Bluetooth unter Linux einrichten und sicher betreiben

Chemnitzer Linux-Tage, 04./05. März 2006

Wilhelm Dolle, Director Information Technology,
CISSP, CISA, BSI-Grundschutz-Auditor,
interActive Systems GmbH

Agenda

- (Technische) Grundlagen von Bluetooth
- Bluetooth unter Linux einrichten (Beispiel Treo 650 / Fedora Core 5 Test 3)
- Bluetooth sicher betreiben

Was ist Bluetooth?

- Offener Industriestandard (IEEE 802.15.1-2002)
- Lizenzfreies Nahbereichsfunkverfahren zur kabellosen Sprach- und Datenkommunikation zwischen IT-Geräten (Kabelersatz und Ad-hoc-Networking)
- Ziele der „Special Interest Group“ (SIG) – 1998 von Ericsson, Nokia, IBM, Intel und Toshiba gegründet
  - Herstellerunabhängiger Standard
  - Peer-to-Peer Datenkommunikation über kurze Distanzen
  - Günstige Hardwarekosten („Ein-Chip-Lösung“)
- Inzwischen durch 2500 Hersteller unterstützt
- Namensgeber: Harald Blåtand (deutsch: Blauzahn, englisch Bluetooth), König von Dänemark (940-981 n. Chr), vereinigte Dänemark und Norwegen

Technische Grundlagen I

- Arbeitet im 2,4 GHz-ISM-Frequenzband (Industrial, Scientific, Medical)
- Global verfügbares und lizenzfreies Band (in Japan, Spanien und Frankreich eingeschränkt)
- Benutzt ganze Bandbreite auf 79 Kanälen mit den Frequenzen
  f = (2402 + k) MHz, k = 0..78
- Übertragung der GFSK (Gaussian Frequency Shift Keying)-modulierten Datenpakete erfolgt zeitschlitzgesteuert (TDD, Time Division Duplex) in Verbindung mit einem Frequenzsprungverfahren (FHSS, Frequency Hopping Spread Spectrum)
  - Frequenzwechsel nach jedem Paket
  - Reduzierung der Empfindlichkeit gegen Störungen (u.a. WLAN und Microwellengeräte benutzen analoge Frequenzen)
  - Hopping ist kein Sicherheitsfeature (erschwert Abhören allerdings etwas)
- Zeitschlitzlänge 625µs
- Frequenzwechselhäufigkeit (für 1-Slot-Pakete) bis zu 1600 hops/s
- Hoppingsequenz ist pseudozufällig und wiederholt sich nach ca. 23,3 Stunden

Technische Grundlagen II

- Asynchrone verbindungslose (ACL-) Übertragung für „best effort“ Verbindungen
  - Asymmetrisch: bis 732,2 KBit/s in eine und 57,6 KBit/s in andere Richtung
  - Symmetrisch: 433,9 KBit/s in beide Richtungen
  - Ab Bluetooth-2.0 Enhanced Data Rate (EDR) mit maximal 3 MBit/s
- Synchrone verbindungsorientierte (SCO-) Übertragung für Sprachverbindungen
  - Drei Kanäle mit je 64 KBit/s
  - Sprachkodierung über PCM (Puls Code Modulation) oder CVSD (Continous Variable Slope Delta-) Modulation
- Reichweiten (drei Geräteklassen)
  - Klasse 1: ca. 100 m (Sendeleistung 1-100 mW, 0 bis 20 dBm)
  - Klasse 2: ca. 10 m (Sendeleistung 0,25 bis 2,5 mW, -6 bis 4 dBm)
  - Klasse 3: ca. 0,1 bis 10 m (Sendeleistung bis 1 mW, bis 0 dBm)
- Sendeleistungsregelung (Power Control) und Stromspar-Modi (Sniff-, Park- und Hold-Mode) sind spezifiziert

Netztopologien

- Punkt-zu-Punkt-Verbindung
  - Zwischen genau zwei Bluetooth Einheiten, Master und Slave
  - Initiator wird Master
- Piconet
  - Kleines Netzwerk mit 1 Master und maximal 7 gleichzeitig aktiven Slaves
  - Benutzen einen gemeinsamen „Kanal“ (Sprungsequenz des Masters)
  - Anzahl der Slaves im Park-Mode nur abhängig vom Speicher des Chips
- Scatternet
  - Zusammenschalten von bis zu 10 Piconets zu einem Scatternet
  - Jeweils ein Gerät Master im eigenen Piconet und Slave in einem anderen Piconet
  - In der Praxis zurzeit selten genutzt

Verbindungsaufbau

- Inquiry
  - Prüfen ob sich andere Geräte im Sendebereich befinden
  - Nach dieser Phase liegen alle Zeittakte und Geräteadressen der gefundenen kommunikationsbereiten Einheiten vor
  - 48 Bit lange öffentlich bekannte und weltweit eindeutige Geräteadresse (Bluetooth Device Adress)

- Paging
  - Paging-Anforderung kann Kommunikationsverbindung aufbauen
  - Gerät das die Verbindung aufbaut wird Master
  - Für Verbindungsaufbau wird die Sprungsequenz des Slaves benutzt (Page-Hopping-Sequence)
  - Für weitere Kommunikation die Sprungsequenz des Masters (Channel-Hopping-Sequence)

Paarung (Pairing)

- Pairing muss stattfinden bevor zwei Geräte kryptographische Sicherheitsmechanismen benutzen können
- Für die beiden Geräte wird ein nur für die Verbindung dieser beiden Geräte genutzer 128 Bit langer Kombinationsschlüssel (Combination Key) erzeugt und auf beiden Geräten gespeichert
- In die Erzeugung des Kombinationsschlüssels geht von beiden Geräten jeweils eine Zufallszahl und die Geräteadressen ein
- Zur gesicherten Übertragung des Zufallszahl wird ein Initialisierungsschlüssel verwendet der aus einer weiteren (öffentlichen) Zufallszahl, einer Geräteadresse und einer PIN erzeugt wird
- PIN kann 1 bis 16 Byte lang sein und ist konfigurierbar oder fest vorgegeben
- 2 Geräte mit fest vorgegebener PIN können nicht gepaart werden

(Weitere) Verbindungsschlüssel

- Standard ist der Kombinationsschlüssel

- Geräteschlüssel (Unit Key)
  - Wird nur einmalig erzeugt und nicht mehr geändert
  - Für Geräte mit nicht genug Speicherplatz für mehr als einen Schlüssel
  - Für Geräte die viele Benutzern zugänglich sein sollen
  - Wird geschützt durch einen Initialisierungsschlüssel übertragen
  - Sehr unsicher, da sich ein Angreifer der den Schlüssel hat dann auch für dieses Gerät ausgeben kann

- Master-Schlüssel (Master Keys)
  - Für die Dauer einer Bluetooth-Sitzung zwischen mehreren Geräten (temporär) vereinbart
  - Wird über einen anderen aktuellen Verbindungsschlüssel auf die Slaves übertragen
  - Sinnvoll wenn der Master viele Slaves unter Verwendung des gleichen Schlüssels erreichen möchte (Punkt-zu-Mehrpunkt-Verbindungen)

Authentisierung

- Genutzt wird ein Challenge-Response-Verfahren auf Basis eines symmetrischen Verschlüsselungsverfahrens
- Jeweils einseitige Authentisierung (Claimant authentisiert sich beim Verifier)
- Wollen sich beide Geräte gegenseitig authentisieren mit vertauschten Rollen wiederholen
- Ablauf der Authentisierung
  - Verifier sendet Zufallszahl an den Claimant
  - Claimant berechnet eine 32 Bit Antwort aus der Zufallszahl, dem Verbindungsschlüssel und seiner Geräteadresse
  - (gleichzeitig wird ein geheimer 96 Bit langer Authenticated Cipher Offset gesendet der bei Bedarf in einen Verschlüsselungsschlüssel eingeht)
  - Verifier wiederholt die Berechnung und vergleicht die Ergebnisse
  - Sind die Ergebnisse identisch ist der Claimant authentisiert

Verschlüsselung

- Verschlüsselung ist optional möglich wenn mindestens eines der Geräte authentisiert ist
- Verschlüsselung kann von beiden Partnern beantragt werden
- Nur der Master startet die Verschlüsselung nachdem er die Parameter mit dem Slave ausgehandelt hat
- Ablauf der Punkt-zu-Punkt-Verschlüsselung
  - Länge des Schlüssels festlegen
  - Master schickt Zufallszahl an den Slave
  - Slave berechnet den Verschlüsselungsschlüssel aus dieser Zufallszahl, einem Cipher Offset (aus Authentisierung) und der Zufallszahl
  - Zur Verschlüsselung wird eine Stromchiffre genutzt (im Standard E0)
  - Jedes Datenpaket bekommt einen neuen Initialisierungsvektor („Spruchschlüssel“) aus Geräteadresse und Zeittakt des Masters
  - Daten sind nur während des Transports verschlüsselt (keine Ende-zu-Ende-Verschlüsselung auf Applikationsebene)

Sicherheitsbetriebsarten

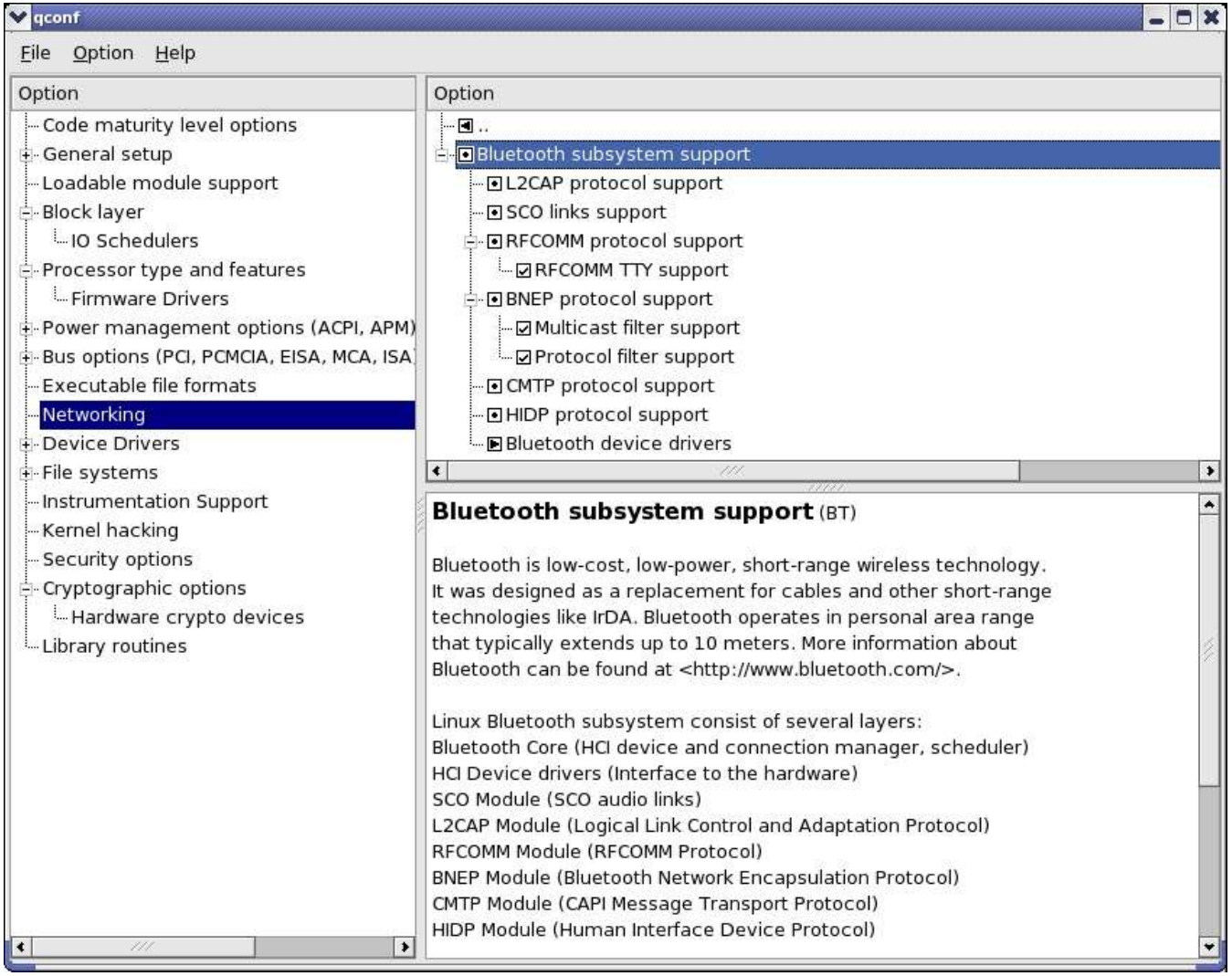
- Non-Secure Mode (Sicherheitsmodus 1)
  - Gerät initiiert keine Sicherheitsmechanismen, reagiert aber auf Authentisierungsanfragen anderer Geräte
  - Abhören lediglich durch Hopping erschwert
- Service-Level Enforced Security (Sicherheitsmodus 2)
  - Auswahl und Nutzung von Sicherheitsmechanismen abhängig vom Gerät („trusted“ bzw. „non-trusted“) und vom Dienst auf der Anwendungsebene
- Link-Level Enforced Security (Sicherheitsmodus 3)
  - Zwei Sicherheitsdienste auf Verbindungsschicht (Link Level)
  - Kryptographische Authentifikation ist generell erforderlich
  - Verschlüsselung der Daten ist optional

Bluetooth und Linux

- Bluetooth-Stack (BlueZ) ursprünglich von der Firma Qualcomm unter Federführung von Max Krasnyansky entwickelt
- Quellcode seit Mai 2001 vollständig unter der GPL
- Seit Version 2.4.6 (Juli 2001) fester Bestandteil des Kernels
- Mittlerweile von allen (gängigen) Distributionen unterstützt

Kernel Konfiguration / Zusätzliche Pakete

- Kernel Bluetooth Unterstützung

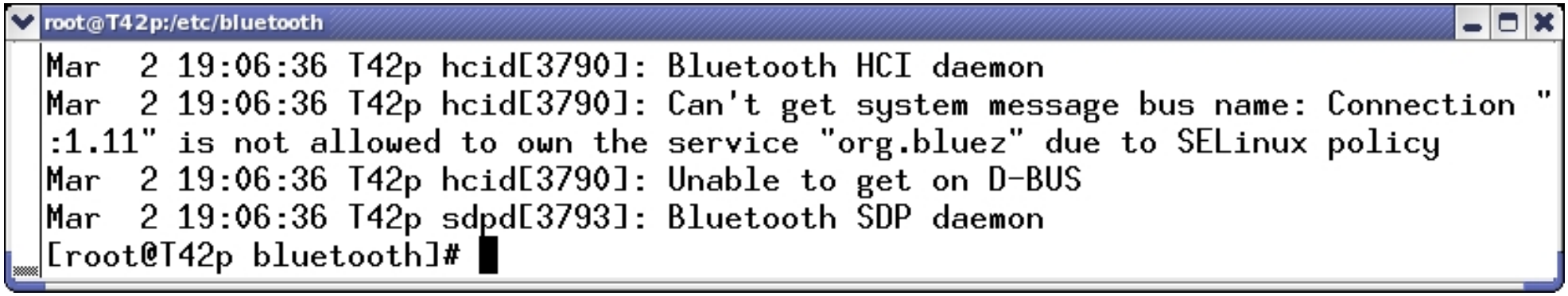


- bluez-pin (PIN helper)
- bluez-utils-cups (CUPS Backend für Drucker)
- bluez-hcidump (Debugging auf Paket-ebene)
- bluez-utils (Core und Kommandozeilenbfehle)
- bluez-libs (Bluetooth-Bibliotheken)
- bluez-bluefw (Firmware Loader)
- gnome-bluetooth (Gnome Integration)

Dienste starten (Fedora Core 5 Test 3)

- Dienste werden normalerweise beim Systemstart per /etc/init.d/bluetooth gestartet und bleiben aktiv

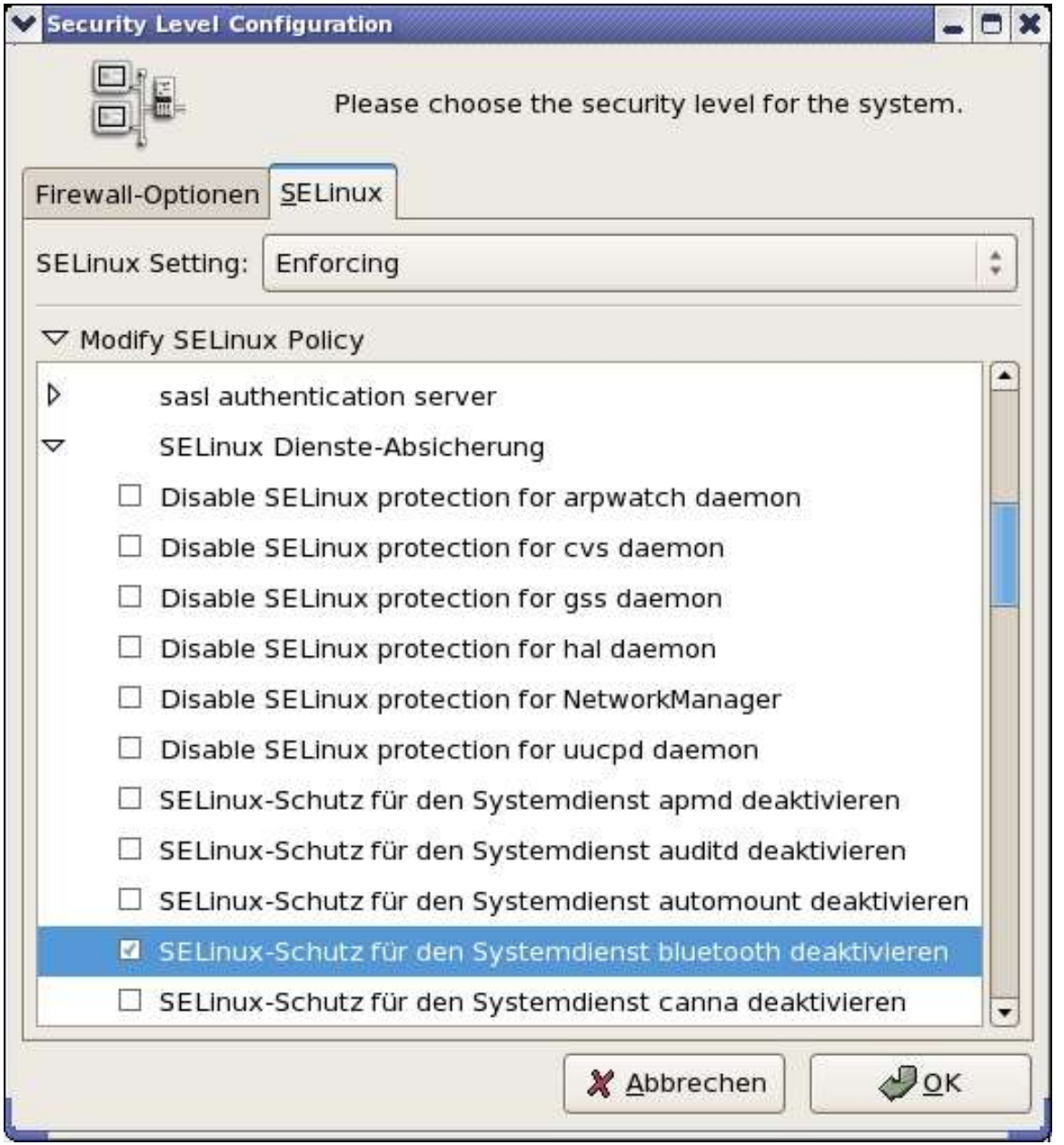
- Bei der getesteten Fedora Core 5 Test 3 gab es ein Problem mit SELinux (hier zu sehen in /var/log/messages)



- Problem wird auch unter Fedora Core 4 berichtet

Dienste starten (Fedora Core 5 Test 3)

- Per „system-config-securitylevel“ läßt sich der Schutz durch SELinux deaktivieren
- Alternativ gibt es im Internet angepasste SELinux-Policys zum Download



Dienste starten (Fedora Core 5 Test 3)

- /etc/init.d/bluetooth restart

```
root@T42p:~
[root@T42p ~]# /etc/init.d/bluetooth restart
Stopping Bluetooth services:                               [  OK  ]
Bluetooth-Dienste starten:                                 [  OK  ]
[root@T42p ~]# /etc/init.d/bluetooth status
hcid (PID 2173) wird ausgeführt...
sdpd (PID 2176) wird ausgeführt...
[root@T42p ~]#
```

- Dienste starten wie gewünscht (/var/log/messages)

```
root@T42p:~
Mar  2 21:28:22 T42p hcid[2100]: Bluetooth HCI daemon
Mar  2 21:28:22 T42p sdpd[2103]: Bluetooth SDP daemon
Mar  2 21:28:23 T42p hcid[2100]: Registering DBUS Path: /org/bluez/Device/hci0
Mar  2 21:28:23 T42p hcid[2100]: HCI dev 0 up
Mar  2 21:28:23 T42p hcid[2100]: Starting security manager 0
Mar  2 21:28:23 T42p hcid[2100]: Registering DBUS Path: /org/bluez/Manager/default/Controller
Mar  2 21:28:23 T42p hcid[2100]: Registering DBUS Path: /org/bluez/Manager/hci0/Controller
[root@T42p ~]#
```

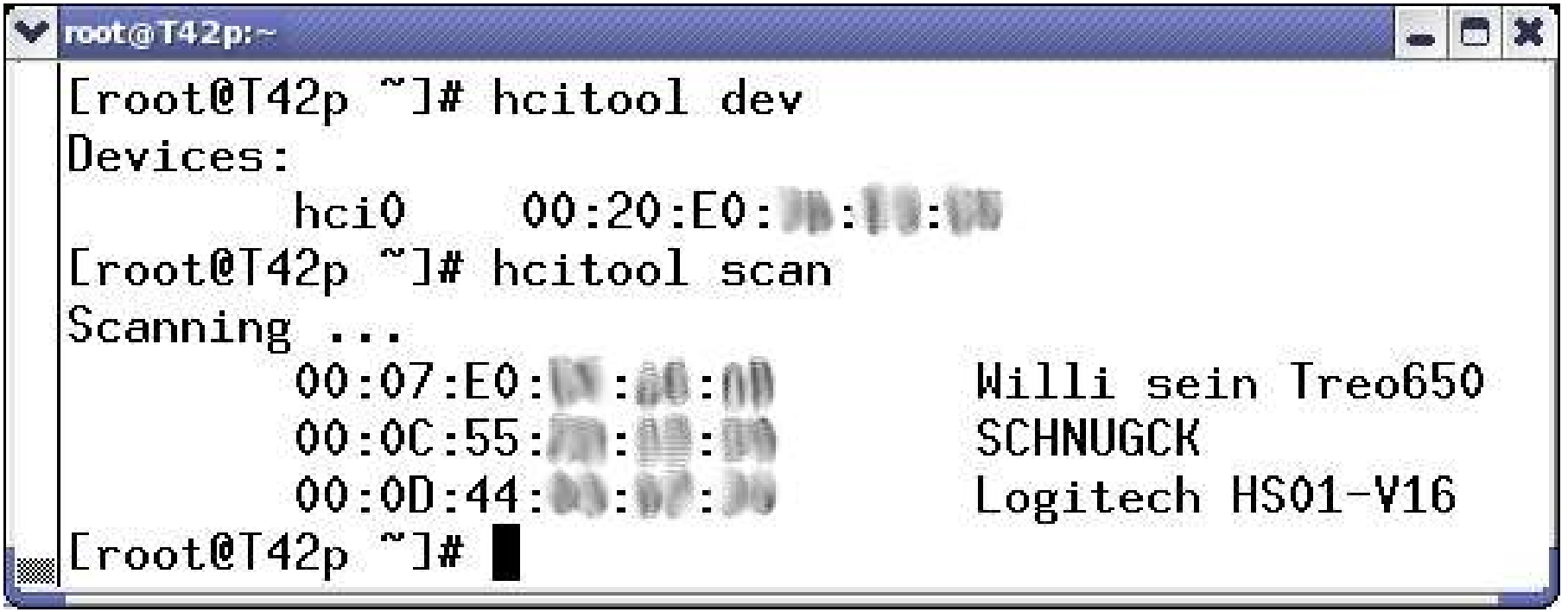
Treo 650 per Bluetooth unter Linux I

- Auf dem Mobiltelefon Bluetooth aktivieren

Bluetooth Scan

- Mit dem Werkzeug hcitool kann man jetzt die benötigten Bluetooth-Adressen bekommen

- „hcitool dev“ zeigt die Adresse des Devices auf dem Rechner an
- „hcitool scan“ sucht nach sichtbaren Geräten in Empfangsreichweite



Konfiguration Bluetooth unter Linux I

- Konfigurationsdatei /etc/bluetooth/hcid.conf

- In der „options-section“ (/usr/bin/bluepin ist ein Script das die Eingabe einer PIN erlaubt)

```
# PIN helper
pin_helper /usr/bin/bluepin;

# D-Bus PIN helper
# dbus_pin_helper;
```

- In der „device-section“

```
# Authentication and Encryption (Security Mode 3)
auth enable;
encrypt enable;
```

Konfiguration Bluetooth unter Linux II

- Konfigurationsdatei /etc/bluetooth/rfcomm.conf

- Serielles Gerät binden

```
# Automatically bind the device at startup
bind yes;
```

- Bluetooth-Geräteadresse angeben (hier Treo 650)

```
# Bluetooth address of the device
device 00:07:E0:ZZ:YY:XX;
```

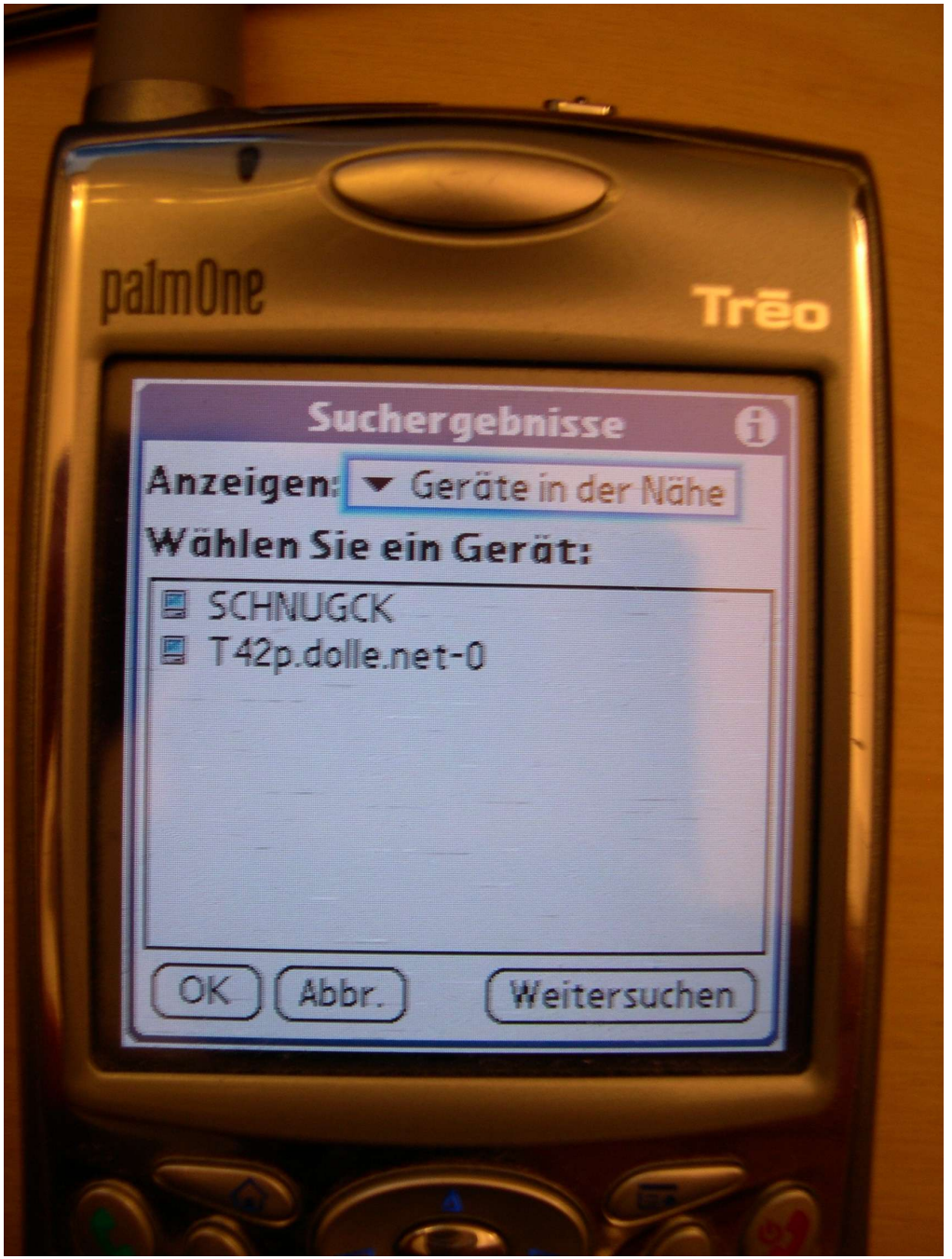
- Anschliessend Dienste neu starten

Treo 650 per Bluetooth unter Linux II

- Auf dem Mobiltelefon ein neues autorisiertes Gerät anlegen

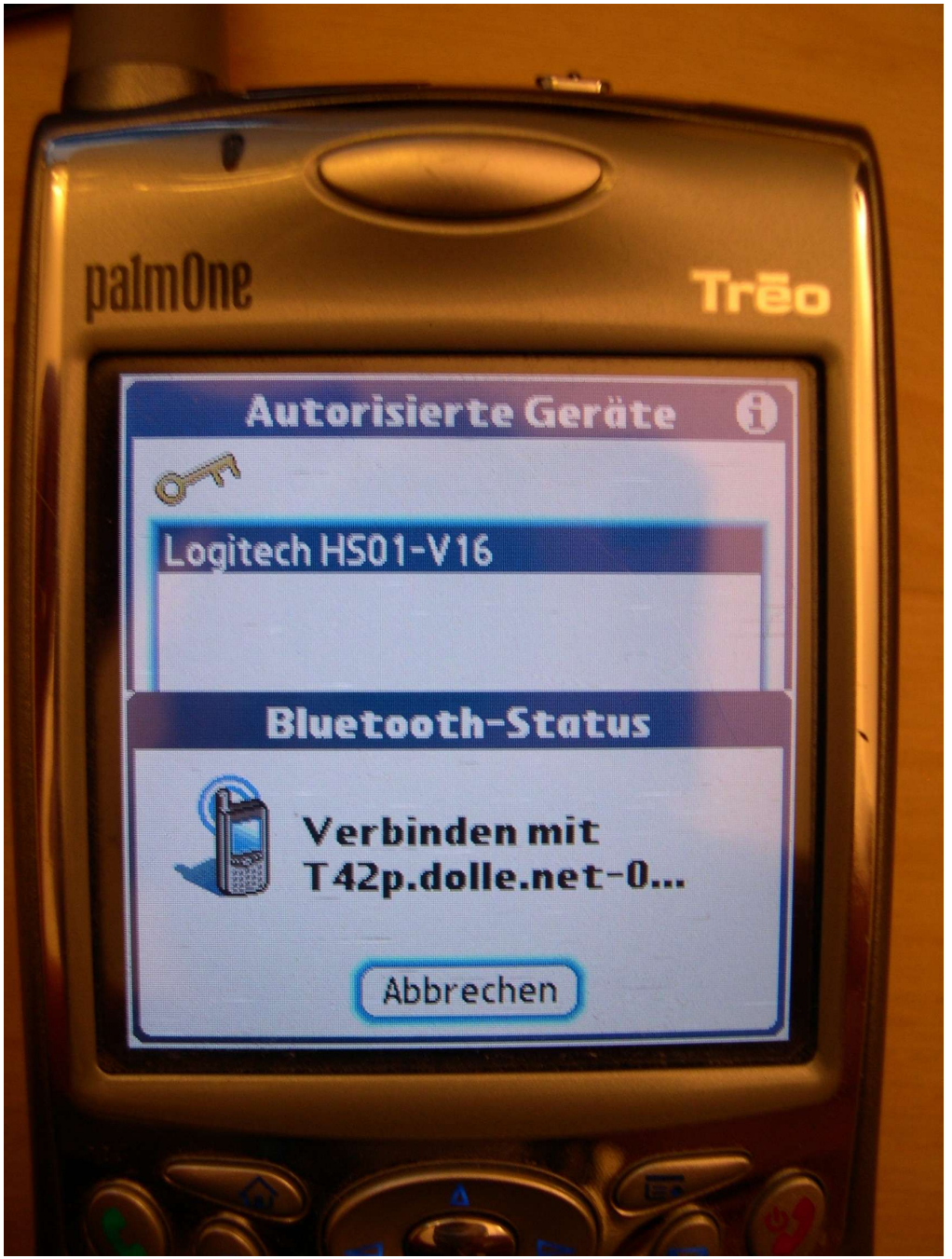
Treo 650 per Bluetooth unter Linux III

- Den Linux-Rechner suchen



Treo 650 per Bluetooth unter Linux IV

- Verbindung zum Linux-Rechner aufbauen



Treo 650 per Bluetooth unter Linux V

- Linux-Rechner kontaktieren
- Bei diesem Vorgang wird auf dem Linux-Rechner eine (temporäre) PIN vom Benutzer erfragt

- Alternativ kann eine feste PIN unter /etc/bluetooth/pin eingegeben werden
- PIN auf dem Mobiltelefon eingeben

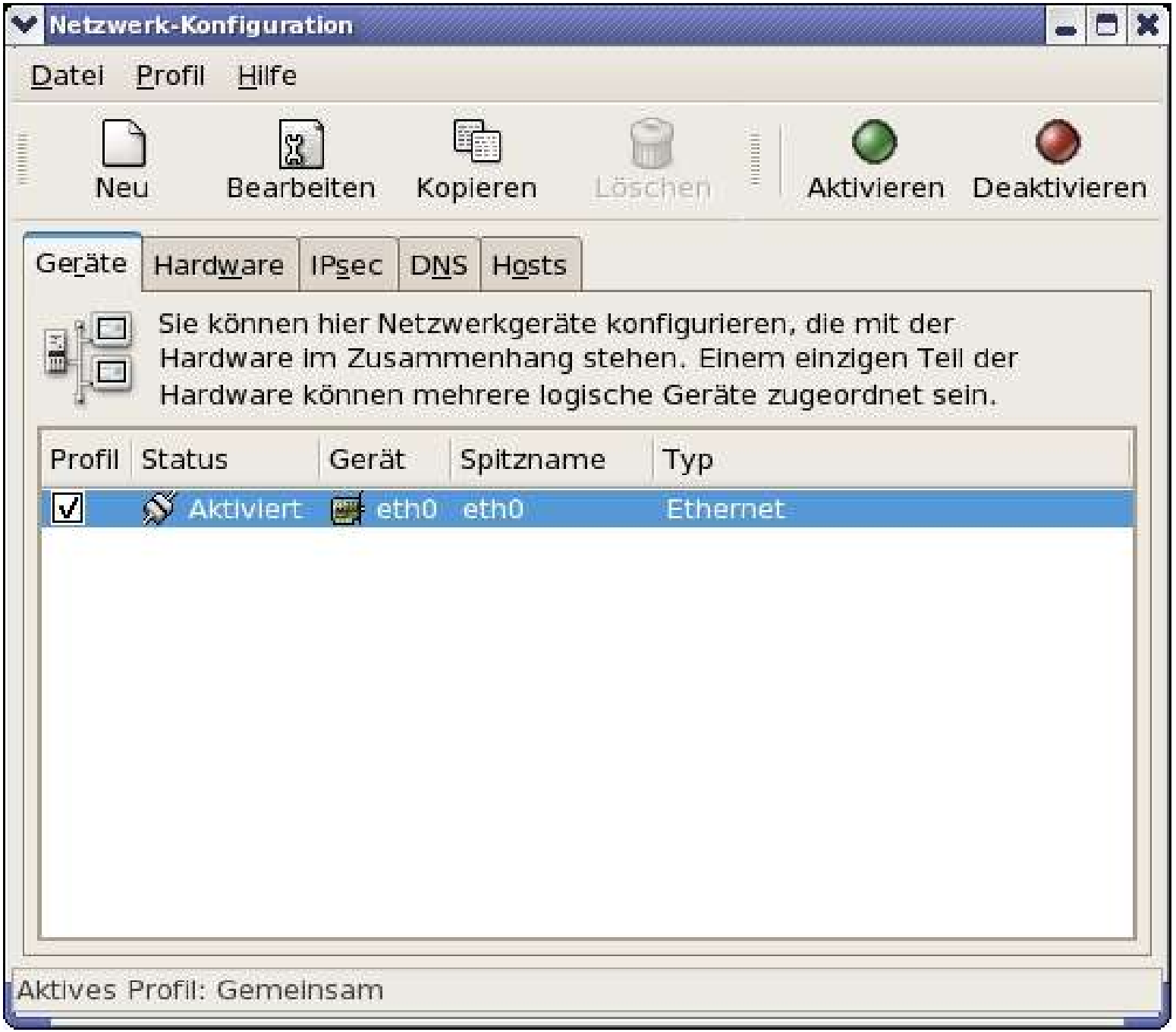
Verbindungsschlüssel unter Linux

- Die beim Scannen (per hcitool) und beim Pairing erhaltenen Informationen legt Linux unter „/var/lib/bluetooth“ in einem Verzeichnis ab, dass den Namen der Bluetooth-Adresse des Rechners hat

- In der Datei „names“ stehen die Bluetooth-Adressen und Namen der bekannten Geräte (auch der noch nicht gepaarten)

- In der Datei „linkkeys“ die beim Pairing ausgehandelten Verbindungsschlüssel mit den Bluetooth-Adressen der jeweiligen Geräte

Modem und Provider einrichten I

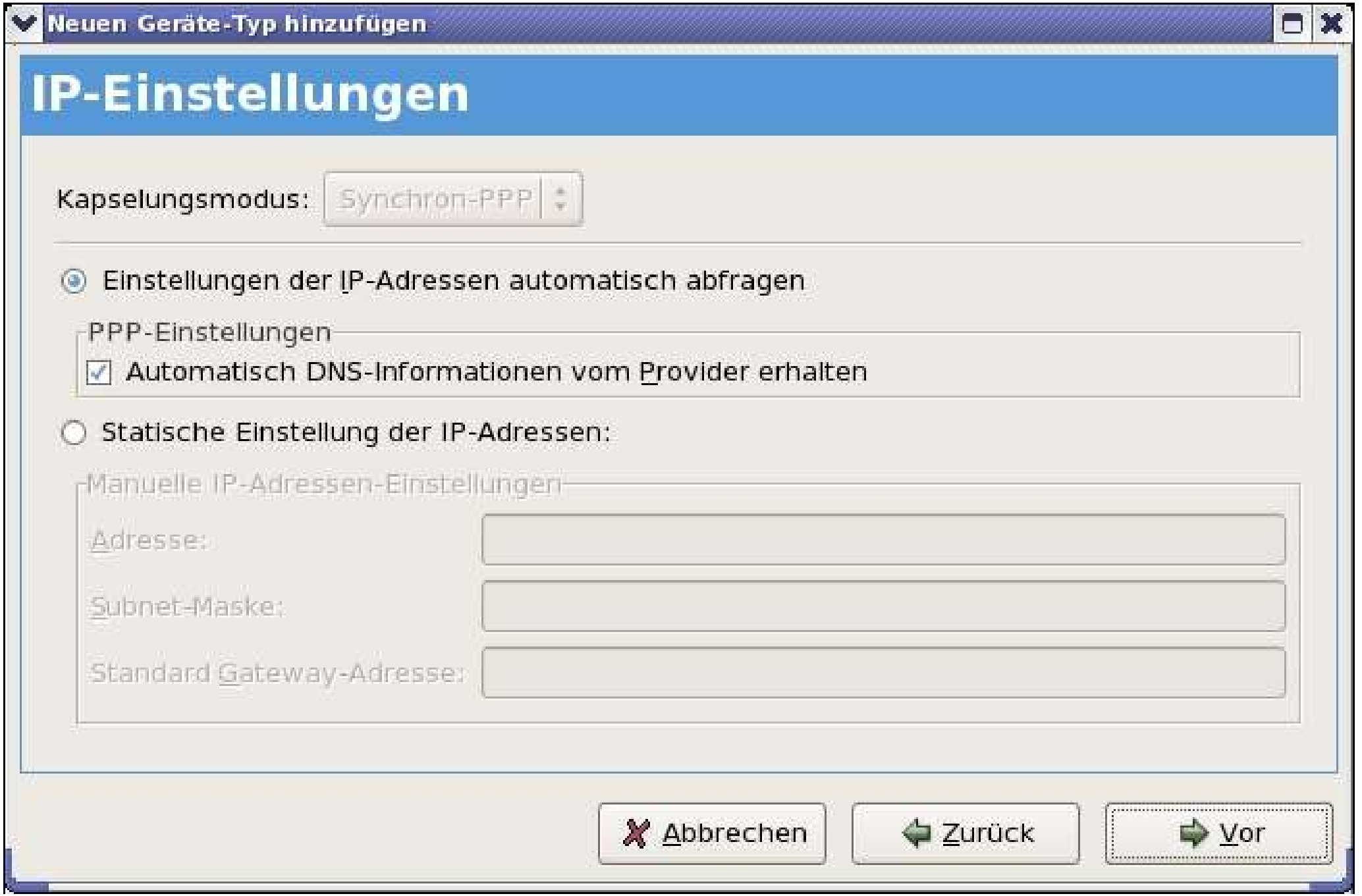


Modem und Provider einrichten II

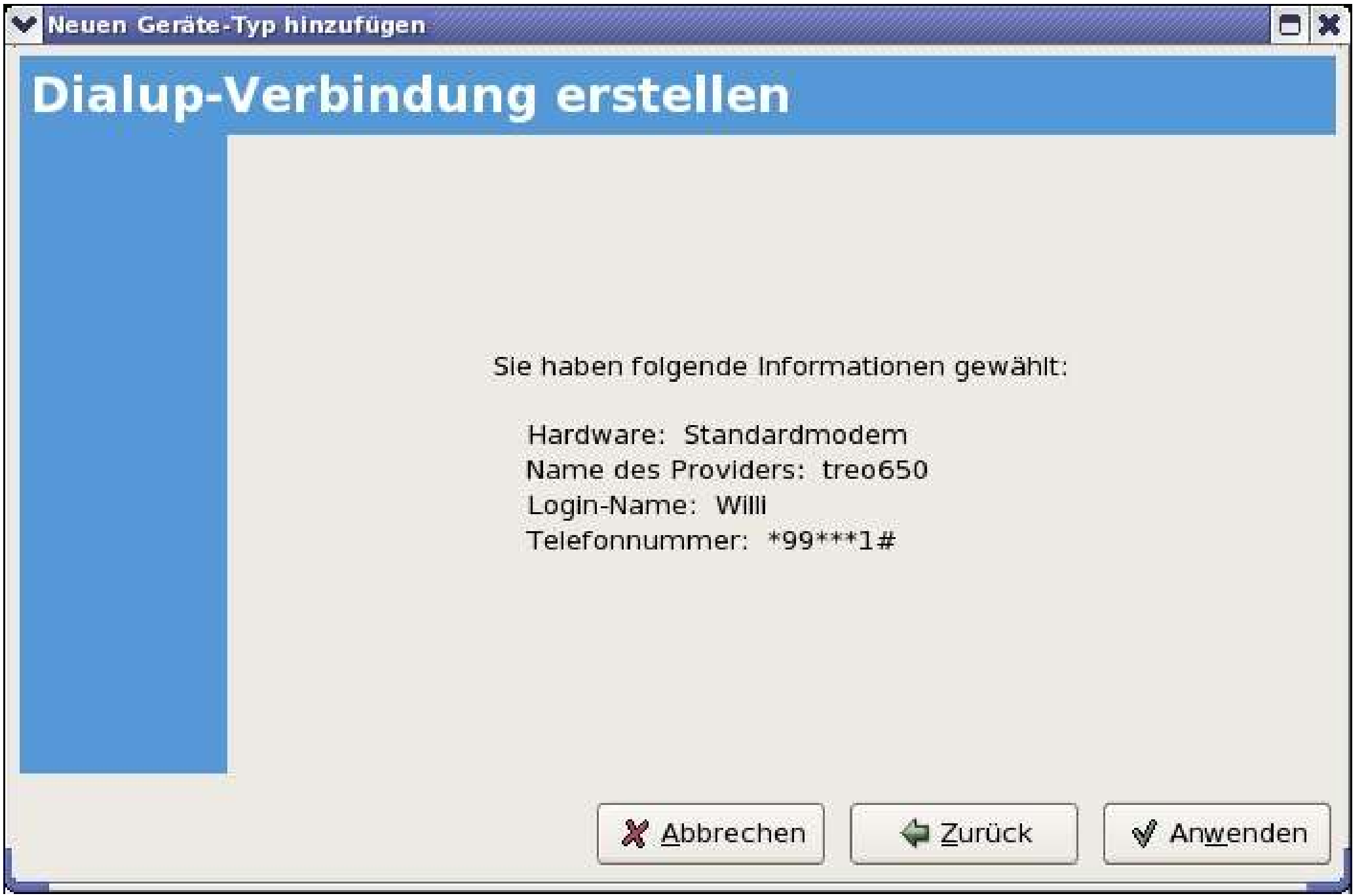
Modem und Provider einrichten III

Modem und Provider einrichten III

Modem und Provider einrichten IV

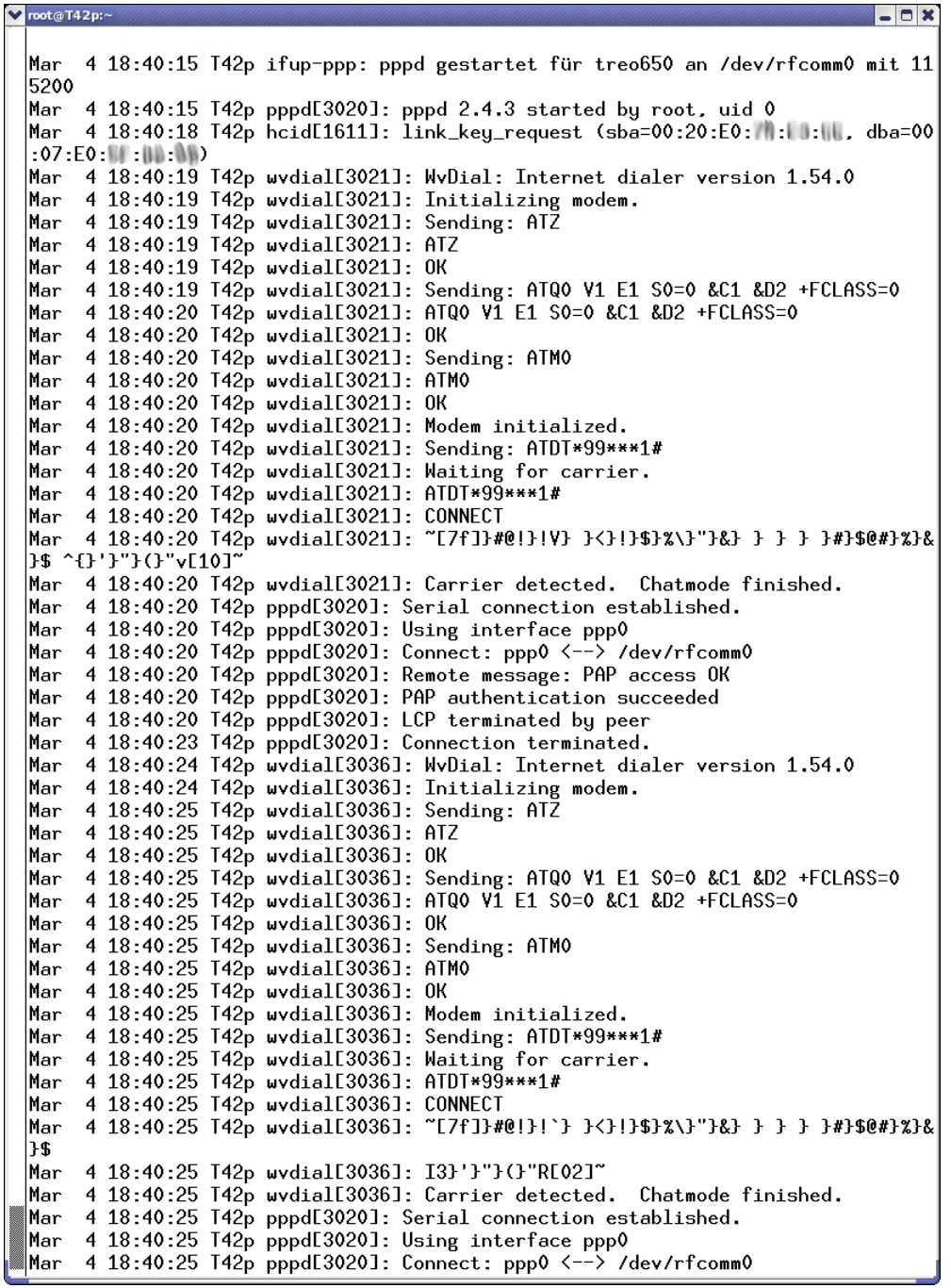


Modem und Provider einrichten V



Starten der Verbindung

- Die eingerichtete Verbindung kann jetzt wie üblich per „ifup treo650“ gestartet und mit „ifdown treo650“ beendet werden



Schwächen im Bluetooth-Sicherheitskonzept

- Verschlüsselung der übertragenen Daten ist nicht vorgeschrieben
- Unsichere Voreinstellungen sind nicht ausgeschlossen
  - Authentisierung und Verschlüsselung oft deaktiviert
  - PINs auf „0000“ oder „1234“ voreingestellt
- Schwache PINs können erraten werden
- Geräteschlüssel (Unit Keys) sind unsicher
- Schwache Integritätssicherung
  - CRC (Cyclic Redundancy Check) entdeckt zwar Übertragungsfehler aber nicht absichtliche Manipulation
  - „Sinnvolle“ Manipulation bei Stromchiffren besonders einfach
- Qualität des Zufallsgenerators nicht vorgegeben

Sicheres Betreiben von Bluetooth-Geräten

- Nicht benötigte Bluetooth-Dienste abschalten, bzw. Bluetooth deaktivieren
- Connectability, Discoverability und Pairability so weit wie möglich einschränken
- Wenn möglich die Sendeleistung so gering wie möglich halten
- Default-PINs möglichst lang und zufällig wählen
- Nach erfolgreicher Authentifizierung immer auch starke Verschlüsselung verwenden
- Schlüssellänge mindestens 64 Bit
- Als Verschlüsselungmodus nur Punkt-zu-Punkt akzeptieren
- Pairing zweier Geräte in „abhörsicherer“ Umgebung durchführen
- Geräte die mehrere Dienste mit unterschiedlichen Sicherheitsniveaus anbieten in Sicherheitsmodus 2 (Service-Level Enforced Security) betreiben
- Geräte die nur einen Dienst mit gleichem Sicherheitsniveau anbieten in Sicherheitsmodus 3 (Link-Level Enforced Security) betreiben
- Bei Verlust eines Gerätes Verbindungsschlüssel in anderen Geräten löschen

Blue spot in Berlin ☺

- Werbebotschaften werden automatisch an sichtbare Handys gesendet

Angriffsarten gegen Bluetooth-Geräte I

- Bluesnarfing
  - Lücke in Implementierung des OBEX-Protokolls (Object Exchange, wird benutzt um über Bluetooth oder Infrarot digitale Visitenkarten oder Kalendereinträge an ein Handy zu übermitteln)
  - Umgehen der Authentifizierung bzw. des Pairings durch einen Trick möglich
  - Daten aus Mobiltelefon herunterladen – Einträge aus Telefonbuch oder Kalender, SMS, Bilder, ...
  - Betroffen (mit alter Firmware): T68, T68i, T670, Z1010, R520m (Sony Ericsson) und 6310, 6310i, 7650, 8910, 8910i (Nokia)
- Bluebugging
  - Fast vollständiger Zugriff auf die AT-Kommandos des Mobiltelefons (SMS bzw. Emails senden, Gespräche führen, ...)
  - Lücke in offenen RFCOMM-Kanälen (Radio Frequency Communication) 17 und 18
  - Betroffene Handys wie oben

Angriffsarten gegen Bluetooth-Geräte II

- Bluejacking
  - Gerät wird mit Opfer gepaart und kann nicht mehr entfernt werden
  - Nachrichten an fremdes Mobiltelefon senden
  - Betroffen: 6310i, 7650 (Nokia)
- HeloMoto
  - Über unauthentisierte OBEX-Push-Verbindung mit einer Visitenkarte wird ein Abbruch erzeugt und das Gerät dauerhaft in die Liste der vertrauenswürdigen Gegenstellen eingetragen
  - Zugriff auf Headset-Profil und AT-Kommandos ohne Authentisierung
  - Betroffen: V80, V6xx, V5xx und E398 (Motorola)
- Long Distance Attacks / Blue Sniper Rifle (Abhören über bis zu 1,2 km per Richtantenne)
- Car-Whispering (Abhören und Einspielen von Daten in Mobiltelefone / Headsets innerhalb (fahrender) Autos – meist per Standard-PIN)

Fazit

- Bluetooth wird von Linux umfassend unterstützt
- Bluetooth ist eine recht sichere Technologie
- Spezifikationen erlauben unsichere Konfigurationen
- Fast alle bekannten Schwachstellen beruhen auf fehlerhaften oder schlampigen Umsetzungen des Bluetooth-Protokoll-Stacks
- Sowohl Anbieter von Bluetooth-Geräten als auch Benutzer sollten sich der Gefahren bewusst sein

Vielen Dank für die Aufmerksamkeit

Folien „zeitnah“ unter: http://www.dolle.net

Wilhelm Dolle, CISA, CISSP, BSI IT-Grundschutz-Auditor
Director Information Technology
interActive Systems GmbH

mail wilhelm.dolle@dolle.net
web http://www.dolle.net