```
                 ______                      ____
                /     /__ ___       ____    /   /___
        __ ____/     //  /  _\______\_  \  /   //  /_
      __\ \\   .____//      \_____   /   \/   //  //
       o/  0\____  \/  \ /   \__    /\       /0  /o
     _//__//_______/____Y_____\    / /___/\  \\__\\_
<<<-Holy.Church.of.0xF00000000/___/--S3M73X\__\2006-->>>
```

**- deDECTion for fun and profit v1.0 -**

5M7X@mail.ru - 2009

**Content**

## 0x00 - preface

Probably everybody know the DECT-cordless phones which give you the ability to take phonecalls without beeing bound to a wire. It‘s fine to be mobile but ist not without risks.
DECT is used for cordless phones, wireless ISDN access, babyphones, emergency calls, remotely controlled door openers, cordless EC terminals, traffic lights and also in the german ICE-train for communication and also for DECT-Computernetworks and maybe some other things.

Since DECT-cordless-phone vendors around the globe dont mark if they have implemented encryption in their phones, a customer has **NO** chance to know wether or not the product fits his personal needs. The sales-personal does not know and some customers even pretend that their hardware is eavasedroping-safe even if it is not true… well thats marketing.

So how can you find out if your phone is leaking information? Take the viewpoint of „the other side“ and attack your DECT-phone to proove ist secure.

Remember that sniffing/tapping into a phone is illegal. In germany there is a law (§201) and probably some others that may put you in jail for 5 years if you do so.

**So do only test your own phone or phones you have explicit permission to do so by the resonsible owner of that phone. I can not be held responsible for the actions one is committing using that document. This paper was written with the intention to help pentesters international to add this to their toolkit.**

## 0x01 - terms

**DECT** - Digital Enhanced Cordless Telecommunication

**DSAA** – DECT Standard Authentication Algorithm

**DSC** - DECT Standart Cipher (DECT-encryption algorithm which is held in private)

**ETSI** - European Telecommunications Standars Institute

**FP** - Fixed Part (thats the DECT-base-station)

**IPUI** - International Portable User Identity (identifies a device in a DECT-network)

**Impersonation Attack** – Fake FP and IPUI to trick a DECT-phone into using your base-station

**PCMCIA** - Personal Computer Memory Card International Association

**PIN** - Personal Identification Number (normally needed when adding PP‘s to FP‘s)

**PP** - Portable Part (thats your DECT-phone-handset)

**RSSI** - Received Signal Strength Indication
(byte-value between 0 - 255 according to IEEE 802.11 the higher the better)

**RFPI** - Radio Fixed Part Identity (thats the id of the DECT-base-station a.k.a. FP) and its subparts
ARC – Access Right Class
ARD – Access Right Details
ARI – Access Right Identity
PARI – Primary ARI
RPN – Radio Fixed Part Number
SARI – Secondary ARI
TARI – Tertiary ARI

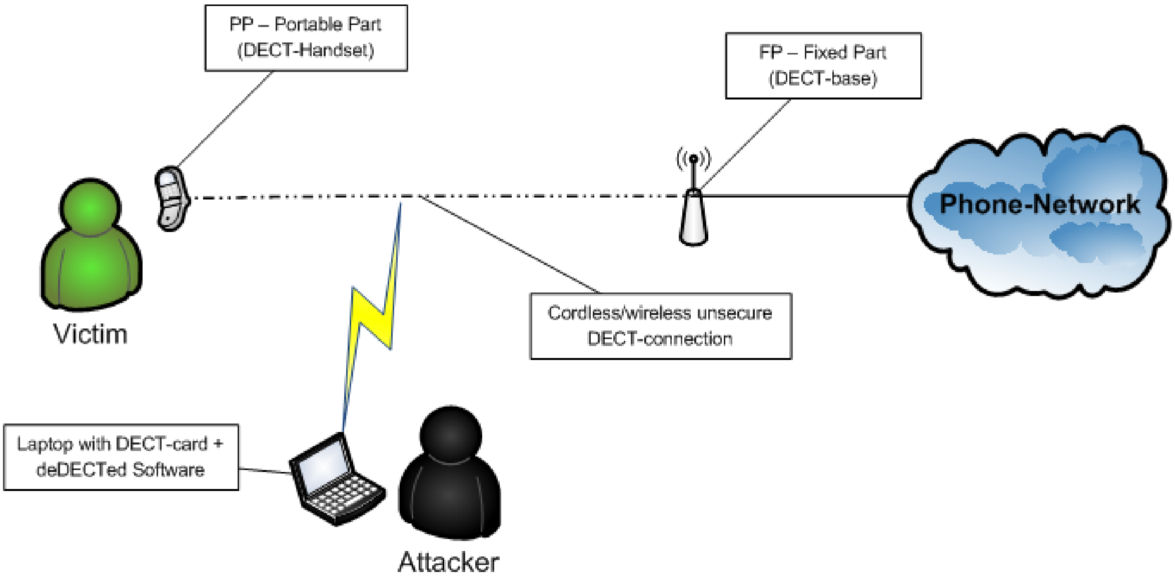
**UAK** - User Authentification Key (a secret random key to secure the communication between FP and PP)

### 0x02 - attack scenarios

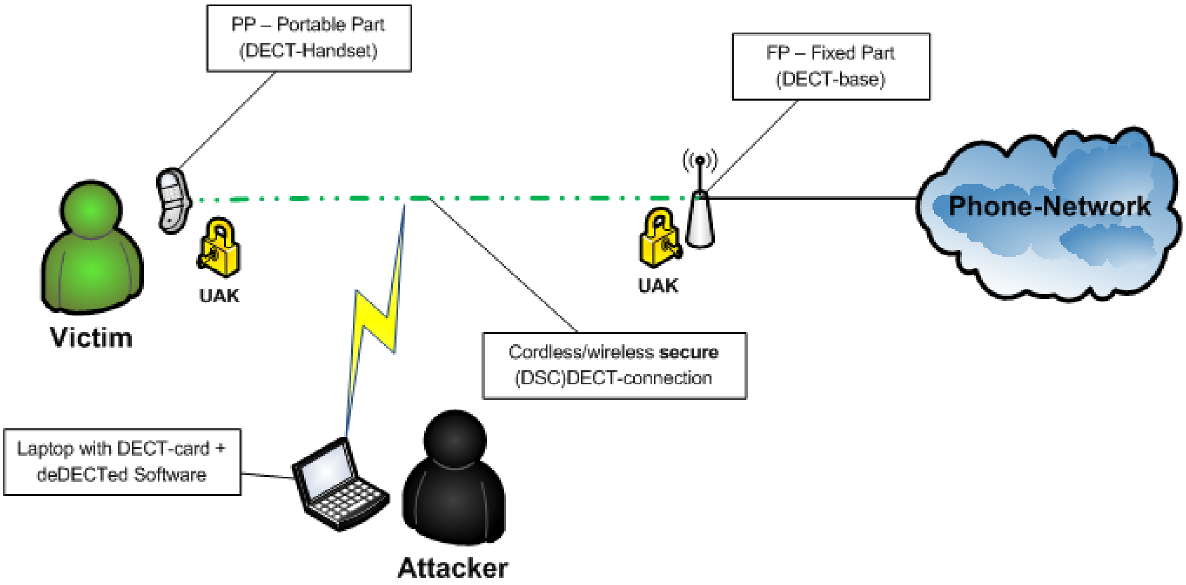
If you‘re unfamiliar with the DECT-protocol i hardly suggest you to check the references/links section at the end of this paper. You should also get into the protocol using the docs/wikipedia and WireShark to do some packet-analysis. Maybe youre also interessted in Alexandra Mengele‘s diploma work which is a much better paper then this since it also contains scientifical crypto-analysis of the algorithms. To make it short DECT has 10 channels each with a width of 1728 kHz which are located in 1880-1900 MHz. In the U.S. ist called DECT 6.0 and the range for the channels is 1920-1930 MHz.. The most interessting parts of a DECT-connection is the C-channel which holds communication-control-infos and the and the B-channel which contains the payload/speech.

### DECT-sniffing

The first scenario is the simplest one where we will just sniff the plain DECT-Connection between our PP and the FP and dump out the payload (speech/dialed numbers etc.) of the B-Channel.

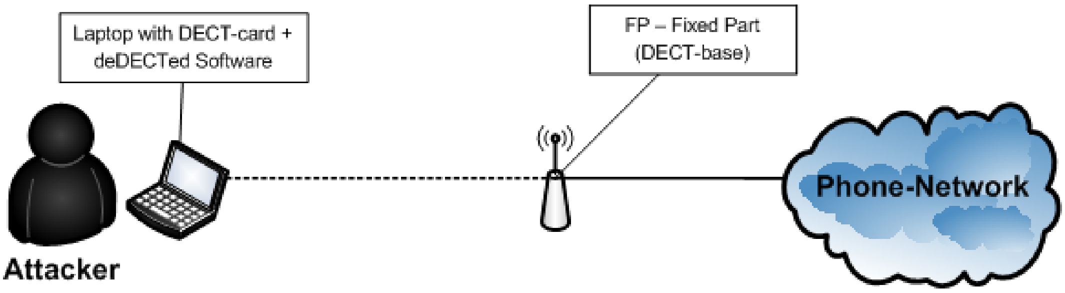


Above you see an attacker sniffing the DECT-connection and listening/examining all the data between PP and FP which is transfered in plaintext. On the bottom you see the same scenario but this time the connection is encryptet via DSC and therefore the attacker is not able the payload until he does not crack it using e.g. an FPGA-cracker (deDECTed-devs are allready reversing the DSC-crypto-algorithm).



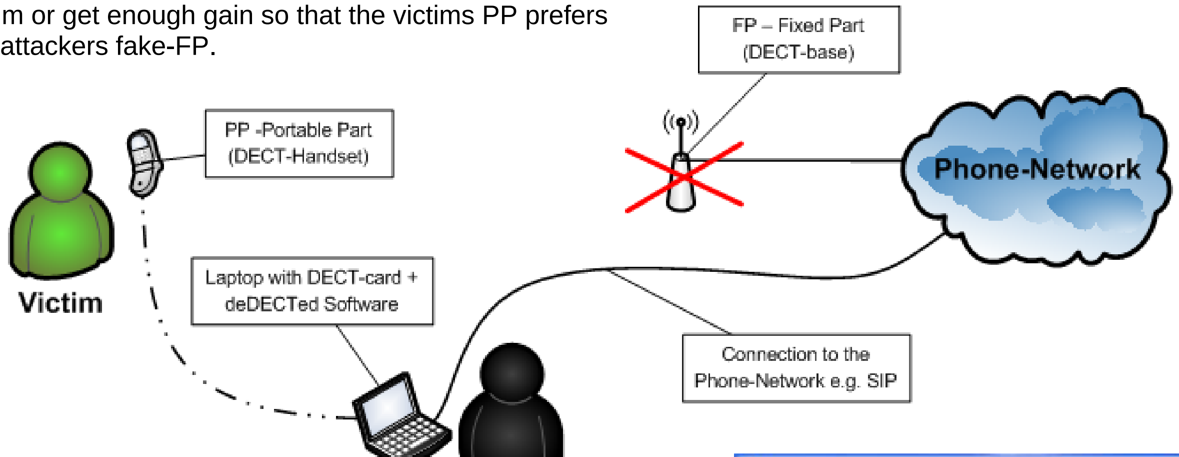
### Attacking the base

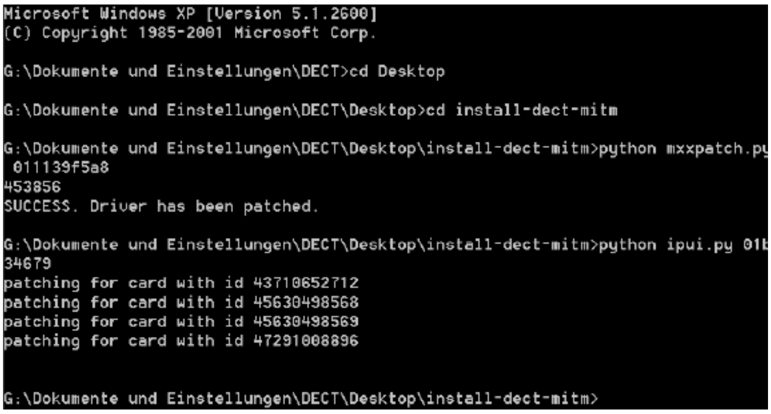
In this scenario the FP does not force the PP to authenticate itself or we have cracked the UAK of a PP so we are able to trick the base into a situation where it thinks we are allowed to use it for making calls. A successfull attack would allow one to do phone-calls via other ppls base-stations (deDECTed-devs are allready attacking the authentication algorithm).



### Impersonating Attack / MITM-Attack

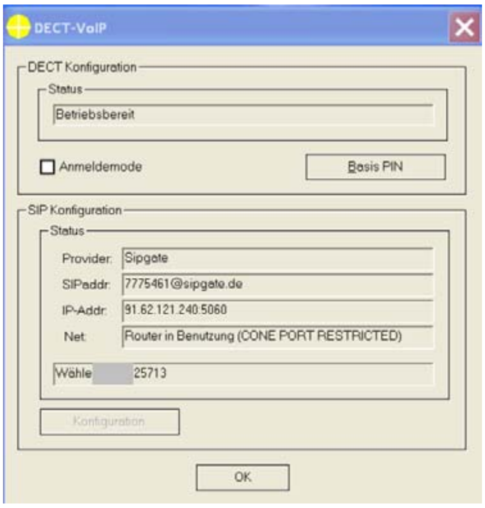
In this scenario we found out the IPUI and the RFPI either by searching all the C-Channel-Data or by using a Fritz!Box 7270 onto which the phone was authed. Then the attacker patches ist card-driver to use the IPUI and RFPI of the FP paired with the PP. Then the attacker has to DDoS the FP or increase gain of his DECT-Card so the Victims PP prefers using the attackers-fake-base instead of his own real base. If the phone uses encryption it can be forced (downgrade-attack) into plaintext-communication. So all information can be retrieved out of that connection. This attack is more complex and its not that easy to DDoS the FP of the victim or get enough gain so that the victims PP prefers the attackers fake-FP.





```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

G:\Dokumente und Einstellungen\DECT>cd Desktop

G:\Dokumente und Einstellungen\DECT\Desktop>cd install-dect-mitm

G:\Dokumente und Einstellungen\DECT\Desktop\install-dect-mitm>python mxxpatch.py
 011139f5a8
453856
SUCCESS. Driver has been patched.

G:\Dokumente und Einstellungen\DECT\Desktop\install-dect-mitm>python ipui.py 01b
34679
patching for card with id 43710652712
patching for card with id 45630498568
patching for card with id 45630498569
patching for card with id 47291008896


G:\Dokumente und Einstellungen\DECT\Desktop\install-dect-mitm>
```

**patching RFPI and IPUI into the driver (A. Mengele)**



DECT-VoIP

DECT Konfiguration

Status: Betriebsbereit

☐ Anmeldemode

SIP Konfiguration

Status

Provider: Sipgate
SIPaddr: 7775461@sipgate.de
IP-Addr: 91.62.121.240:5060
Net: Router in Benutzung (CONE PORT RESTRICTED)

Wähle [redacted] 25713

**Successful impersonation attack (A. Mengele)**

## 0x03 - hardware

***PCMCIA-DECT-cards***

The people from deDECTed.org developed a driver for the „DOSCH & AMAND - COM-ON-AIR“ Type II-cards (a.k.a. CoA) which puts it into some kinda monitor mode. Their research was presented at the 25C3 (a yearly german hacker-convention) and can be reviews at the deDECTed-wiki . Unfortunately these cards are now being traded far beyond their market-prices because of their new usage. Their current pricing (if your lucky enough to find one on ebay.de) is around 50 - 300,-EU where one has to know that these cards initially where priced around 30,-EU.

So i researched for an other affordable card to be used for my experiments.
Somewhere in the deDECTed-ticket-system i came over the „ASCOM - Voo:doo“ which only costs about 30,-EU. The funny thing is that it internally the same as the CoA Type III card so since the support for Type-III-cards has been included into the deDECTed-driver/software this card does also work. But unfortunately it seems that the „original“ D&A CoA Type-II card has a better range since the driver has been cleaner developed for that.

I bought two of the Voo:doo-Cards with two different hardware-revisions. The first one is a „KED“ and the second one is a „KEE“ i will test both devices in this paper. I also physically disassembled the card since i wanted to have an insight-view.

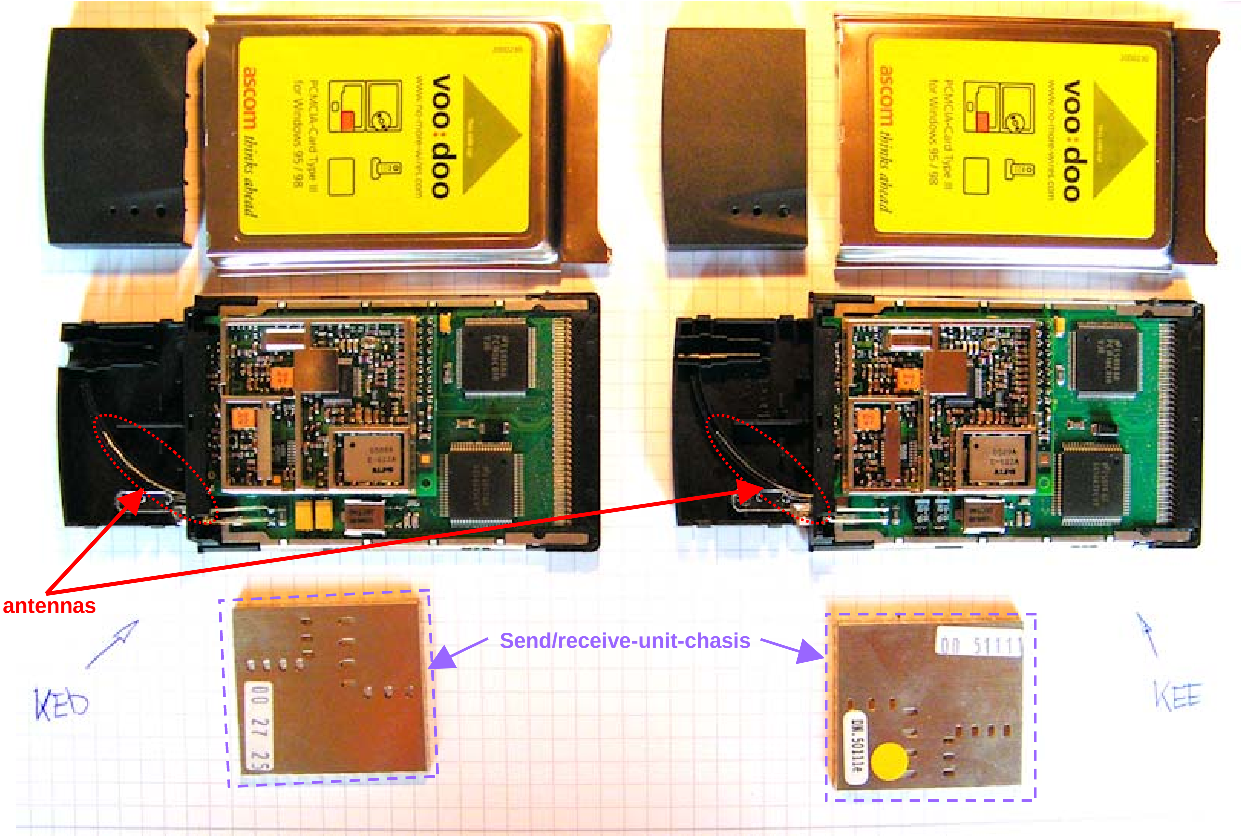
**Ascom Voo:doo Typ III PCMCIA (ARC-Computer)**

**Ascom Voo:doo Typ III PCMCIA (ARC-Computer)**

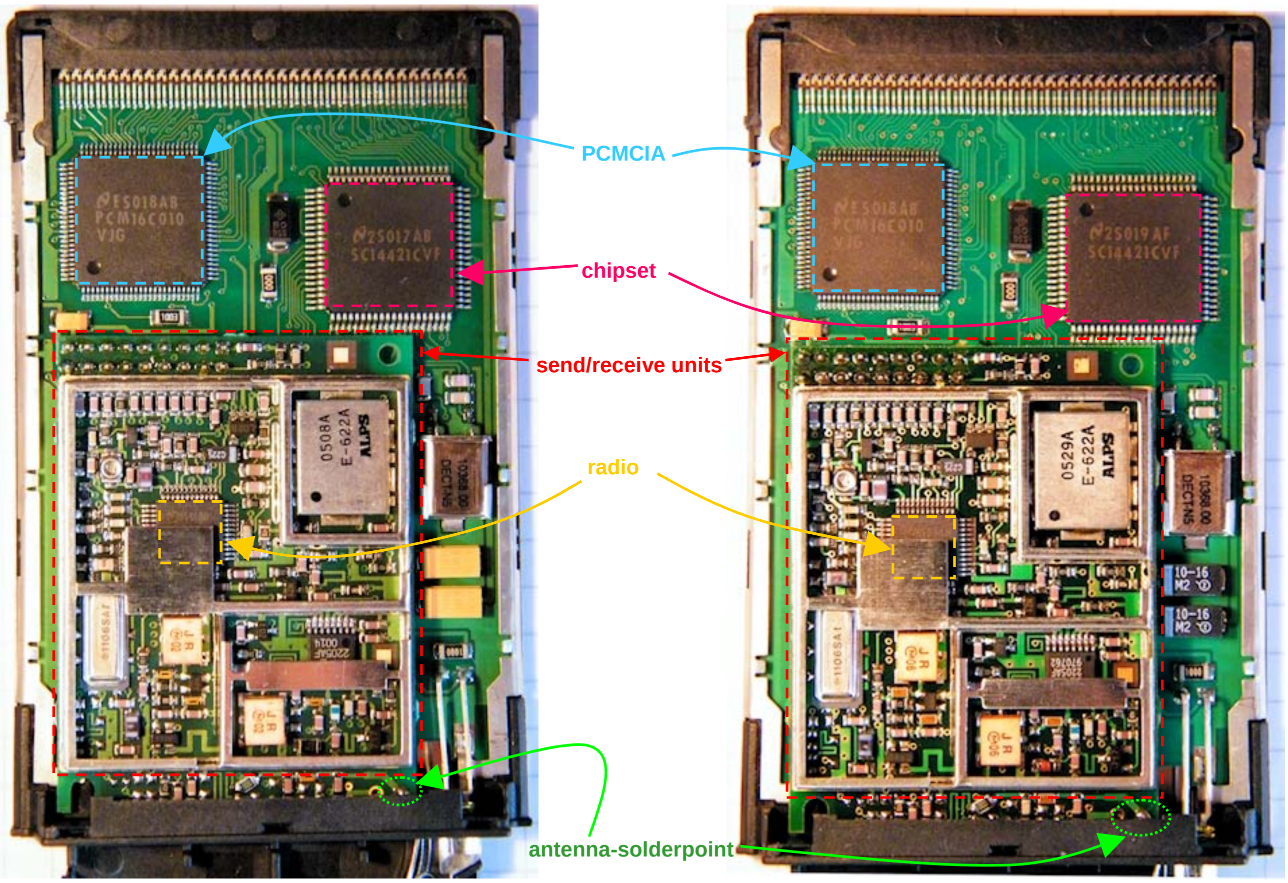
**Ascom Voo:doo Typ III PCMCIA package after opening**

The card can ship in two different optical tastes. Te left one is what i use in this tutorial and the right one is the same but looks more similar to the DOSCH & AMAND COM-ON-AIR Type III cards.

On the left you see the package after opening it. It contains the PCMCIA-DECT-card, a short installation-manual with a serial on ist back for the windows-software and a CD-ROM with drivers for Win95/98, the RVS-COM lite Software and documentation as PDF. The Software seems to have some interessting features but unfortunately i was not able to install it under WindowsXP (even not using win95/98-compatibility-mode) so i have to install win95/98 some day to test it out *ugh!*. The software may give you the abilty to create a FP-station at least the original CoA-software seems to be able to do so (https://dedected.org/trac/wiki/COM-ON-AIR).



My voo:doo's after disassembling. On the left the KED-Hardware-Revision on the right the KEE-Revision



Voo:doo KED in detail (note the chip-identifications)

Voo:doo KEE in detail (note the chip-identifications)

On the right you can see a table with some informations about the internal-hardware of the Ascom Voo:doo Card which i have found on my card.

I added missing info from a deDECTed-track-ticket where more info about other cards and chipsets can be found.

The Chips are from National Semiconductors. Datasheet for the Radio-Chip can be found at the datasheetcatalog-page.

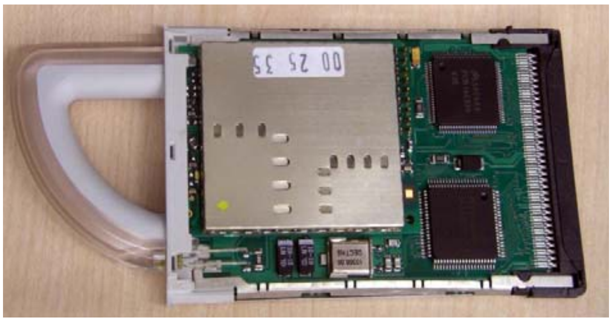
| | |
|---|---|
| Type: | III |
| Radio | NatSemi LMX3161 |
| Chipset: | SC14421CVF |
| EEPROM: | 93LC86 (16k) |
| PCMCIA: | PCM16010 |
| Brand | Ascom Voo:doo |
| Remarks | Different antenna then DOSCH AMAND COM-ON-AIR-Type-III-card but the rest is similar |

**COA Type II with external antenna mod by Erik Tews**

Now here you can see the „original“ Dosch & Amand COM-ON-AIR DECT-Cards used by the deDECTed-research-team. The pictures have also been taken from the deDECTed-Homepage.

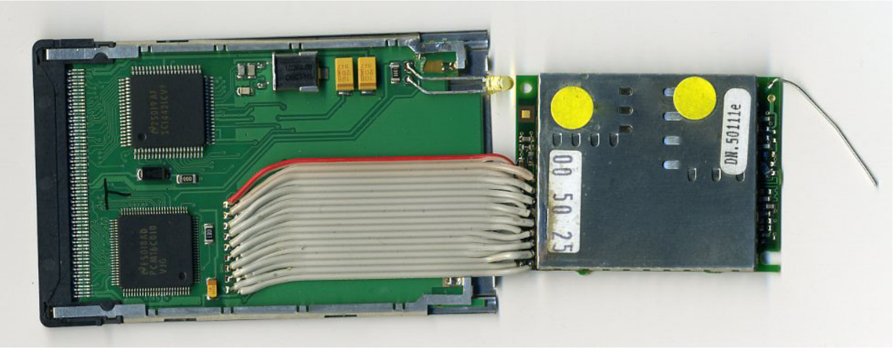
**COM-ON-AIR Type II PCMCIA-card**

**COM-ON-AIR Type III PCMCIA-card**

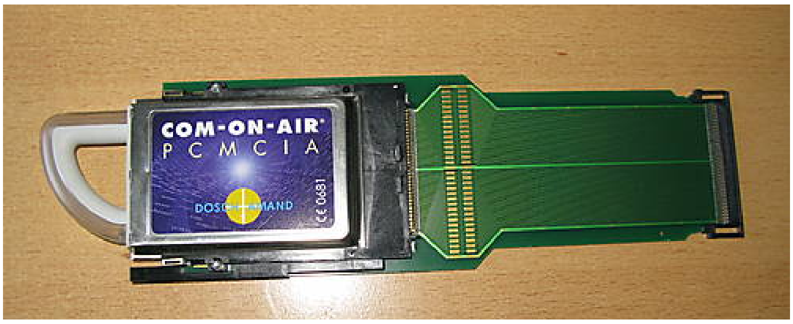


**COM-ON-AIR Type III card after opening it**

On the right side you can see a Type-III-card-mod where somebody desoldered the send/receive-unit and reconnected it using an old IDE -cable to make it fit into a Type-II-slot.



**Type III to Type II mod**



CoA Type III to Type II Adapter (ebay)

DECT-antenna 12dBi biquad (brennpunkt-srl.de)

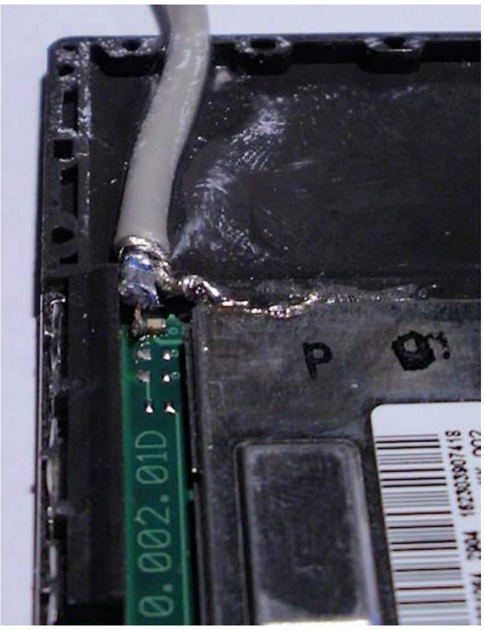
It's also possible to add an external antenna what makes sense especially on the Ascom Voo:doo-cards since their range is very limited probably because of some missing tweaks in the deDECTed-software.
Those antennas can be obtained for about 20,-EU to 30,-EU.

You can also built one yourself like the BiQuad-antenna out of an old CD/DVD-box you just have to calculate the length of the antenna-sides out of the DECT-frequency. Information about that can be found at mydarc.

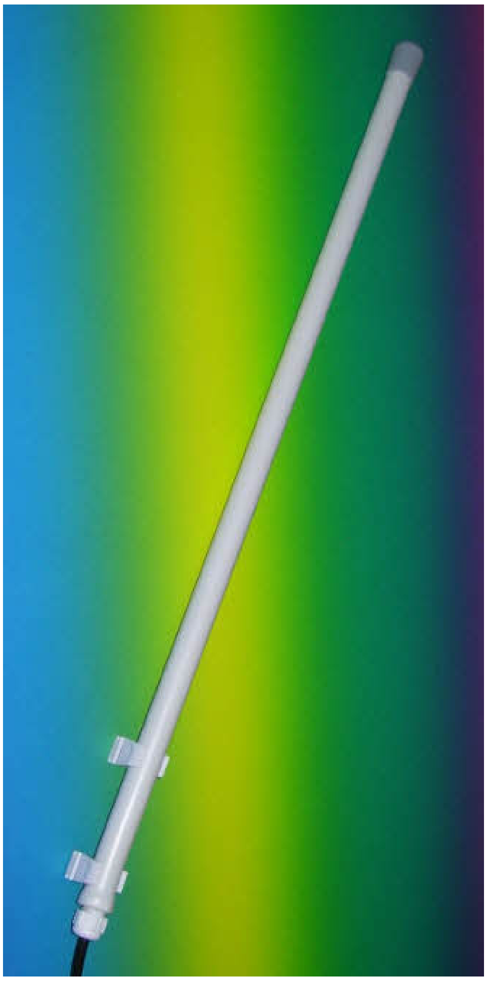
There is also a nice software which helps you design and simulating all kind of antennas which is free for private use and can be found at mmhamsoft.amateur-radio.ca.

You should be familiar with soldering if you want to add one and of course you will void your waranty!

You also can built your own Type-III to Type-II adapter out of an old IDE-cable if you want since ist a simple extension nothing else.



external antenna soldered (Tews)



DECT antenna 9dBi omni (brennpunkt-srl.de)

### *Laptop*

I am using an old „IBM Thinkpad T41“ Laptop which i bought from ebay around 200,-EU disassembled/cleaned it, added fresh thermal grease to the cpu-heatpipe, 1 GB RAM, a new extended accumulator (the cheap ones from china ;D) and now it works pretty fine for me. In fact it doesnt matter which one u use as long as it runs linux and has a PCMCIA-Type III slot. You can also use a Type II slot with an adapter or an PC with an PCI-PCMCIA-Adapter.

You can also go the hard way and desolder the sender/receiver unit from a Type-III-card to flaten it so it fits in a Type-II-Slot if you feel bored.

**IBM T41 Notebook**

### *DECT-Phones*

I am testing the attacks on the following DECT-phone hardware:
Siemens Gigaset S452 which consists of a base (FP) a handset-phone (PP) and i also used a Siemens Gigaset 4000L Comfort as a Client to the S452 base. In that test the calls where encryptet. But in a seperate test i had a Gigaset A250 on which i was able to obtain the speech since ist not using encryption.

After some research i found out that i can also obtain the speech from the S452 and the 4000L Comfort when setting the S452 base-station to go into the „repeater“-mode which makes all calls be in plaintext.
I did that using the Gigaset S452-PP since i found that point in some settings-submenue to be able to have a working setup for my experiment.

**Siemens Gigaset S452**

**Gigaset 4000L Comfort**

**Siemens Gigaset A250**

If you want to know wheter or not your phone is using encryption and on which attacks it may be vulnerable but you dont want to go through that effort of testing it then the following references are for you.

Siemens official list of vulnerable phones (contains false info according to the computerBILD-tests)
https://dedected.org/trac/blog/list-of-phones-progress
https://dedected.org/trac/wiki/ListOfPhones
computerBILD sponsors students to test 50 phones for vuln
Alexandra Mengeles Diploma Thesis

But the best way to verify this is of course to sniff your DECT-phone yourself to be sure ist at least safe against this kind of attack. If you have more confidental information you want to talk about it would be best to use a wired-phone or even better one that uses-end-to-end-encryption e.g. secure SIP.

## 0x04 - software

The Software used in this paper is the one from the deDECTed.org and i use the most current via a svn-checkout. Packages for BackTrack4 will soon be added to the repository by muts (offsec-blog).

First of i tested the software on Xubuntu 9.04 x86 32bit on my laptop and that worked just fine.
In here i will describe the installation for the world most famous pentesting-liveCD BackTrack in version 3. You will also find a script, which does that automaticially for you, attached to this paper. If you dont know BackTrack take a look at www.remote-exploit.org and www.offensive-security.com to learn about it.

It is  a LiveCD whereas BackTrack 3 is based on Slackware12/SLAX-LiveCD and BackTrack 4 is based on Ubuntu 8.04 LTS which ships with the common tools for hacking-attacks which are used by it-security professionals to verify the (in)security of the companies network in a legal pentest.

There is also the CHAOX-project which is also a liveCD with all deDECTed-stuff preinstalled.

### Install deDECTed driver and software  on BackTrack 3 (liveCD)

First of all you need the Sources for the Linux-Kernel that is used within BackTrack3 since they are not included within the BackTrack3-LiveCD. You may retrieve em using Firefox, lynx, curl or wget in a shell. This is needed to built the driver/kernel-module that controls the dect-card.

```
wget http://www.offensive-security.com/kernel.lzm
```

After that the kernel is installed via lzm2dir into the BackTrack3-directory-structure

```
lzm2dir kernel.lzm /
```

Now we make a new „dect“ folder in the linux-kernel-sources dir

```
cd /usr/src/
mkdir dect
cd dect
```

After that we will checkout the latest dedected software-sources directly from the subversion-repository. Note that it may cause trouble sometimes to use bleeding-edge software and if you are a linux-newb without any C-programming-knowledge i would suggest to use an oder „stable“ version if it problems occur. In this tutorial i used revision 89.

Note also that you will receive a ssl-certificate warning while the checkout starts. Just press „p“ to permanently accept the key of the dedected.org-SVN-server.

```
svn co https://dedected.org/svn/trunk deDECTed
```

It's time to compile the driver and create a new device in the filesystem which represents our dect-card which will be */dev/coa*.

```
cd deDECTed/com-on-air_cs-linux
make && make –C tools
make node
```

If you havent allready done so put in your pcmcia-dect-card into your computer **NOW!**
Then load the driver/kernelmodule. To unload it do a *rmmod com_on_air_cs.ko*

```
insmod com_on_air_cs.ko
```

Alternatively you can use the *make load* and *make unload* to load/unload the driver

Now check the dmesg output

```
dmesg | tail –n 37
```

If you get an output simillar to the following the card and its drivers are probably loaded correctly.

```
pccard: PCMCIA card inserted into slot 1
cs: memory probe 0xe8000000-0xefffffff: excluding 0xe8000000-0xefffffff
cs: memory probe 0xc0200000-0xcfffffff: excluding 0xc0200000-0xc11fffff
0xc1a00000-0xc21fffff 0xc2a00000-0xc31fffff 0xc3a00000-0xcc1fffff 0xcca00000-
0xcd1fffff 0xcda00000-0xce1fffff 0xcea00000-0xcf1fffff 0xcfa00000-0xd01fffff
pcmcia: registering new device pcmcia1.0
pccard: card ejected from slot 1
pccard: PCMCIA card inserted into slot 1
pcmcia: registering new device pcmcia1.0
>>> loading com_on_air_cs
com_on_air_cs: >>>>>>>>>>>>>>>>>>>>>>>>
com_on_air_cs: card in slot        com_on_air_cs
com_on_air_cs: prod_id[0]          DOSCH-AMAND
com_on_air_cs: prod_id[1]          MMAP PCMCIA
com_on_air_cs: prod_id[2]          MXM500
com_on_air_cs: prod_id[3]          V1.00
com_on_air_cs: ioremap()'d baseaddr f932e000
com_on_air_cs: registered IRQ 3
com_on_air_cs: valid client.
com_on_air_cs: type          0x118
com_on_air_cs: function      0x0
com_on_air_cs: Attributes    1
com_on_air_cs: IntType       2
com_on_air_cs: ConfigBase    0x1020
com_on_air_cs: Status 0, Pin 0, Copy 0, ExtStatus 0
com_on_air_cs: Present       1
com_on_air_cs: AssignedIRQ   0x3
com_on_air_cs: IRQAttributes 0x12
com_on_air_cs: BasePort1     0x0
com_on_air_cs: NumPorts1     0x10
com_on_air_cs: Attributes1   0x10
com_on_air_cs: BasePort2     0x0
com_on_air_cs: NumPorts2     0x0
com_on_air_cs: Attributes2   0x0
com_on_air_cs: IOAddrLines   0x0
com_on_air_cs: has function_config
com_on_air_cs: get_card_id() = 2
com_on_air_cs: -----------------------
```

Card-status can also be checked using *pccardctl status* and *pcccardctl ident*

```
pccardctl status
Socket 0:
  no card
Socket 1:
  5.0V 16-bit PC Card
  Subdevice 0 (function 0) bound to driver "com_on_air_cs"
```

Now the card should just work fine and you can proceed with the „deDECTion“-chapter if you want.

You also probably want to make this a module via the *dir2mo* command tutorials for that can be found in the remote-exploit-forums and the BackTrack-wiki.

### Autoload drivers in BackTrack3

If you dont want to load/unload the driver/kernelmodule each time but want it to load automaticially when u put in the pcmcia-card just follow these steps.

First of we have to copy the sources of the deDECTed-driver from */usr/src/dect/* into the kernel-source-folder and update dependencies.

```
cd /usr/src/dect/dedected/com-on-air_cs-linux/
cp com_on_air_cs.ko /lib/modules/$(uname –r)/kernel/drivers/
depmod –a
```

To have the driver automaticially loaded by the *udev*-system when you put in the card into the PCMCIA-slot you have to set a rule. Start with reading out the *udevinfo* properties of the currently plugged in card.

```
udevinfo -a -p /sys/bus/pcmcia/devices/1.0

Udevinfo starts with the device specified by the devpath and then
walks up the chain of parent devices. It prints for every device
found, all possible attributes in the udev rules key format.
A rule to match, can be composed by the attributes of the device
and the attributes from one single parent device.

  looking at device '/devices/pci0000:00/0000:00:1e.0/0000:02:00.1/1.0':
    KERNEL=="1.0"
    SUBSYSTEM=="pcmcia"
    DRIVER==""
    ATTR{modalias}
=="pcmcia:m0204c0000fFEfn00pfn00pa4BC552E7pb0DF519BBpc09E43C7Cpd3488C81A"
    ATTR{prod_id4}=="V1.00"
    ATTR{prod_id3}=="MXM500"
    ATTR{prod_id2}=="MMAP PCMCIA"
    ATTR{prod_id1}=="DOSCH-AMAND"
    ATTR{card_id}=="0x0000"
    ATTR{manf_id}=="0x0204"
    ATTR{func_id}=="0xfe"
    ATTR{pm_state}=="on"
    ATTR{function}=="0x00"

#// –– rest is cut off –– //#
```

As you can see, we need the name of the card which is to be found after ***ATTR{prod_id1}*** so its „DOSCH-AMAND“. On a original DOSCH-AMAND COM-ON-AIR TypeII-Card it would probably be something like „DECTDataDevice“. With this infos we create a *99-dect.rules* in */etc/udev/rules.d/* via the editor nano, or as i prefer vi (in fact you can use any editor you want e.g. kate which is the most easiest to handle) and put in the following rules:

```
# first entry is for “original“ DOSCH-AMANND-Cards
ACTION=="add", SUBSYSTEM=="pcmcia", ATTR{prod_id1}=="DECTDataDevice", RUN+="/
bin/mknod /dev/coa --mode 666 c 3564 0"
# second entry is for TypeIII-Cards like the Ascom Voo:doo
ACTION=="add", SUBSYSTEM=="pcmcia", ATTR{prod_id1}=="DOSCH-AMAND", RUN+="/bin/
mknod /dev/coa --mode 666 c 3564 0"
```

Now you can plug in the card without having to load the kernelmodule by hand each time and also the device-nodes in */dev/coa* will be created automaticially. Now your card should be plug&play.

### Install deDECTed driver and software on BackTrack 4 pre final

This fist step should only be used if youre using a harddisk-install of BackTrack4. We update the packet-manager and upgrade BT4pf with the latest security-patches which may take a little while depending on your internet-connection.

```
apt-get –y update && apt-get –y upgrade
```

Normally you dont need to install anything on BackTrack 4 pre final and muts will add the necessary packages to the BackTrack4-repositories but you do it pretty much the same as on BackTrack3 just without the need to add anything despite the deDECTed-software. So first of load the PCMCIA-kernel-driver.

```
insmod /lib/modules/2.6.29.4/kernel/drivers/pcmcia/pcmcia.ko
```

After inserting the PCMCIA-Kernel-drivers checkout the deDECTed-source from the repository.

```
svn co https://dedected.org/svn/trunk deDECTed
```

Compile the software and drivers. After that a new device node.

```
cd deDECTed/com-on-air_cs-linux
make && make –C tools
make node
```

Now we insert the kernel-module/driver.

```
insmod com_on_air_cs.ko
```

And then we check the dmesg-log if it has been loaded sucessful.

```
dmesg | tail –n 37
```

You need to download and install the decoder for the .IMA to .WAV files yourself.

```
mkdir /usr/src/g72x
cd /usr/src/g72x/
wget http://www.ps-auxw.de/g72x++.tar.bz2
bzip2 -d g72x++.tar.bz2
tar xfv g72x++.tar
cd g72x
./build.sh
```

Now copy the *decode-g72x* binary into the folder of the deDECTed-tools
You also need to add the *decode.sh* script which uses sox, decode-72x and lame to create .wav/.mp3 files.

Now you‘re done and can continue with deDECTion.

For the usage you can also check my short video on youtube.

## 0x05 - deDECTion

So after meeting the prequisites for this experiment we can now start to pentest our cordless-DECT-phone at home.

For testing-purpose i did a call to a service that tells you the current time which can be reached via **+49 30 2 555 555 7**.

Since i am running short on time i point you to my youtube-video Which shows a session in detail.

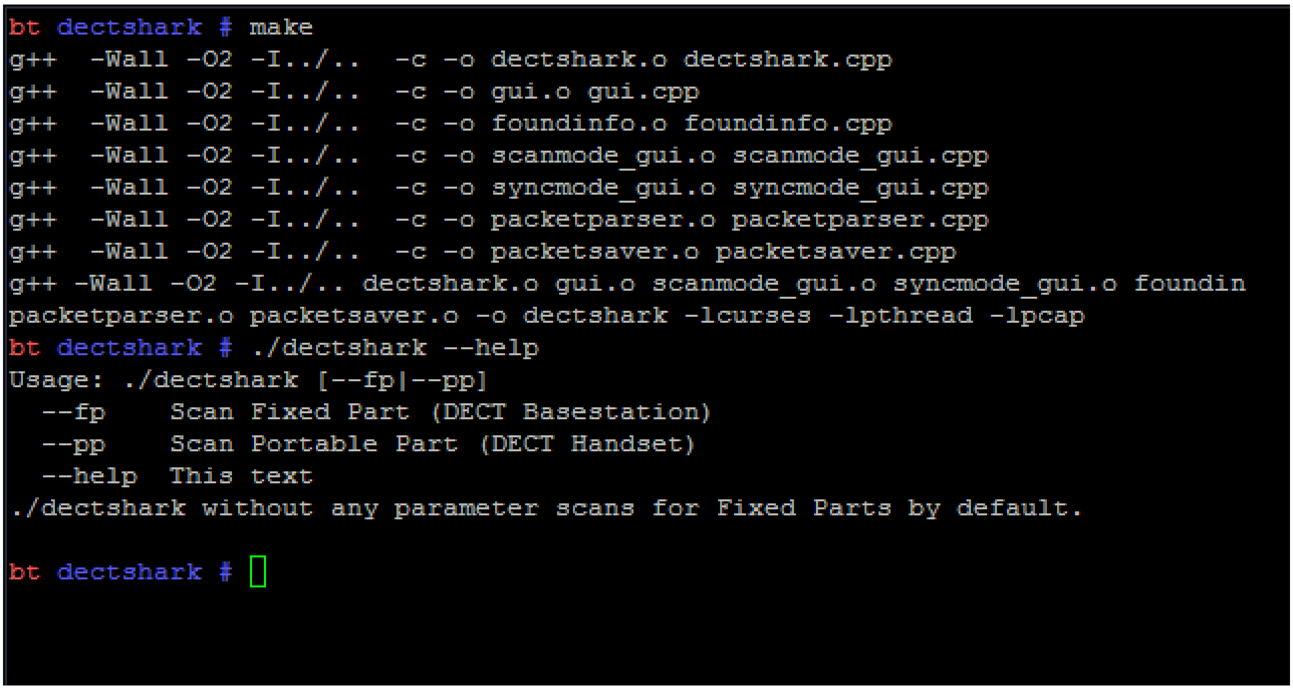
To make it short ist just *fpscan* scanning for base-stations and you will get more information if you use *verb* to turn on verbosity. After that use the *ignore <rfpi>* to ignore all bases-stations but yours. You can find out which is yours via using *callscan* and then make a call so you see what station your call is on. Finally start *autorec* which will synch into a call if it finds one and dump the snifflog into pcap-files.

Using the *decode.sh*-script you can easily get the payload/speec out of the dump-files.

## 0x06 - plugins/tools

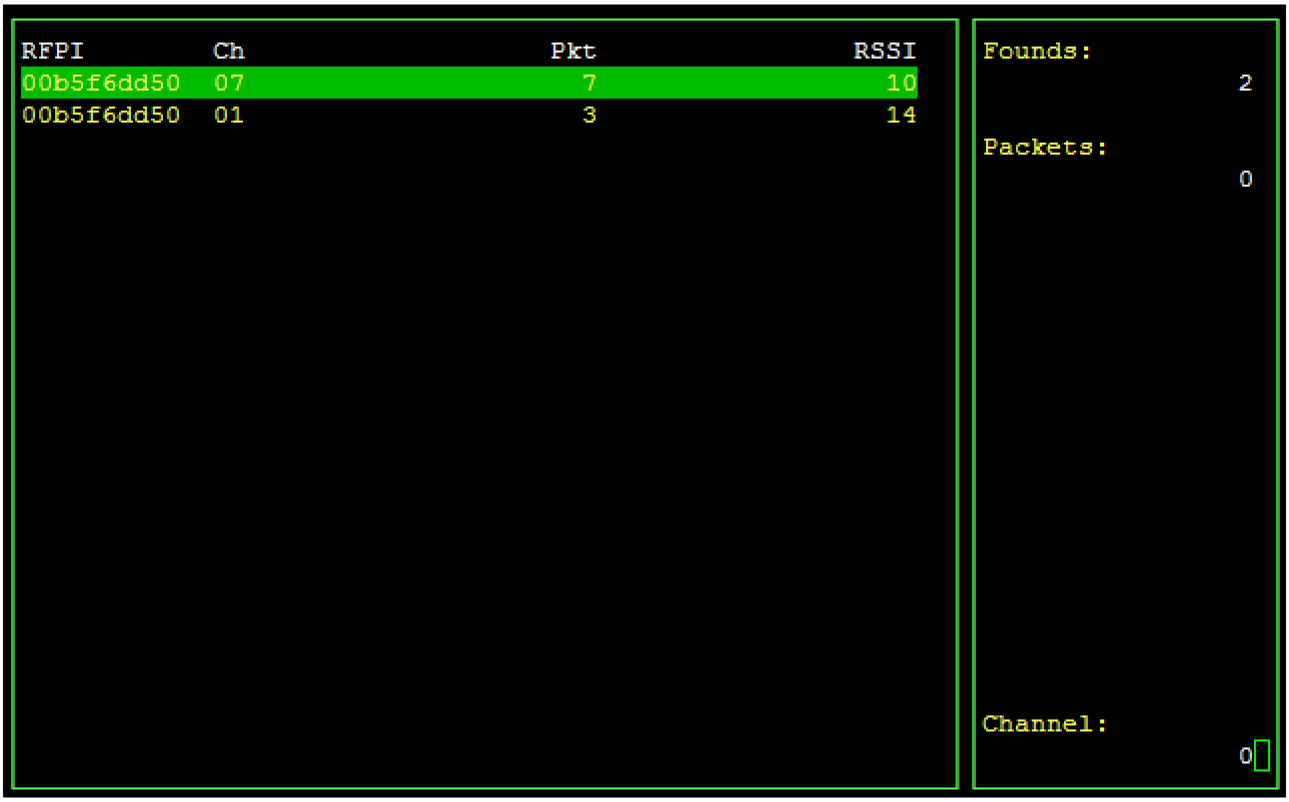
### DECTshark

There is also a nice tool called dectshark in the ***tools/dectshark*** dir which can be compiled via ***make***. Help is available via the ***./dectshark --help***



```
bt dectshark # make
g++  -Wall -O2 -I../..  -c -o dectshark.o dectshark.cpp
g++  -Wall -O2 -I../..  -c -o gui.o gui.cpp
g++  -Wall -O2 -I../..  -c -o foundinfo.o foundinfo.cpp
g++  -Wall -O2 -I../..  -c -o scanmode_gui.o scanmode_gui.cpp
g++  -Wall -O2 -I../..  -c -o syncmode_gui.o syncmode_gui.cpp
g++  -Wall -O2 -I../..  -c -o packetparser.o packetparser.cpp
g++  -Wall -O2 -I../..  -c -o packetsaver.o packetsaver.cpp
g++ -Wall -O2 -I../.. dectshark.o gui.o scanmode_gui.o syncmode_gui.o foundin
packetparser.o packetsaver.o -o dectshark -lcurses -lpthread -lpcap
bt dectshark # ./dectshark --help
Usage: ./dectshark [--fp|--pp]
  --fp    Scan Fixed Part (DECT Basestation)
  --pp    Scan Portable Part (DECT Handset)
  --help  This text
./dectshark without any parameter scans for Fixed Parts by default.

bt dectshark # 
```

The above picture shows how dectshark is to be compiled under BackTrack3.



```
RFPI         Ch                      Pkt                RSSI    Founds:
00b5f6dd50   07                        7                  10                  2
00b5f6dd50   01                        3                  14
                                                                Packets:
                                                                                0



                                                                Channel:
                                                                                0
```

Here you can see my DECT-base-station on channel 07 and the DECT-phone on channel 01 after calling the time-telling-service.

### metasploit plugin

There is also a metasploit plugin included in the deDECTed-Software. Recording is buggy but loggin phone -calls works. To install it just go into the */usr/src/dect/deDECTed/metasploit-dect* and take a look in the README which tells you to do the following.

```
1. Copy coa.rb [msf directory]/lib/msf/core/exploit/
2. Edit [msf directory]/lib/msf/core/exploit.rb and add require 'msf/core/
exploit/coa'

Example Scanner Modules:

1. Create [msf directory]/modules/auxiliary/scanner/coa/ directory
2. Copy call_scanner.rb and station_scanner.rb to the above directory
```

Unfortunately it did not work for me on BackTrack3 and i got several errors. After examining the code and comparing it to other Framework3-Plugins i was able to some little changes to make it work (code is attached). I had a conversation with H.D. Moore who told me that this is okay for short-time but he is working on a complete rewritte of the code and will add this to the Metasploit-Framework-3-repos soon. So plz check via *msfupdate* before using the attached code since it will be probably outdated.
http://lists.gnumonks.org/pipermail/dedected/2009-September/000719.html

### WireShark plugin

This has no more to be compiled since WireShark after v1.2.0 has built in support for DECT since Timo Boettcher submitted the patch to the wishlist some time ago. A sample pcap-file can also be found there
http://wiki.wireshark.org/SampleCaptures#head-b790c6f5019c289abdb35ea5d4c98b2ea467aaeb

So if you want to use that on BackTrack3 just download and install the latest WireShark-Version from www.wireshark.org or just use the latest windows-install if youre really lazy.

### kismet plugin

I pretty much followed the hints from the deDECTed homepage to compile the plugin which can be found at https://dedected.org/trac/wiki/COM-ON-AIR-Kismet. After fiddling with it it worked well. One has just to note that ~ triggers the menue in kismet-new-core and that the plugin has to be loaded via the menu to make it really work fine.

### realtime listeing patch

You can rewrite a patch which enables you to listen to phone-calls in realtime if you want. This one can be found on the deDECTed-mailinglist:
http://lists.gnumonks.org/pipermail/dedected/2009-January/000205.html

The file `audioOTF-patch-revision38-v0.2.tar.bz2` has to be unpacked and ist content copied into the deDECTed-tools folder. But be aware that this code is for an revision which was out at january 2009 so ist not up to date. Since licenseing-problems arose because of the codec used in that patch it was not included to the official-repositories. So if you want that feature look at its code and make the necessarry changes.

## 0x07 - references & links

**deDECTed-Homepage**
https://dedected.org/trac
http://lists.gnumonks.org/pipermail/dedected/
https://dedected.org/trac/report

**List of phones and their vulnerability**
http://gigaset.com/shc/0,1935,de_de_0_168074_rArNrNrNrN,00.html
http://www.gigaset.com/repository/1675/167555/Gigaset_DECT_Verschluesselung.pdf
https://dedected.org/trac/wiki/ListOfPhones
https://dedected.org/trac/blog/list-of-phones-progress

**External antennas**
http://www.mydarc.de/dl7afb/projects/DECT-WIFI-Antennas.htm
http://www.brennpunkt-srl.de/

**Test PCAP-files**
https://dedected.org/trac/ticket/3
http://www.lessradiation.co.uk/RFPI_01_14_71_59_e0.pcap (Sagem D16T)
http://wiki.wireshark.org/SampleCaptures#head-b790c6f5019c289abdb35ea5d4c98b2ea467aaeb

**DECT-Standart and infos**
http://www.dectweb.com
http://en.wikipedia.org/wiki/Digital_Enhanced_Cordless_Telecommunications
http://de.wikipedia.org/wiki/Digital_Enhanced_Cordless_Telecommunications
http://wireless.subsignal.org/index.php?title=Vergleich_DECT_und_WLAN
http://www.etsi.org/WebSite/homepage.aspx
https://dedected.org/trac/wiki/protocol
https://dedected.org/trac/wiki/DSAA-Reversing
http://www.datenschutz-praxis.de/lexikon/r/rfpi.html
https://www.fehcom.net/fh-frankfurt/vorlesungen/2008_WS/itsec/vortraege/DECT-ppt.pdf
https://www.fehcom.net/fh-frankfurt/vorlesungen/2008_WS/itsec/vortraege/FI-DECT_ffa.pdf
http://www.ralf-woelfle.de/elektrosmog/technik/dect_2.htm
http://broadcasting.br.funpic.de/dect_inside_gigaset_repeater.html

**deDECTed-Tutorials**
http://www.ccc-mannheim.de/wiki/Dedected
http://www.wardriving-forum.de/wiki/DeDECTed
http://www.dev-tec.de/2009/01/30/dedected-howto-fur-backtrack-3/
https://dedected.org/trac/wiki/COM-ON-AIR-Kismet
https://dedected.org/trac/ticket/1
http://www.youtube.com/watch?v=vAZLZ8dMIL0

**Linux-LiveCDs for deDECTion (note on BackTrack4 the module will be added soon)**
BackTrack3 and 4prefinal - www.remote-exploit.org
BackTrack remote-exploit-wiki - https://wiki.remote-exploit.org/backtrack/
BackTrack offsec-wiki - http://backtrack.offensive-security.com/index.php/Main_Page
BackTrack4 german review - http://blog.tuxpost.de/2009/02/11/backtrack-4-beta-freigegeben/
BackTrack4 german tutorials - http://backtrack.1rss.de/
Chaox - http://blag.chaox.net/

**Presentation on the 25th Chaos Comunication Congress in Berlin**
http://www.computerbild.de/videos/DECT-Sicherheitsluecken-aufgedeckt-4068888.html
http://chaosradio.ccc.de/cre102.html
https://dedected.org/trac/wiki/25C3
http://www.tis-gmbh.de/produkte/dect-rfpi-sniffer-pro/
http://www.tis-gmbh.de/fileadmin/TIS_PDF/PInfoRFPIsnifferpro_de.pdf

**Press-related**
http://www.golem.de/0812/64331.html
http://www.mitternachtshacking.de/blog/807-25c3-dect
http://www.mathias-schindler.de/2008/12/29/dect-25c3/
http://idw-online.de/pages/de/news295118
http://www.heise.de/newsticker/25C3-Schwere-Sicherheitsluecken-beim-Schnurlos-Telefonieren-mit-DECT--/meldung/120988
http://news.magnus.de/sicherheit/artikel/dect-ist-leicht-abhoerbar.html
http://it.slashdot.org/article.pl?sid=08/12/30/133222
http://www.focus.de/digital/handy/schnurlostelefone-dect-geraete-oft-nicht-abhoersicher_aid_358946.html
http://www.theregister.co.uk/2008/12/31/dect_hack/
http://www.dect.org/UserFiles/file/Press%20releases/DF_Press%20Information_DECT%20Technology_01132009.pdf
http://www.n-tv.de/incoming/DECT-Telefone-unsicher-article48968.html
http://www.heise.de/newsticker/DECT-Abhoerkarte-ist-ausverkauft--/meldung/122230
http://www.heise.de/security/Bundesdatenschutzbeauftragter-warnt-vor-Risiken-bei-DECT-Telefonen--/news/meldung/122033
http://planetopia.de/archiv/2009/planetopia/01_25/1_auswahl.html
http://frontal21.zdf.de/ZDFde/inhalt/3/0,1872,7505859,00.html

## 0x08 - credits

the team from wardriving-forum.de
the ascom-company for producing affordable DECT-cards which make this fun working
the participants of the deDECTed-Mailinglist for Ascom Voo:doo betatesting
Benjamin Schrödl from dev//tec-blog
Muts Maharoni a.k.a. muts from offensive-security.com for testing in the U.S.

and of course

Andreas Schuler, Erik Tews and Ralf-Philipp Weinmann from the deDECTed.org-Team for their stunning work and their presentation on the 25c3 and for their effort to develop the drivers for Type III DECT-cards.