **We sniff the AUTH RAND and AUTH SRES which both occur in clear-text (the last stage of the bond)**
- **We do some number crunching and have enough data in order to be able to recover the PIN, LinkKey and all inputs used for both**

** We need to sync the frequency hopping or capture entire 2.4ghz spectrum and do off-line*

New attacks: Off-line PIN (via Kinit) recovery



New attacks: Off-line PIN (via Kinit) recovery

- **So in summary**
  - We need to sniff the bonding process between two devices
    - This can-not be done with off the shelf consumer equipment
  - We need to perform 3 iterations of SAFER+ and 3 XOR's for each crack cycle
  - Assuming we can perform 80,000 crack cycles (3 SAFER+'s and 3 XORs) a second* that gives us 12.5 seconds to recover the key for a 6 digit PIN or 1,446,759 days for a 16 digit PIN on a single CPU

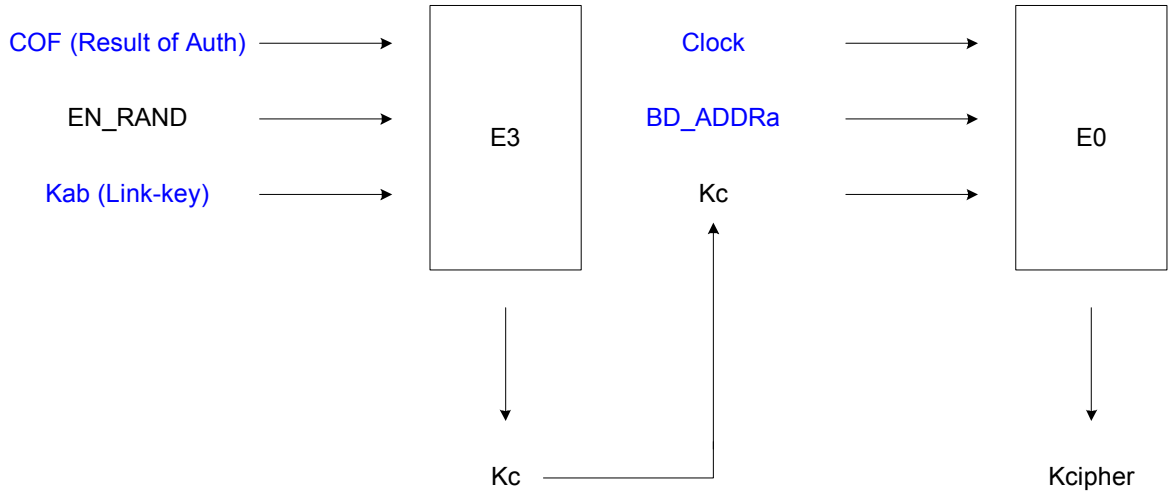
** Using a Pentium III 850mhz machine using libmcrypt*

New attacks: On-line PIN cracking

- **Attack possible is fixed PIN exists in device (i.e. same PIN is used for every connecting device)**
  - We change our Bluetooth address each time and try a PIN
  - Then try the next PIN
  - This will bypass the ever increasing delay between retries counter measure
  - Specifications do not provide solution to this problem

New attacks: Off-line encryption key (via Kc)

- **Extends on from the Kinit recovery attack**
  - Very similar method as we now know 2 of 3 (i.e.master clock and Kc) seeds we need, we simply sniff the EN_RAND in addition



New attacks: Impact & Counter Measures

- **We now have attacks that can**
  - Allow us to obtain the PIN/LinkKey which in turn allows us to abuse existing bonds on a device
  - Have the inputs required to generate Kc (encryption key) and thus decrypt off-line capture sessions
    - Voice phone calls
    - PIM (Personal Information Manager) synchronization
    - Unencrypted network traffic
- **Counter Measures**
  - Perform initial bonding in non-hostile locations (as specifications say)
  - Use long PINs (i.e. 16 bytes)

    Use alpha-numeric if at all possible (i.e. PC to Keyboard/Mouse)
  - Use Diffie Helman for ultra sensitive deployments (specifications cater for this)

Hunting Bluetooth

- Theory
- Shopping list
- Building
- Results

Hunting Bluetooth: Theory

- **802.11 operates within 2.4ghz so does Bluetooth**
  - The @stake theory was: If we use an external 802.11 antenna (i.e. Yagi) it should work
- **Bluetooth device already has a external antenna**
  - The @stake theory was: If we cut that off and replace with pig-tail socket we should have a working rig
- **So if we had one of these devices we could get distance increase**
  - The @stake theory was: We might seen a increase in distance of maybe 5 fold (we didn't do any calculations)

Hunting Bluetooth: Shopping list

- **For this recipe you will need:**
  - 1 broken Lucent Orinoco card (Gold)
  - A Bluetooth device with external antenna (preferably Class 1)
  - Pigtail
  - An external Yagi antenna
  - Some spare wire
  - Minor soldering skills

Hunting Bluetooth: Construction

Connect Lucent plug to outer housing

Solder on to antenna connector on PCB

Side view

Top view

Hunting Bluetooth: Results

14dbi Antenna
SonyEricsson T68i (Class 3 Device)

Result:
Contactable from 150m's away
15 fold increase in range

RedFang 3.0

- **Introduction**
  - First released in June 2003
  - Does a brute force attack again Bluetooth addresses performing a name inquiry to ascertain the existence of a device
  - With input from QinetiQ multi-threaded version released in October 2004
  - Supports up to 127 concurrent USB devices
  - 2.5 also introduced vendor range seeding to speed up the process
- **Changes in 3.0**
  - Supports OBEX – Auto ‘BlueStumbling’ on every found device if possible
  - State storage (i.e. stop and resume a session)
  - Increased number of vendor ranges received from the community

RedSnarf 1.0

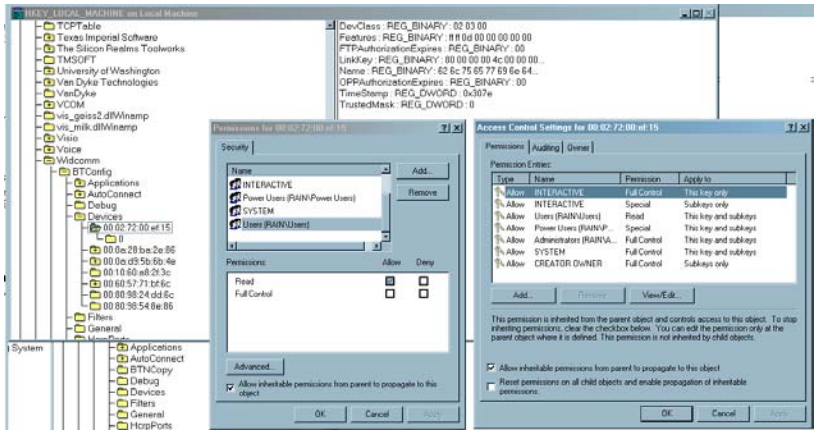
- **Introduction**
  - Our implementation of ‘Bluestumbling’
    - OBEX PULL’ing / Snarf’ing
  - Implements all functionality describe on bluestumbler.org site
- **Useful for**
  - Can be combined with BlueFang for Enterprises looking to enforce corporate policy on Bluetooth enabled devices in certain localities

BlueFang 1.0

- **Introduction**
  - Performs a constant device inquiry
    - Similar to TDK's BlueAlert tool but for Linux
- **Useful for**
  - Enterprises looking to enforce corporate policy on Bluetooth enabled devices in certain localities
    - i.e. Embed device which scans for active Bluetooth devices at entry points and within secure locations

Miscellaneous research

- **WidComm Windows Stack**
  - Link key's are readable by any users on the machine!



- **Belkin Bluetooth Printer Server & Access Point**
  - PAN mode has no authentication, no restrictions

Miscellaneous research

- **Bluetooth Enabled Cell-Phones**
  - If you compromise the device you can use the GSM AT command extensions via the Bluetooth modem to:
    - Obtain all SMS's
    - Contact Information
    - IMEI, Phone number*, Voice mail number*
    - Setup call forwarding*
    - Establish GPRS secondary primary and secondary connections
  - Via OBEX
    - IMEI via device file via OBEX
    - SMS's via OBEX

** Depends on cellular provider*

Continued research

- **What's left**
  - Research into how practical MitM (Man in the Middle) attacks are
    - This is due to mutual authentication being optional
    - Also being able to provide null's for rand(s)
  - Feasibility of building cheap base band hardware and software implementations
    - 3,000 GBP and 10,000 GBP development kits and sniffers respectively is out range for most enterprise security departments
  - Reviewing pseudo random sources used in implementations
  - Reviewing documented attacks again SAFER+ and see if applicable to Bluetooth

Questions

Thanks for listening, @stake's Bluetooth research team:

Stephen Kapp
Software Development &
Peer Review

Graham Murphy
Hardware

Ollie Whitehouse
Software Development &
Research

Appendices

- **Bluetooth security resources**
  - http://www.bluetooth.org/
  - http://www.bluejackq.com/
  - http://www.bluestumbler.org/
  - http://www.shmoo.com/~gdead/dc-11-brucepotter.ppt
  - http://www.niksula.cs.hut.fi/~jiitv/bluesec.html
  - http://www.atstake.com/research/tools/info_gathering/
  - http://www.rsasecurity.com/rsalabs/staff/bios/mjakobsson/bluetooth/bluetooth.pdf
- **Bluetooth security tools**
  - BlueSniff
    - http://bluesniff.shmoo.com/bluesniff-0.1.tar.gz
  - PSMScan
    - http://www.betaversion.net/btdsd/psm_scan.tar.gz
  - BTScanner
    - http://www.pentest.co.uk/cgi-bin/viewcat.cgi?cat=downloads&section=01_bluetooth
  - BlueAlert
    - http://www.tdksystems.com/software/apps/content.asp?id=4

Appendices

- **Hardware Bluetooth sniffers capable of baseband**
  - BPA105
    - http://www.fte.com/
- **Development kits**
  - CSR Casira
    - http://www.csr.com/products/casira.htm
  - CSR BlueSuite
    - http://www.csr.com/products/bluesuite.htm
  - CSR BlueLab
    - http://www.csr.com/products/bluelab.htm