

HACK THE SIGNAL

HACK THE SIGNAL

A curious hacker's guide to hardware
& protocol reverse engineering

NIEL NIELSEN

SEC1.DK

HACK THE SIGNAL

HACK THE SIGNAL

A curious hacker's guide to hardware
& protocol reverse engineering



NIEL NIELSEN

SEC1.DK

Contents

1	Preface: Before You Touch Anything	1
1.1	Who this is for	1
1.2	The hardware thread	1
1.3	How wrong turns work here	2
1.4	Building this book yourself	2
1.5	A note on cost	2
2	What's Inside a Device	5
2.1	The anatomy of an embedded system	5
2.2	The nRF52840 in particular	5
2.3	Talking to the ChameleonUltra	6
2.4	Reading the datasheet (without drowning)	6
2.5	Hands-on: your first diagnostic session	7
3	Your First RFID Scan	9
3.1	Two frequency worlds	9
3.2	What a scan actually does	9
3.3	Running your first scans	10
3.4	LF scanning	10
3.5	Reading a MIFARE Classic card	11
3.6	Your first emulation	11
3.7	Understanding what you are (and are not) doing	11
3.8	Slot management	11
4	Getting the Firmware Out	13
4.1	The debug interface: SWD	13
4.1.1	Connecting SWD to the nRF52840	13
4.1.2	Read-out protection	14
4.2	The bootloader route: DFU and UF2	14
4.2.1	Nordic DFU	14
4.2.2	UF2 bootloader: drag-and-drop firmware	14
4.3	Physical flash access: SPI NOR	15
4.4	FEL mode: Allwinner's back door	15
4.5	What to do with a firmware binary	16
5	Disassembly Without Fear	17
5.1	Tools	17
5.2	ARM Thumb2 in ten minutes	18
5.3	Finding the entry point	18
5.4	The ST-Link bootloader: five theories, one answer	19
5.4.1	What the disassembly actually said	20

5.5	Recognising peripheral accesses	20
5.6	Hands-on: finding the NFCT peripheral in ChameleonUltra	20
6	Closed-Source Device Teardown	23
6.1	What we know before we start	23
6.2	Cracking the installer	23
6.3	Reading the ELF files	24
6.4	The board photo contradiction	25
6.4.1	Three hypotheses, not killed	25
6.5	Reading the STM32 blob	25
6.6	The hardnested tables	26
6.7	Hands-on: compare with ChameleonUltra	26
7	USB Protocol Archaeology	27
7.1	How USB works (the parts that matter)	27
7.2	Capturing with usbmon	28
7.3	The ZD1211 case: four layers of indirection	28
7.3.1	Step 1: usbmon capture under Windows (via Wireshark and USBPcap)	28
7.3.2	Step 2: identify the command structure	29
7.3.3	Step 3: brute-force bRequest scan	29
7.3.4	Step 4: the downgrade protection twist	30
7.4	Hands-on: capture ChameleonUltra's own USB traffic	30
8	LF Protocols in Detail	31
8.1	The 125 kHz carrier	31
8.2	Modulation schemes	31
8.2.1	ASK: Amplitude Shift Keying	31
8.2.2	FSK: Frequency Shift Keying	32
8.2.3	PSK: Phase Shift Keying	32
8.3	Manchester and Biphase encoding	32
8.4	EM410x: the ubiquitous read-only tag	32
8.5	EM4x05: reader-talk-first and the debugging journey	32
8.5.1	The EM4305 READ command	33
8.5.2	Building lf_gap.c	33
8.5.3	Dead end 1: RTF was disabled	33
8.5.4	Dead end 2: bit timing nearly 2× off	34
8.5.5	Dead end 3: timeslot too short	34
8.5.6	Root cause 1: the PWM pin release bug	34
8.5.7	Root cause 2: antenna ringing	34
8.5.8	What the journey taught	35
9	HF Protocols and Passive Sniffing	37
9.1	The ISO 14443-A stack	37
9.2	The activation sequence, frame by frame	37
9.3	Authentication in MIFARE Classic	38
9.4	Passive sniffing with ChameleonUltra	38
9.4.1	Running a sniff session	38
9.4.2	What you are seeing	39
9.4.3	Hands-on: sniff your own card against a phone	39
9.5	The hardware limit: load modulation	39
10	MIFARE Classic: The Classic Attack Surface	41

10.1	Memory structure	41
10.1.1	Access bits	41
10.2	The crypto1 cipher	42
10.3	Attack 1: Default key scan	42
10.4	Attack 2: Nested authentication (mfkey32)	42
10.5	Attack 3: Hardnested attack	43
10.6	Autopwn: sector-by-sector key recovery	43
10.7	PRNG type detection	43
10.8	A real-world case: Rejsekort	43
10.9	Loading a full dump	44
11	Relay and Standalone Attacks	45
11.1	Relay attacks: the concept	45
11.1.1	Starting a BLE relay session	45
11.2	RESCAN_REQ and field power management	46
11.3	Standalone mode: design goals	46
11.3.1	Chord gestures	46
11.3.2	The chord detector bug	47
11.3.3	FDS write crash	47
11.3.4	Idle state reset in armed mode	48
11.4	AuthTrace: the complete workflow	48
12	Rooting a Sealed Embedded Linux Device	51
12.1	What is FEL mode?	51
12.2	Finding the FEL pad	51
12.3	Reading the SPI NOR flash	52
12.4	Parsing the firmware	52
12.5	Reading the firmware update format	52
12.6	Credential scan	53
12.7	Injecting a telnet backdoor	53
12.8	What to do with root access	54
12.9	Responsible disclosure checklist	55
13	When the OS Is the Target	57
13.1	What MDE device isolation does	57
13.2	WSL as a blind spot	57
13.3	Building the tunnel	58
13.4	Adding Tor	58
13.5	Microsoft's partial fix	58
13.6	The methodology, generalised	58
13.7	Hands-on: audit your own endpoint	59
14	Contributing Back	61
14.1	The PR model	61
14.1.1	What makes a good PR	61
14.1.2	The PR description	62
14.2	Writing the blog post	62
14.2.1	Tools	63
14.3	Conference submissions	63
14.4	Responsible disclosure	63
14.5	Your contribution loop	64

15 About the Author**65**

Chapter 1

Preface: Before You Touch Anything

This book began as a blog. Posts about pulling firmware from a sealed device, building an RFID reader out of a handful of nRF52840 peripheral registers, disassembling a bootloader to answer a question that nagged at me for weeks. Each post was complete on its own: a problem, the false starts, the fix, what the fix taught me.

What you are holding is those posts reorganised into something with a spine. The problem they collectively answer is: *how do you get started in hardware reverse engineering when nobody hands you a syllabus?*

1.1 Who this is for

You are curious. You can write a small script, read a block of C without panicking, and you have probably taken apart something you should not have. You do not necessarily have a formal background in electronics or security. You have heard the words “RFID attack” or “firmware dump” and wanted to know what they actually mean in practice, from someone who has done them rather than described them abstractly.

This book is not for people who already know this field. It is for people on the outside of the field, looking for a door.

1.2 The hardware thread

Every hands-on section in this book uses one device: the **ChameleonUltra**. It is an open-source, open-hardware RFID research tool built on the Nordic nRF52840, an ARM Cortex-M4 with an integrated 2.4 GHz radio and a dedicated NFC peripheral. It does LF and HF RFID. It reads, emulates, and sniffs. It connects over USB-C and BLE. It costs around \$60 from the usual channels and the full schematic is on GitHub.

The reason I chose one device rather than a toolkit is that depth beats breadth for learning. By the end of this book you will understand one piece of hardware well enough to modify its firmware, debug its peripherals, and extend its protocol support. That skill transfers. The ChameleonUltra is the vehicle, not the destination.

You will need:

- One ChameleonUltra (RfidResearchGroup edition or NielDK fork, both work)
- A USB-C cable
- A Linux machine or VM (the examples use Arch Linux, but any distro works)

- A handful of RFID cards to practise on: a hotel key, a transit card, a parking fob

For Chapters 11 and 12 you will not use the ChameleonUltra at all. Those chapters cover embedded Linux targets and endpoint security. Different territory, same methodology.

1.3 How wrong turns work here

Every chapter in this book includes the dead ends. The hypothesis that looked solid until one byte contradicted it. The firmware bug that crashed the device in a way that pointed nowhere obvious. The five wrong theories about a bootloader, killed in order.

This is deliberate. In real reverse engineering work, the dead ends are where the learning happens. A tidy account that skips from question to answer teaches you the answer. A messy account that shows you *how to eliminate wrong answers* teaches you to think.

When you see a section labelled “Dead End” or a hypothesis killed by evidence, read it carefully. That is the part most textbooks leave out.

1.4 Building this book yourself

The source for this book is Pandoc-flavoured Markdown. To build an epub or PDF from source:

```
# Install Pandoc (Arch Linux)
sudo pacman -S pandoc

# Build epub
pandoc --metadata-file=metadata.yaml --toc --number-sections \
  -o hack-the-signal.epub ch00-preface.md ch01-*.md ... ch13-*.md

# Build PDF (requires texlive)
pandoc --metadata-file=metadata.yaml --pdf-engine=xelatex \
  -o hack-the-signal.pdf ch00-preface.md ch01-*.md ... ch13-*.md

# Or use the build script
chmod +x build.sh && ./build.sh
```

The source lives at sec1.dk alongside the original blog posts it grew from.

1.5 A note on cost

This book is free. If it saves you hours of dead ends or points you at something you would not have found otherwise, a small donation is appreciated but never expected: paypal.me/nieldk

Let's start with what is actually inside the devices you want to hack.

© 2026 Niel Nielsen

Redigering: Niel Nielsen

Korrektur: Niel Nielsen

Bidragydere: Niel Nielsen

Forlag: BoD · Books on Demand, Strandvejen 100, 2900 Hellerup, bod@bod.dk

Tryk: Libri Plureos GmbH, Friedensallee 273, 22763 Hamburg

ISBN: 978-87-430-9123-3

