

Practical Hardware Pentesting

A guide to attacking embedded systems and protecting them against the most common hardware attacks

Jean-Georges Valle



BIRMINGHAM—MUMBAI

Practical Hardware Pentesting

Copyright © 2021 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

Group Product Manager: Wilson D'souza

Publishing Product Manager: Rahul Nair

Senior Editor: Arun Nadar

Content Development Editor: Romy Dias

Technical Editor: Nithik Cheruvakodan

Copy Editor: Safis Editing

Project Coordinator: Neil D'mello

Proofreader: Safis Editing

Indexer: Manju Arasan

Production Designer: Nilesh Mohite

First published: March 2021

Production reference: 1040321

Published by Packt Publishing Ltd.

Livery Place

35 Livery Street

Birmingham

B3 2PB, UK.

ISBN 978-1-78961-913-3

www.packt.com

To my father. I wouldn't be who I am without you.

Contributors

About the author

Jean-Georges Valle is a hardware penetration tester based in Belgium. His background was in software security, with hardware being a hobby, and he then started to look into the security aspects of hardware. He has spent the last decade testing various systems, from industrial logic controllers to city-scale IoT, and from media distribution to power metering. He has learned to attack embedded systems and to leverage them against cloud-scale infrastructure. He is the lead hardware technical expert in an offensive security team of a big four company.

Jean-Georges holds a master's degree in information security and focuses on security at the point of intersection with hardware and software, hardware and software interaction, exploit development in embedded systems, and open source hardware.

I wish to thank my parents for supporting me and loving me unconditionally, Vito and Jon for giving me an opportunity when I needed it, and Ieva for accepting that this book was competing with her for my time and attention.

About the reviewers

Ryan Slauch has been a maker and breaker of things for over 20 years. Ryan got his start in electrical systems, and augmented his learning to include the analog, digital, embedded, software, and cybersecurity fields. He continues to practice and add to his skill sets in his home lab, and this allows him to do what he loves the most: solve problems with technology. When not working with technology, Ryan enjoys traveling around the globe and exploring the less inhabited areas of the Pacific Northwest. His greatest joy is being with his family on their small hobby farm in Washington State, USA.

Neeraj Thakur is a manager in the risk advisory practice of Deloitte and comes with more than 9 years' experience in the area of information and cybersecurity. He holds a master's degree in cybersecurity from the Indian Institute of Information Technology, Allahabad, and has extensive experience in penetration and security testing of various embedded devices and IoT-enabled products. He is a certified ISA/IEC 62443 cybersecurity fundamentals specialist and has worked extensively in the areas of industrial automation and control system security. He has delivered multiple sessions on IoT and ICS security, as well as in the security community, including Nullcon and CySeck. Neeraj is passionate about reverse engineering and security innovations using Python.

Table of Contents

Preface

Section 1: Getting to Know the Hardware

1

Setting Up Your Pentesting Lab and Ensuring Lab Safety

Prerequisites – the basics you will need	4	Small tools and equipment	21
Languages	5	Renting versus buying	23
Hardware-related skills	5	The component pantry	23
System configuration	5	The pantry itself	23
Setting up a general lab	7	The stock	24
Safety	8	Sample labs	25
Approach to buying test equipment	9	Beginner	25
Home lab versus company lab	9	Amateur	26
Approaching instrument selection	10	Pro	27
What to buy, what it does, and when to buy it	11	Summary	27
		Questions	28

2

Understanding Your Target

The CPU block	30	The storage block	34
CPU roles	30	RAM	34
Common embedded systems architectures	31	Program storage	34
		Storing data	35

The power block	35	Analog sensors	41
The power block from a pentesting point of view	35	Digital sensors	42
The networking blocks	36	The actuator blocks	42
Common networking protocols in embedded systems	36	The interface blocks	43
The sensor blocks	41	Summary	43
		Questions	44
		Further reading	44

3

Identifying the Components of Your Target

Technical requirements	46	Continuing system exploration – identifying and putting components in the diagram	54
Harvesting information – reading the manual	47	Opening the Furby	54
Taking a system analysis approach	47	Manipulating the system	54
For our Furby manual	47	Dismantling the Furby	55
Harvesting information — researching on the internet	49	Identifying chips	55
For the Furby	49	Chips in the Furby	56
Starting the system diagram	52	Identifying unmarked/mysterious chips	59
For our Furby	53	Furby — the mystery meat	61
		The borders of functional blocks	68
		Summary	68
		Questions	69

4

Approaching and Planning the Test

The STRIDE methodology	72	Protection of secrets or security elements	79
Finding the crown jewels in the assessed system	74	Reaching the crown jewels – how do we create impacts?	80
Security properties – what do we expect?	77	STRIDE through the components to compromise properties	80
Communication	78	For the example system – the Furby	82
Maintenance	78	Planning the test	85
System integrity and self-testing	79		

Balancing your scenarios	85	Questions	91
Summary	91	Further reading	91

Section 2: Attacking the Hardware

5

Our Main Attack Platform

Technical requirements	96	Flashing the chip	104
Introduction to the bluepill board	97	Putting it into practice for the bluepill	104
A board to do what?	97	Introduction to C	106
What is it?	97	Operators	107
Why C and not Arduino?	98	Types	108
The documentation	99	The dreaded pointer	109
Memory-projected registers	100	Preprocessor directives	110
		Functions	111
The toolchain	100	Summary	112
The compilation process	101	Questions	112
Driving the compilation	102	Further reading	112

6

Sniffing and Attacking the Most Common Protocols

Technical requirements	114	Understanding UART	134
Hardware	114	Mode of operation	135
Understanding I2C	115	Sniffing UART	137
Mode of operation	115	Injecting UART	137
Sniffing I2C	123	UART – man in the middle	138
Injecting I2C	128	Understanding D1W	139
I2C man in the middle	128	Mode of operation	139
Understanding SPI	129	Sniffing D1W	141
Mode of operation	130	Injecting D1W	141
Sniffing SPI	132	D1W – man in the middle	142
Injecting SPI	133	Summary	142
SPI – man in the middle	133	Questions	143

7

Extracting and Manipulating Onboard Storage

Technical requirements	146	Understanding unknown storage structures	151
Finding the data	146	Unknown storage formats	151
EEPROMs	146	Well-known storage formats	152
EMMC and NAND/NOR Flash	147	Let's look for storage in our Furby	153
Hard drives, SSDs, and other storage mediums	147	Mounting filesystems	159
Extracting the data	148	Repacking	160
On-chip firmware	148	Summary	161
Onboard storage – specific interfaces	149	Questions	161
Onboard storage – common interfaces	149	Further reading	161

8

Attacking Wi-Fi, Bluetooth, and BLE

Technical requirements	164	Bluetooth basics	169
Basics of networking	164	Discovering Bluetooth	171
Networking in embedded systems using Wi-Fi	165	Native Linux Bluetooth tools – looking into the joystick crash	175
Selecting Wi-Fi hardware	165	Sniffing the BT activity on your host	178
Creating our access point	165	Sniffing raw BT	179
Creating the access point and the basic network services	166	BLE	182
Networking in embedded systems using Bluetooth	169	Summary	188
		Questions	188

9

Software-Defined Radio Attacks

Technical requirements	190	Understanding and selecting the hardware	191
Introduction to arbitrary radio/SDR	190	Looking into a radio device	192

Receiving the signal – a look at antennas	192	MSK	204
		Getting back to our signal	205
Looking into the radio spectrum	194	Demodulating the signal	206
Finding back the data	198	Clock Recovery MM	210
Identifying modulations – a didactic example	200	WPCR	211
AM/ASK	201	Sending it back	212
FM/FSK	202	Summary	212
PM/PSK	203	Questions	213

Section 3: Attacking the Software

10

Accessing the Debug Interfaces

Technical requirements	218	Very hard – JTAGulating	228
Debugging/programming protocols – What are they and what are they used for?	218	Using OpenOCD	231
Legitimate usage	218	Installing OpenOCD	232
Using JTAG to attack a system	219	The adapter file	233
		The target file	234
Finding the pins	224	Practical case	240
The PCB "plays nicely"	225	Summary	246
A bit harder	228	Questions	247

11

Static Reverse Engineering and Analysis

Technical requirements	250	Dump structure – the bluepill as an example	257
Executable formats	250		
Understanding operating system formats	251	Analyzing firmware – introduction to Ghidra	258
Dump formats and memory images	256	Getting to know Ghidra with a very simple ARM Linux executable	258

Going into second gear – Ghidra on raw binaries for the STM32	268	Reversing our target function	277
First identification pass	272	Summary	278
		Questions	279

12

Dynamic Reverse Engineering

Technical requirements	282	Exploring the most useful ARM instructions	291
What is dynamic reverse engineering and why do it?	282	Using dynamic reverse engineering – an example	296
Leveraging OpenOCD and GDB	283	First Ghidra inspection	297
GDB? But... I know nothing about it!	285	Reversing the expected password	297
Understanding ARM assembly – a primer	287	Of course, I aced the test	307
General information and syntax	288	Summary	308
		Questions	308

13

Scoring and Reporting Your Vulnerabilities

Scoring your vulnerabilities	312	Report quality	318
Being understandable to everyone	316	When engineers do not want to re-engineer	319
Building your report template	316	Summary	322
Usage of language in a report	317	Questions	322

14

Wrapping It Up – Mitigations and Good Practices

Industry good practices – what are they and where to find them	324	Common problems and their mitigations	328
OWASP IoT top 10	324	Establishing a trust relationship between the backend and a device	328
The CIS benchmarks	327	Storing secrets and confidential data	330
NIST hardware security guidelines	328		

Cryptographic applications in sensitive applications	330	What about now? Self-teaching and your first project	332
JTAG, bootloaders, and serial/UART interfaces	331	Closing words	333

Assessments

Chapter 1	335	Chapter 8	341
Chapter 2	335	Chapter 9	342
Chapter 3	336	Chapter 10	342
Chapter 4	338	Chapter 11	343
Chapter 5	338	Chapter 12	344
Chapter 6	340	Chapter 13	345
Chapter 7	341		

Other Books You May Enjoy

Index
