Know your devices: Why penetration testing is an essential security process





Who is





X-Force Red is an autonomous team of veteran hackers, within IBM Security, hired to break into organizations and uncover risky vulnerabilities that criminal attackers may use for personal gain.

X-Force Red offers offensive security services which include penetration testing, vulnerability management services, red teaming, code reviews, static analysis and vulnerability assessments.

Their goal is to help security leaders identify and remediate security flaws, covering their entire digital and physical ecosystem.

- -200+ people globally
- -Decades of experience personal and professional
- Researchers, developers, engineers, programmers, business leaders, thought leaders

-Testified before Congress, present at top conferences, built renown hacker communities



IoT Security Challenges

Common Security Challenges

- Legacy, unpatched hardware
- Product functionality information left on production device
- Development tooling / logging left on production device
- Default credentials
- Millions of end user devices, lack of visibility
- Engineers are not security experts
 - Reliance on manufacturers to release patches
 - Assumption of security of 3rd party components
 - Complexity of inter-component communications



Testing for IoT – When / Why / How



Define scope and objectives

Data collection and device reconnaissance

Dynamic and static testing for App/OS/HW

Reverse engineering and threat modeling **Report and follow**

up

- Costs and risks increase throughout production life cycle
 - Better to eliminate issues in initial design
 - Remediation requires push out to installed user base
 - May not be possible if certified device

Meanwhile, there are so many components.

Circuit Board Interfaces Processor Flash & Logic Anti Tamper WiFi / Zigbee / BLE Flash & RAM

Sensors

Buttons



All can be targeted...

- Architecture and Design
- Electronics (Schematic and PCB)
- Storage Systems
- Cryptography
- Anti-counterfeiting
- Communications (WIFI, Bluetooth, ZigBee etc)
- Software (implementation and SDLC)
- Firmware
- API
- Cloud services
- Build Standards
- Supply Chain



Case Study - Hardware penetration test on a high end IoT device

Objective: Gain access to file system to retrieve hash of root user.

Description: Stringent software security and access controls prevented typical penetration test "privileged escalation" attacks access to file system.

Solution: Rather than continuing a software and network services attack, a fine pitch rework station was used to remove the BGA (ball grid array) 32GB Flash from the IoT device.

Most flash can simulate an SD card interface. As such, a little work under the Microscope resulted in only 7 wires being connected from the flash to a standard SD card adaptor.

This was connected to a standard laptop and the attacker gained unrestricted access to the file system and the desired root user password hash.

Note: Software safeguards were not sufficient.



Case Study - Hardware penetration test on secure phone

Obscurity through to full compromise

Objective: Full compromise of device.

Description: Operating system proved to be secure and attempts to compromise boot process failed due to robust secure boot controls.

Solution: Examination of the device PCB (circuit board) itself showed a number of unloaded 0402 components. This is often an indicator that "option resistors" have been included in the design. Option resistors are options set through the loading or not loading of certain components on a circuit. Often this includes enabling debugging mode by loading a part to allow developers more access.

Each of the "option resistor" pads were probed during a reboot of the device. The result was as expected and one of the unloaded components was indeed an option resistor that enabled a debug mode of operation.

This debug mode option resistor was checked by the BIOS during very early stages of a reboot, if enabled, it would allow access to a BIOS screen.



With access to the BIOS screen the attacker disabled secure boot and the previous attacks to compromise the boot process were then trivial to apply.

Note: A tiny hidden solder pad was enough to allow a thorough attacker to fully compromise a "secure" mobile phone.

Case Study – Capture & Replay of proprietary encrypted RF signal

Objective: Demodulate, decode and decrypt to enable spoofing.

Description: Unknown RF signal controlled "pushbutton" functions on an IoT access control device -Open/Close/Etc.

Solution: Capture signal with SDR. Compare repeated signals to work out packet structure/CRC etc. Identify encrypted payload and apply cryptographic attacks. Re-encode and transmit with COTS SDR.



'Raiden' Glitching Tool

Hardware

- In-house developed opensource glitching tool – 'Raiden'
- Based on off-the-shelf FPGA dev board
- Extremely accurate and high speed – 2.5 - 10ns resolution (hardware dependant).
- Selectable glitch voltage levels
- External trigger module
- Device RESET/Power Cycle
- Python API (via serial UART)
- Low Cost
- Simple to use

X-Force Red has developed its own glitching tool for fault injection attacks. Designed to use off the shelf technologies that mirror those that would be available in the real world, this allows us to realistically simulate attacks likely to be used against our clients' devices.

Glitching/fault injection is a very powerful technique for bypassing security checks and causing unexpected behavior in microcontrollers. By carefully controlling the timing and intensity of the faults being induced, an attacker can potentially cause the system to behave in a way that gives them advantage.

MSO7014 Wed January 22 16:25:18 2020



R & D – power analysis and glitching

- Provide some kind of feedback that chip is still alive
- Avoid destroying your only sample
- Confirm desired effect is even possible in a 'safe' environment
 - Enable bootloader
 - Enable SWD
 - Bypass read protection
 - Bypass code signing

A chip doing nothing, no power...



A chip powered up, running user code...



A chip powered up, running bootloader...



The magic shunt resistor!

Position of the shunt in the circuit for current measuring



The magic shunt resistor!



DPA – Differential Power Analysis

- Capture multiple runs
 - Compare power usage
 - Reveal bits in crypto operations

- Compare timing
 - Reveal how far an operation got e.g. password check



MSO7014 Wed February 12 16:09:33 2020

When to glitch?

When to glitch? A common approach

- Capture multiple runs with persistence
 - Switch modes
 - Bootloader vs User code
 - Glitch at point of divergence



MSO7014 Wed February 12 16:09:33 2020

Reconnaissance

- Extract bootloader
 - SWD read out ROM
 - Reverse with Ghidra / IDA / Binary Ninja

											¥ A
Library function 📃 Regular function	Instruction 📃 Data 📕 Une	xplored E	xternal symbol 📕 L	Lumina fund	ction						
📝 Functions window 🛛 🕲 🕲	8 1	IDA View-A	🕲 🚺 Hex View-1	8 A	Structures	8 🗄	Enums	8	Imports	8 🛃	Exports
Function name			var_1C= -	Ox1C							
f sub_1FFF00EC			; FUNCTION	N CHUNK AT	ROM: 1FFF00F4 S	IZE 0000000	08 BYTES				
F sub_1FFF0140			LDR R	R4-R7,LR} 4, =0x4003C	000						
F sub_1FFF01E8			LDR R	P, SP, #0x1 0, [R4]	4						
f sub_1FFF0278			BICS R	5, #0x40 ; 0, R5							
J sub_1FFF02FC			LDR R	0, [R4] 0, =crp_on_	flash						
f sub_1FFF0314			LDR R	1, [R0]							
f sub_1FFF03AA			STR R	1, [R0, #1]	000						
f sub_1FFF03DE			ORRS R	0, [R4]							
f sub_1FFF0424			LDR R	0, -DESDO							
Juser_code_valid			LDR R	0, [R0]	BCD						
flow_check			BEQ 1	oc_1FFF04B2							
f sub_1FFF0640				_		10 - 4 X	*	-			
F sub_1FFF0660			LDR RO, [S	R4]		loc IFFF	0482	1			
J sub_1FFF06EC			STR RO, [F	R4]		LDR	RO, -DESHC				
f sub_1FFF085C						LDR	RO, [RO]	50			
f sub_1FFF0C88						LDR	R7, -01400483				
f sub_1FFF0C98						CMP	RO, RI				
f sub_1FFF0CA4									-		
Line 1 of 19				LDR	R0. =0x1234	5678			10		
👬 Graph overview 🗆 🖸 🙆 🔕			loc_1FFF04B0	F04B0 LDR	RO, [R7, 10	IOI Loc_1	FFF04D2 R1, #0xC				
			<u> </u>	LDR STR MOVS STR	RO, [RO] RO, [R6,#02 RO, #0 RO, [R7,#02	LDR STR LDR STR	RO, -01500 R1, [SP, 00] R0, [R0] R0, [SP, 00]	x28+var_28	1		
				в	loc_iFFF04F	2 LDR STR LDR ADD	R0, =off_11 R1, [SP,#0 R0, [R0] R1, SP, #0	FFF1FF8 x28+var_20 ; unl x28+var_1C	k_1FFF1E80		
電	100.00% (-639,183) (158,74	10) 000000F8	1FFF00F8: flow_c	check-38A	A (Synchroni	zed with	Hex View-	1)			

When to glitch? Add precision: Breakpoint

IDA - LPC11U24-bootloader-db32.idb (LPC11U24-bootloader.bin) /Users/adamlaurie/software/unpa	acked/nxp-lpc11ux	x-bootloader/LP	C11U24-boot	loader-db32.i	db - Running			
🗢 🗸 🔿 🗸 🐴 🍓 🔖 🐛 🙍 : 🖬 🕥 : 🗐 🗷 🗽 🔎 🐼 🦣 🖨 🞯 🔛 : 🚮	af 's†▼	🖈 🖬 🗙 :		Remote GE	DB debugger	٢		
					•			
unction 📃 Regular function 📕 Instruction 📃 Data 📕 Unexplored 🗾 External symbol 📕 Lumina functi	ion							
IDA View-PC, Breakpoints, General registers, Modules, Threads, Hex View-1, Stack view	🛛 🕅 St	tructures	🛞 🔃	Enums				
IDA View-PC 🛛 🕲 🛐 Breakpoints			👿 General	registers				
ROM: 1FFF03DE ROM: 1FFF03DE ROM: 1FFF03DE ROM: 1FFF03DE ROM: 1FFF03DE ROM: 1FFF03DE ROM: 1FFF03DE ROM: 1FFF03DE ROM: 1FFF03DE ROM: 1FFF03DE DDR RO, =unk_4003C000 ROM: 1FFF03DE DDR RO, =unk_4003C000 ROM: 1FFF03DE DDR RO, =unk_4003C000 ROM: 1FFF03DE DDR RO, =unk_4003C000 ROM: 1FFF03BE DDR R2, [R0] ROM: 1FFF03BE DDR R2, [R0] ROM: 1FFF03BE DDR R2, [R0] ROM: 1FFF03BE DDR R2, =unk_430 ROM: 1FFF03BE DDR R4, [R2] ROM: 1FFF03BE DDR R2, =unk_5E0 ROM: 1FFF03BE DDR R2, [R0] ROM: 1FFF03FE DDR R2, [R0] ROM: 1FFF03FE STR R2, [R0] ROM: 1FFF03FE STR R2, [R0] ROM: 1FFF03FE STR R2, [R0]		1	R0 FFFFFFF R1 1000050 R2 0000005 R3 90850000 R4 FFFF984 R5 FFFF9884 R6 40048000 R7 40048300 R8 08080000 Path S S GDB rd	emote process	\$>			
ROM: 1FFF03FC LDR R0, [R0] ROM: 1FFF03FC LDR R0, [R0]			🕤 Threads					
ROM:1FFF0400 STR RO, [R1,#(unk_10000) ROM:1FFF0402 BL sub_1FFF03AA Dupping			Decimal	Hex Sta	te	Name		
ROM: 1FFF0406 LSLS R0, R4, #0x1F RUITINING ROM: 1FFF0408 BNE loc_1FFF0420			57005	DEAD Re	ady	FFFF		
37,11) (612,26) 000003F8 1FFF03F8: sub_1FFF03DE+1A (Synchronized with								

Breakpoint



Breakpoint



Want to learn more?

- Visit the X-Force Red homepage: <u>https://www.ibm.com/security/services/offensive-security-services</u>
- Read the whitepaper: https://www.ibm.com/downloads/cas/MY6L2O89

- Watch the video: https://www.ibm.com/security/services/iot-testing

Thank you

Follow us on:

ibm.com/xforcered

@xforcered

securityintelligence.com

ibm.com/security/community

xforce.ibmcloud.com

youtube.com/ibmsecurity

© Copyright IBM Corporation 2019. All rights reserved. The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. Any statement of direction represents IBM's current intent, is subject to change or withdrawal, and represent only goals and objectives. IBM, the IBM logo, and other IBM products and services are trademarks of the International Business Machines Corporation, in the United States, other countries or both. Other company, product, or service names may be trademarks or service marks of others.

Statement of Good Security Practices: IT system security involves protecting systems and information through prevention, detection and response to improper access from within and outside your enterprise. Improper access can result in information being altered, destroyed, misappropriated or misused or can result in damage to or misuse of your systems, including for use in attacks on others. No IT system or product, should be considered completely secure and no single product, service or security measure can be completely effective in preventing improper use or access. IBM systems, products and services are designed to be part of a lawful, comprehensive security approach, which will necessarily involve additional operational procedures, and may require other systems, products or services to be most effective. IBM does not warrant that any systems, products or services are immune from, or will make your enterprise immune from, the malicious or illegal conduct of any party.







